

Analysis of a Randomized Space Filling Algorithm and its Linear Program Formulation

Ka.Shrinivaasan (ka.shrinivaasan@gmail.com)

March 19, 2013

Abstract

In this article a grid gadget with binary contents is defined and a parallel algorithm to fill the grid coordinates with 1s is presented and its Linear Programming Formulation is analysed

1 Definition of a 2-dimensional Space Filling Grid Gadget

Let S be a finite 2 dimensional grid space with coordinates denoted by (x,y) to uniquely choose a point in the grid. Let the length of the finite space S be $2^{\log(N)/2}$ for $N > 0$ and breadth be $2^{\log(N)/2}$ making it a square grid. Thus the total number of points in the grid are $2^{\log(N)/2} * 2^{\log(N)/2} = N$. Thus any tuple that uniquely identifies a point in the grid ranges from $(0,0)$ to $(2^{\log(N)/2}, 2^{\log(N)/2})$. Initially each point in the grid is set to 0. Goal is to find an efficient algorithm to set the maximum number of points in the grid to 1s (flipping 0 to 1 in that coordinate) or it can be above a threshold $K < N$. Some natural phenomena which use this grid filling algorithm:

1. Rainfall in which cloud "computes" the random coordinates on the earth to place the raindrops that too in parallel (this is an indefinitely parallel random coordinate computation which will be humongous task even for a supercomputer to simulate)
2. Randomly strewn sand with particles scattered at random points which again has a natural indefinite parallel computation of random coordinates.

2 Parallel Randomized Algorithm to Fill the Grid Gadget

Naive algorithm to fill the grid is to sequentially scan the rows and set the coordinates to 1 from 0 till the threshold is reached. This gives a polynomial or linear time algorithm. This can be parallelized with a parallel pseudorandom generator that outputs pseudorandom bits in parallel. There are pseudorandom generators that can output pseudorandom bits in parallel through multiple processors, for example PRG explained in [Tygar-Reif] which is a Randomized NC algorithm (RNC) and the Chaos theoretic Pseudorandom generator variant of Tygar-Reif which used Lehmer's generator in each processor.

Let the number of parallel processors be $k * \log N$ in Tygar-Reif RNC algorithm. By executing this parallel PRG, $k * \log N$ pseudorandom bits are generated in parallel. This $k * \log N$ bit string can be split into k strings of $\log N$ bit length each. Each of the $\log N$ bit string can be split into ordered pair 2-tuples of the form $\langle \text{size}(\log N/2), \text{size}(\log N/2) \rangle$. This tuple (x, y) denotes a coordinate point in the grid since grid is a square of side $\log N/2$. Thus k ordered pairs are generated in parallel

by Tygar-Reif RNC algorithm. Thus k coordinates in the grid can be set to 1 in parallel. Thus instead of a linear time sequential algorithm ,a randomized NC parallel algorithm for grid filling is obtained.

3 Linear Programming and Grid Filling

Each coordinate in the grid can be denoted by a unique variable x_i where $1 < i < N$. Thus a linear program objective function can be defined as

$$\text{Maximize}(x_1 + x_2 + \dots + x_N) \quad (1)$$

with N constraints being

1. $x_1 \leq 1$
2. $x_2 \leq 1 \dots$
3. $x_N \leq 1$

Though simplistic the above is in standard linear program format with objective function and constraints. It is a known result that Linear Programming is P-Complete which implies LP is not parallelizable unless $NC = P$. The Simplex Algorithm uses Gaussian Elimination with Partial Pivoting(GEPP) which is a P-Complete problem with no known NC algorithm. Above LP formulation is a special case of linear program with only free variables in the constraints. The RNC algorithm to fill the grid coordinates with 1(s) in parallel is an indirect way to set the free variables in the Linear Program for the grid to 1 in parallel and at random. Thus there is an RNC algorithm for a special case linear programming with free variables in constraints. But this may not imply an RNC algorithm for the P-Complete problem of GEPP used by Simplex algorithm.

4 Acknolwdgement

I dedicate this article to God.

5 Bibliography

References

- [1] RANDOM NUMBER GENERATORS ARE CHAOTIC -Charles Herring and Palmore,CACM, Vol. 38, Nu. 1, Technical Correspondence. (Appeared in ACM SIGPLAN Notices, 11, October 1989)
- [2] Efficient Parallel Pseudorandom number generation - Tygar and Reif, SIAM journal of computing, Vol 17 No.2,1988
- [3] Tygar-Reif variant with Lehmer's Chaotic Linear Congruence Generator replacing the bit computing function in each processor
- [4] Various resources on Chaos theory,non-linear dynamics and attractors
- [5] Various Lecture Notes and Resources on Simplex algorithm

- [6] Various Lecture Notes and Resources on P-Complete problems
- [7] Various Lecture Notes and Resources on Gaussian Elimination with Partial Pivoting or Gauss-Jordan algorithm

K.Srinivasan

Open Source Architect – Krishna iResearch – NeuronRain:

http://sourceforge.net/users/ka_shrinivaasan

<https://github.com/shrinivaasanka>

<http://neuronrain-documentation.readthedocs.io/en/latest/>

Research (Guided and Unguided) – <http://sites.google.com/site/kuja27>

About Myself

- Worked for various IT companies from 1999
- Doctoral research in CMI till 2011
- Presently working on a non-funded, not-for-profit open source initiative – repositories are in Sourceforge and GitHub – new machine learning powered linux kernel – NeuronRain - <http://neuronrain-documentation.readthedocs.io/en/latest/>

Academics/Awards/Recognition

- B.A(Hindi)-Praveen Uttarardh - Dakshin Bharat Hindi Prachar Sabha, Chennai - 1988-1992
- B.E(Computer Science and Engineering)-PSG College of Technology,Coimbatore – 1995-1999 – Proficiency award – Gold Medal recipient (2000)
- Master of Science(Theoretical Computer Science) – Chennai Mathematical Institute, Chennai – 2008-2010
- PhD(Theoretical Computer Science)-JRF-Chennai Mathematical Institute,Chennai – 2010-2011(took a break from PhD)
- Present – doing private independent academic research and also an open online free teaching: https://github.com/shrinivaasanka/Grafit/tree/master/course_material
- Teaching interests – algorithms, programming and open source, big data analysis, machine learning, cloud computing

Work/Research Experience

- BaaN Infosystems/SSA Global – 1999-2000 (Associate Software Engineer)
- iPlanet/Netscape/Sun Microsystems/Oracle – 2000-2005 (Member of Technical Staff)
- Krishna iResearch – Open Source Initiative of Self – Non-funded, Non-profit, Non-commercial – 2003 - (Architect and Developer) – NeuronRain Linux Kernel + Machine Learning - <http://neuronrain-documentation.readthedocs.io/en/latest/>
- Verizon – 2005 (System Analyst)
- webMethods/SoftwareAG – 2006-2008 (Specialist)
- Chennai Mathematical Institute,Chennai – 2010-2011 (PhD-Junior Research Fellow)
- Global Analytics/GAIN credit – 2011-2013 (Consultant and Architect)
- Clockwork Interviews/PiQube – 2013-2014 (Consultant)
- Cusdelight/CloudEnablers/Corestack – 2015 (Architect)

Publications/Preprints

- Decidability of Complement functions – 2011 -
<http://arxiv.org/abs/1106.4102>
- Algorithms for Intrinsic Merit – TAC version – 2010 -
http://www.nist.gov/tac/publications/2010/participant.papers/CMI_II_T.proceedings.pdf
- Algorithms for Intrinsic Merit – 2010 -
<http://arxiv.org/abs/1006.4458>
- Google Scholar -
<https://scholar.google.co.in/citations?user=eLZY7CIAAAAJ&hl=en>

Private Research (2012-)

- Major expansion of previous official PhD research till 2011
- Draft Publications are in - <http://sites.google.com/site/kuja27>
- Notable (unreviewed): Complexity of Majority and Non-majority boolean and non-boolean social choice(Margulis-Russo), Decidability of Complement diophantines, NC-PRAM Computational Geometric Factorization and Pell Diophantine, New Graph Theoretic Intrinsic Merit/Fitness measures, Recursive Lambda Function Growth based Graph Tensor Neuron Network Intrinsic Merit(Algorithmic Graph Theoretic models) for text graphs, Isomorphism of Integer partitions and tabulation/locality sensitive hashing(LSH), Majority voting and LSH etc.,

A Chaos theoretic Parallel Pseudorandom generator in RNC For Majority Voting and Pseudorandom Choice

Ka.Shrinivaasan (ka.shrinivaasan@gmail.com)

March 20, 2013

Abstract

In this article a Parallel Pseudorandom bit generator which exhibits Chaos theoretic behaviour and is in Randomized NC is presented.

1 Introduction

Theory of Chaotic systems and non-linear dynamics had created a widespread interest in the previous century and its applications were felt in numerous fields like weather forecast, economic phenomena like stock markets etc., As the Chaotic systems study non-linearity and inherent unpredictability or randomness in nature, they are the obvious choice for a pseudorandom source of stream of bits whose necessity often arises in Cryptographic functions, Randomized Algorithms in class BPP (Monte carlo and Las Vegas algorithms).

2 Logistic function and Lorenz Strange Attractors

In the middle of previous century, longterm weather forecasting received a jolt when Lorenz discovered a chaotic behaviour in weather predictive model equations due to a floating point round-off error which made the computer generated predictions (also called Strange attractors due to oscillation of the attractor between two centroids) to sharply deviate from expected values. Proverbially, this sensitive dependence on initial conditions is described as "flapping butterfly causes hurricane in atlantic". Verhulst's logistic function used in actuarial science $x_{n+1} = \lambda * x_n * (1 - x_n)$ is one such equation which exhibits chaotic behaviour. This recursive function initially oscillates with some periodicity and gradually chaos sets in to exhibit aperiodic, apparent randomness.

3 Parallel Pseudorandom bit generators in RNC with Chaotic behaviours

Lehmer's Prime Modulus Multiplicative Linear Congruence Pseudorandom Generators was one of the oldest Pseudorandom number generators and is defined as follows:

$$x_{n+1} = B * x_n \bmod M \quad \text{where } B > 0 \quad (1)$$

and M is a large prime and $x_0 > 0$ is the initial condition.

Palmore's Pseudorandom generator which was a later PRNG, is defined as,

$$x_{n+1} = B * x_n \mod 1 \quad (2)$$

where x_0 is the initial input fraction and B is the radix in which x is expressed.

Lehmer's and Palmore's Pseudorandom generators exhibit Chaotic behaviour due to sensitive dependence on initial input. Many Chaotic Pseudorandom number generators have been studied. [Tygar and Rief] gives an implementation of a parallel pseudorandom generator that is in Randomized NC. Each processor i in the parallel algorithm evaluates the function $B(k * s^i \mod N)$ and gets set of bits from all processors in parallel, where

$$B(x) = 0 \quad if \quad x < N/2 \quad and \quad 1 \quad if \quad x \geq N/2 \quad (3)$$

Input to function B can be replaced with Lehmer's or Palmore's pseudorandom generator in each processor of the Tygar-Rief algorithm giving a Chaotic Parallel Pseudorandom generator in Randomized NC. This PRG can be applied to the Randomized Circuits for Pseudorandom Choice and Majority Voting giving a RNC circuit (instead of a BPNC circuit).

4 Acknowledgement

I dedicate this article to God.

5 Bibliography

References

- [1] RANDOM NUMBER GENERATORS ARE CHAOTIC -Charles Herring and Palmore,CACM, Vol. 38, Nu. 1, Technical Correspondence. (Appeared in ACM SIGPLAN Notices, 11, October 1989)
- [2] Efficient Parallel Pseudorandom number generation - Tygar and Reif, SIAM journal of computing, Vol 17 No.2,1988
- [3] Various resources on Chaos theory,non-linear dynamics and attractors
- [4] Lower Bounds For Majority Voting and Pseudorandom Choice - <https://sites.google.com/site/kuja27/LowerBoundsForMajorityVotingPseudorandomChoice.pdf?attredirects=0>
- [5] Circuit For Computing Error Probability of Majority Voting - <https://sites.google.com/site/kuja27/CircuitForComputingErrorProbabilityOfMajorityVoting.pdf?attredirects=0>
- [6] Indepth analysis of a variant of Majority Voting with ZFC - <https://sites.google.com/site/kuja27/IndepthAnalysisOfVariantOfMajorityVotingwithZFC.pdf?attredirects=0>

Circuit For Computing Error Probability of Majority Voting - Updated Draft

*SrinivasanKannan(alias)Ka.Shrinivaasan(alias)ShrinivasKannan
IndependentOpenSourceDeveloper, ResearcherandConsultant*

Ph : 9789346927, 9003082186, 9791165980

*KrishnaiResearchOpenSourceProducts : http://sourceforge.net/users/ka_shrinivaasan,
https://www.ohloh.net/accounts/ka_shrinivaasan
ResearchWebsite : https://sites.google.com/site/kuja27/
(ka.shrinivaasan@gmail.com, shrinivas.kannan@gmail.com, kashrinivaasan@live.com)*

April 10, 2014

Abstract

In this article a DC uniform circuit for computation of error probability in majority voting is constructed and thus shown to be in PH. This error probability in majority voting is nothing but the RHS of the P(good) equation published earlier as mentioned in bibliography.

1 Derivation of P(good) expression

In a distributed systems (e.g cloud computing environments) with $2n$ nodes, the leader is elected by a majority voting if it obtains $n + 1$ or more votes. We would like to bound the probability that such an outcome of a majority vote is good. This is precisely the probability that atleast $n + 1$ nodes have made a good decision. Assumption here is that a good majority decision by majority of the nodes with high probability also results in a good outcome in the majority voting (rather it is trivial and is accepted axiomatically without proof). Thus by probability union bound,

```
Pr(leader elected by majority voting is good) =  
Pr(atleast n+1 nodes have made a good decision)  
  
= Pr (n+1 nodes have made good decision) +  
Pr(n+2 nodes have made good decision) +  
... +  
Pr(2n nodes have made good decision)  
- (0 for intersection probability)
```

If $\Pr(\text{good decision by a node}) = 0.5$
(assuming a uniform distribution and thus good and bad decisions are equally probable) then LHS of the P(Good) equation is 0.5 and the RHS is obtained through the series distribution as,

$$= 2nC_{(n+1)} * (0.5)^{(n+1)} * (0.5)^{(n-1)} + \dots + 2nC_{2n} (0.5)^{(2n)} \quad (1)$$

$$= (0.25)^n [2nC_{(n+1)} + 2nC_{(n+2)} + \dots + 2nC_{(2n)}] \quad (2)$$

$$= [(2n)!/(4^n)] * [1/(n+1)!(n-1)! + 1/(n-2)!(n+2)! + \dots + 1/(2n)!] \quad (3)$$

This series tends to 0.5 which can be confirmed by simple convergence test. The fraction $[nthterm - (n-1)thterm]/n$ tends to zero as n tends to infinity. Closed formula for summation of binomial coefficients in the summation above requires Hypergeometric functions as follows:

$$\sum_{i=0}^m mC_i = 2^m \quad (4)$$

and

$$\sum_{i=k}^m mC_i = 2^m - \sum_{i=0}^{k-1} mC_i \quad (5)$$

is the quantity within the parenthesis in above summation and

$$\sum_{i=0}^{k-1} mC_i \quad (6)$$

has a closed formula that can be derived using Hypergeometric functions (Gosper Algorithm, Zeilberger algorithm, Wilf-Zeilberger pairs etc.,) which is a different field by itself and is non-trivial and thus LHS can be computed and substituted in the above parenthesis with binomial coefficients.

Output of a computer program which computes the above is in appendix (<http://sourceforge.net/p/asfer/code/HEAD/tree/cpp-src/miscellaneous/pgood.cpp>). For odd number of voters, probability can be analogously derived by rounding off to nearest integer to get past the halfway point. For odd number of the voter population, say m , the halfway point is $ceiling(m/2)$. Let $x = ceiling(m/2)$. Thus above series becomes,

$$= mC_{(x+1)} * (0.5)^m + mC_{(x+2)} * (0.5)^m + \dots + mC_m * (0.5)^m \quad (7)$$

$$= (0.5)^m * [mC_{(x)} + mC_{(x+1)} + \dots + mC_m] \quad (8)$$

$$= m!/(2^m) [1/x!(m-x)! + \dots + 1/m!] \quad (9)$$

Assumption here is that a good majority decision with high probability also results in a good outcome in the majority voting (rather it is trivial because a good decision implies good judgement of the majority choice by the voters which is the outcome and is stated without proof).

2 A simple majority voting example with 5 voters

Following are the 32 possible voting patterns in terms of nature of individual decision for 5 voters to elect a leader (0 means voter has made bad decision and 1 means voter has made good decision):

00000, 10000
00001, 10001
00010, 10010
00011, 10011
00100, 10100

```

00101, 10101
00110, 10110
00111, 10111
01000, 11000
01001, 11001
01010, 11010
01011, 11011
01100, 11100
01101, 11101
01110, 11110
01111, 11111

```

From the above the probability that atleast $ceiling(5/2)$ (or more than or equal to 3 in above example) voters have made a good decision, can be computed easily by glancing. Out of 32 patterns, 16 have 3 or more 1s (good decisions). Thus the probability that elected leader is good is 16/32. This can be derived from above series also as,

$$\begin{aligned}
&= 1/32 [5C3 + 5C4 + 5C5] \\
&= 16/32 = 0.5
\end{aligned}$$

3 Circuit for computing P(Good) error probability from voter decision patterns

Complexity of computing above series in RHS of P(Good) equation is exponential in n because of computation of factorials (can be approximated by Stirling's formula). P(Good) series implies that any leader election algorithm that involves majority voting (under zero bias space where $Pr[decision = 1] - Pr[decision = 0] = 0$) is no better than a (pseudo)random choice. Translating P(good) series into circuit requires computation of majority function on 2^{2n} possible inputs corresponding to all possible voting patters by the $2n$ voters.

Following is the algorithm for drawing the above circuit:

1. Have 2^{2n} majority circuits. Each of these majority circuit takes as input one voting decision bit pattern each with each bit as a decision(good or bad) input for corresponding voter(as explained in example above). Majority function can be computed in polynomial size by sorting networks (Ajtai et al) or through non-uniform NC1 circuit (Barrington) or Valiant's non-constructive majority circuit of size $n^{5.3}$. Thus each majority circuit computes the majority of the voting pattern and outputs 1 or 0 depending on which of the bits are in majority (1 means majority decision is good and 0 means bad for that voting pattern). Thus 2^{2n} majority circuits compute the majority of each of the 2^{2n} voting bit patterns and output 0 or 1.
2. An addition circuit then adds up the 2^{2n} bits output by all of the above majority circuits. This addition circuit has exponential fan-in and thus exponential in n and thus NC1 cannot compute this addition which requires bounded fanin ,logdepth and polysize circuit. Exponential sized circuits are in DC-uniform family characterized by Polynomial Hierarchy(PH) which is defined as circuit having AND,OR and NOT gates and size 2^{2n} with unbounded fanin.
3. Output of the above addition circuit is the numerator of the P(Good) fraction. Thus a division circuit is needed to divide this numerator by denominator which is 2^{2n} . Division can be performed in TC0. Thus summing up we have a 3 step circuit ($NC1 + DC - uniform + TC0$).

Subsuming NC1 and TC0 in DC-uniform gives a DC-uniform exponential sized circuit to compute the $P(\text{Good})$ RHS probability.

4 Conclusion

Thus a series expression for the error probability in majority voting has been derived and a DC-uniform circuit has been constructed for it.

5 Appendix for $P(\text{Good})$ computation with even number voter population output by a computer program (probability is in percentage)

```
Probability of good choice for population of 0=0
prob - prevprob = 0
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2=25
prob - prevprob = 0.25
Convergence test: (sum - prevsum)/prevsum = inf
Probability of good choice for population of 4=31.25
prob - prevprob = 0.0625
Convergence test: (sum - prevsum)/prevsum = 0.25
Probability of good choice for population of 6=34.375
prob - prevprob = 0.03125
Convergence test: (sum - prevsum)/prevsum = 0.1
Probability of good choice for population of 8=36.3281
prob - prevprob = 0.0195312
Convergence test: (sum - prevsum)/prevsum = 0.0568182
Probability of good choice for population of 10=37.6953
prob - prevprob = 0.0136719
Convergence test: (sum - prevsum)/prevsum = 0.0376344
Probability of good choice for population of 12=38.7207
prob - prevprob = 0.0102539
Convergence test: (sum - prevsum)/prevsum = 0.0272021
Probability of good choice for population of 14=39.5264
prob - prevprob = 0.00805664
Convergence test: (sum - prevsum)/prevsum = 0.0208071
Probability of good choice for population of 16=40.181
prob - prevprob = 0.00654602
Convergence test: (sum - prevsum)/prevsum = 0.0165611
Probability of good choice for population of 18=40.7265
prob - prevprob = 0.00545502
Convergence test: (sum - prevsum)/prevsum = 0.0135761
Probability of good choice for population of 20=41.1901
prob - prevprob = 0.00463676
Convergence test: (sum - prevsum)/prevsum = 0.0113851
Probability of good choice for population of 22=41.5906
```

```
prob - prevprob = 0.00400448
Convergence test: (sum - prevsum)/prevsum = 0.00972193
Probability of good choice for population of 24=41.941
prob - prevprob = 0.00350392
Convergence test: (sum - prevsum)/prevsum = 0.00842479
Probability of good choice for population of 26=42.2509
prob - prevprob = 0.00309962
Convergence test: (sum - prevsum)/prevsum = 0.00739043
Probability of good choice for population of 28=42.5277
prob - prevprob = 0.00276752
Convergence test: (sum - prevsum)/prevsum = 0.00655019
Probability of good choice for population of 30=42.7768
prob - prevprob = 0.00249077
Convergence test: (sum - prevsum)/prevsum = 0.00585681
Probability of good choice for population of 32=43.0025
prob - prevprob = 0.00225726
Convergence test: (sum - prevsum)/prevsum = 0.00527683
Probability of good choice for population of 34=43.2083
prob - prevprob = 0.00205809
Convergence test: (sum - prevsum)/prevsum = 0.00478597
Probability of good choice for population of 36=43.397
prob - prevprob = 0.00188658
Convergence test: (sum - prevsum)/prevsum = 0.00436624
Probability of good choice for population of 38=43.5707
prob - prevprob = 0.00173764
Convergence test: (sum - prevsum)/prevsum = 0.00400406
Probability of good choice for population of 40=43.7315
prob - prevprob = 0.00160732
Convergence test: (sum - prevsum)/prevsum = 0.00368898
Probability of good choice for population of 42=43.8807
prob - prevprob = 0.00149251
Convergence test: (sum - prevsum)/prevsum = 0.00341289
Probability of good choice for population of 44=44.0198
prob - prevprob = 0.00139075
Convergence test: (sum - prevsum)/prevsum = 0.00316938
Probability of good choice for population of 46=44.1498
prob - prevprob = 0.00130005
Convergence test: (sum - prevsum)/prevsum = 0.00295332
Probability of good choice for population of 48=44.2717
prob - prevprob = 0.00121879
Convergence test: (sum - prevsum)/prevsum = 0.00276058
Probability of good choice for population of 50=44.3862
prob - prevprob = 0.00114567
Convergence test: (sum - prevsum)/prevsum = 0.00258781
Probability of good choice for population of 52=44.4942
prob - prevprob = 0.00107957
Convergence test: (sum - prevsum)/prevsum = 0.00243222
Probability of good choice for population of 54=44.5962
```

```
prob - prevprob = 0.00101959
Convergence test: (sum - prevsum)/prevsum = 0.00229152
Probability of good choice for population of 56=44.6927
prob - prevprob = 0.000964972
Convergence test: (sum - prevsum)/prevsum = 0.0021638
Probability of good choice for population of 58=44.7842
prob - prevprob = 0.00091506
Convergence test: (sum - prevsum)/prevsum = 0.00204745
Probability of good choice for population of 60=44.8711
prob - prevprob = 0.000869307
Convergence test: (sum - prevsum)/prevsum = 0.0019411
Probability of good choice for population of 62=44.9538
prob - prevprob = 0.000827243
Convergence test: (sum - prevsum)/prevsum = 0.0018436
Probability of good choice for population of 64=45.0327
prob - prevprob = 0.000788466
Convergence test: (sum - prevsum)/prevsum = 0.00175395
Probability of good choice for population of 66=45.1079
prob - prevprob = 0.000752627
Convergence test: (sum - prevsum)/prevsum = 0.00167129
Probability of good choice for population of 68=45.1799
prob - prevprob = 0.000719423
Convergence test: (sum - prevsum)/prevsum = 0.00159489
Probability of good choice for population of 70=45.2487
prob - prevprob = 0.00068859
Convergence test: (sum - prevsum)/prevsum = 0.00152411
Probability of good choice for population of 72=45.3147
prob - prevprob = 0.000659899
Convergence test: (sum - prevsum)/prevsum = 0.00145838
Probability of good choice for population of 74=45.378
prob - prevprob = 0.000633146
Convergence test: (sum - prevsum)/prevsum = 0.00139722
Probability of good choice for population of 76=45.4388
prob - prevprob = 0.000608154
Convergence test: (sum - prevsum)/prevsum = 0.00134019
Probability of good choice for population of 78=45.4973
prob - prevprob = 0.000584763
Convergence test: (sum - prevsum)/prevsum = 0.00128692
Probability of good choice for population of 80=45.5536
prob - prevprob = 0.000562835
Convergence test: (sum - prevsum)/prevsum = 0.00123707
Probability of good choice for population of 82=45.6078
prob - prevprob = 0.000542243
Convergence test: (sum - prevsum)/prevsum = 0.00119034
Probability of good choice for population of 84=45.6601
prob - prevprob = 0.000522877
Convergence test: (sum - prevsum)/prevsum = 0.00114646
Probability of good choice for population of 86=45.7106
```

prob - prevprob = 0.000504637
Convergence test: (sum - prevsum)/prevsum = 0.0011052
Probability of good choice for population of 88=45.7593
prob - prevprob = 0.000487434
Convergence test: (sum - prevsum)/prevsum = 0.00106635
Probability of good choice for population of 90=45.8064
prob - prevprob = 0.000471186
Convergence test: (sum - prevsum)/prevsum = 0.00102971
Probability of good choice for population of 92=45.852
prob - prevprob = 0.000455821
Convergence test: (sum - prevsum)/prevsum = 0.000995103
Probability of good choice for population of 94=45.8962
prob - prevprob = 0.000441274
Convergence test: (sum - prevsum)/prevsum = 0.000962387
Probability of good choice for population of 96=45.9389
prob - prevprob = 0.000427484
Convergence test: (sum - prevsum)/prevsum = 0.000931416
Probability of good choice for population of 98=45.9803
prob - prevprob = 0.000414398
Convergence test: (sum - prevsum)/prevsum = 0.000902063
Probability of good choice for population of 100=46.0205
prob - prevprob = 0.000401966
Convergence test: (sum - prevsum)/prevsum = 0.000874212
Probability of good choice for population of 102=46.0596
prob - prevprob = 0.000390143
Convergence test: (sum - prevsum)/prevsum = 0.000847759
Probability of good choice for population of 104=46.0974
prob - prevprob = 0.000378889
Convergence test: (sum - prevsum)/prevsum = 0.000822607
Probability of good choice for population of 106=46.1343
prob - prevprob = 0.000368166
Convergence test: (sum - prevsum)/prevsum = 0.000798669
Probability of good choice for population of 108=46.1701
prob - prevprob = 0.000357939
Convergence test: (sum - prevsum)/prevsum = 0.000775864
Probability of good choice for population of 110=46.2049
prob - prevprob = 0.000348177
Convergence test: (sum - prevsum)/prevsum = 0.000754119
Probability of good choice for population of 112=46.2388
prob - prevprob = 0.000338851
Convergence test: (sum - prevsum)/prevsum = 0.000733366
Probability of good choice for population of 114=46.2717
prob - prevprob = 0.000329934
Convergence test: (sum - prevsum)/prevsum = 0.000713544
Probability of good choice for population of 116=46.3039
prob - prevprob = 0.000321401
Convergence test: (sum - prevsum)/prevsum = 0.000694595
Probability of good choice for population of 118=46.3352

```
prob - prevprob = 0.00031323
Convergence test: (sum - prevsum)/prevsum = 0.000676465
Probability of good choice for population of 120=46.3658
prob - prevprob = 0.000305399
Convergence test: (sum - prevsum)/prevsum = 0.000659108
Probability of good choice for population of 122=46.3955
prob - prevprob = 0.000297889
Convergence test: (sum - prevsum)/prevsum = 0.000642477
Probability of good choice for population of 124=46.4246
prob - prevprob = 0.000290682
Convergence test: (sum - prevsum)/prevsum = 0.000626531
Probability of good choice for population of 126=46.453
prob - prevprob = 0.000283761
Convergence test: (sum - prevsum)/prevsum = 0.00061123
Probability of good choice for population of 128=46.4807
prob - prevprob = 0.000277111
Convergence test: (sum - prevsum)/prevsum = 0.00059654
Probability of good choice for population of 130=46.5078
prob - prevprob = 0.000270716
Convergence test: (sum - prevsum)/prevsum = 0.000582426
Probability of good choice for population of 132=46.5342
prob - prevprob = 0.000264563
Convergence test: (sum - prevsum)/prevsum = 0.000568858
Probability of good choice for population of 134=46.5601
prob - prevprob = 0.00025864
Convergence test: (sum - prevsum)/prevsum = 0.000555806
Probability of good choice for population of 136=46.5854
prob - prevprob = 0.000252935
Convergence test: (sum - prevsum)/prevsum = 0.000543244
Probability of good choice for population of 138=46.6101
prob - prevprob = 0.000247436
Convergence test: (sum - prevsum)/prevsum = 0.000531146
Probability of good choice for population of 140=46.6343
prob - prevprob = 0.000242134
Convergence test: (sum - prevsum)/prevsum = 0.000519488
Probability of good choice for population of 142=46.658
prob - prevprob = 0.000237018
Convergence test: (sum - prevsum)/prevsum = 0.000508249
Probability of good choice for population of 144=46.6812
prob - prevprob = 0.000232081
Convergence test: (sum - prevsum)/prevsum = 0.000497407
Probability of good choice for population of 146=46.704
prob - prevprob = 0.000227312
Convergence test: (sum - prevsum)/prevsum = 0.000486945
Probability of good choice for population of 148=46.7262
prob - prevprob = 0.000222704
Convergence test: (sum - prevsum)/prevsum = 0.000476842
Probability of good choice for population of 150=46.7481
```

```
prob - prevprob = 0.00021825
Convergence test: (sum - prevsum)/prevsum = 0.000467082
Probability of good choice for population of 152=46.7695
prob - prevprob = 0.000213942
Convergence test: (sum - prevsum)/prevsum = 0.00045765
Probability of good choice for population of 154=46.7904
prob - prevprob = 0.000209775
Convergence test: (sum - prevsum)/prevsum = 0.000448529
Probability of good choice for population of 156=46.811
prob - prevprob = 0.000205741
Convergence test: (sum - prevsum)/prevsum = 0.000439707
Probability of good choice for population of 158=46.8312
prob - prevprob = 0.000201834
Convergence test: (sum - prevsum)/prevsum = 0.000431168
Probability of good choice for population of 160=46.851
prob - prevprob = 0.00019805
Convergence test: (sum - prevsum)/prevsum = 0.000422901
Probability of good choice for population of 162=46.8704
prob - prevprob = 0.000194382
Convergence test: (sum - prevsum)/prevsum = 0.000414894
Probability of good choice for population of 164=46.8895
prob - prevprob = 0.000190826
Convergence test: (sum - prevsum)/prevsum = 0.000407136
Probability of good choice for population of 166=46.9083
prob - prevprob = 0.000187378
Convergence test: (sum - prevsum)/prevsum = 0.000399615
Probability of good choice for population of 168=46.9267
prob - prevprob = 0.000184032
Convergence test: (sum - prevsum)/prevsum = 0.000392323
Probability of good choice for population of 170=46.9447
prob - prevprob = 0.000180784
Convergence test: (sum - prevsum)/prevsum = 0.000385248
Probability of good choice for population of 172=46.9625
prob - prevprob = 0.000177631
Convergence test: (sum - prevsum)/prevsum = 0.000378383
Probability of good choice for population of 174=46.98
prob - prevprob = 0.000174568
Convergence test: (sum - prevsum)/prevsum = 0.000371718
Probability of good choice for population of 176=46.9971
prob - prevprob = 0.000171593
Convergence test: (sum - prevsum)/prevsum = 0.000365247
Probability of good choice for population of 178=47.014
prob - prevprob = 0.000168701
Convergence test: (sum - prevsum)/prevsum = 0.00035896
Probability of good choice for population of 180=47.0306
prob - prevprob = 0.000165889
Convergence test: (sum - prevsum)/prevsum = 0.00035285
Probability of good choice for population of 182=47.0469
```

```
prob - prevprob = 0.000163155
Convergence test: (sum - prevsum)/prevsum = 0.000346912
Probability of good choice for population of 184=47.063
prob - prevprob = 0.000160494
Convergence test: (sum - prevsum)/prevsum = 0.000341137
Probability of good choice for population of 186=47.0787
prob - prevprob = 0.000157906
Convergence test: (sum - prevsum)/prevsum = 0.00033552
Probability of good choice for population of 188=47.0943
prob - prevprob = 0.000155386
Convergence test: (sum - prevsum)/prevsum = 0.000330056
Probability of good choice for population of 190=47.1096
prob - prevprob = 0.000152933
Convergence test: (sum - prevsum)/prevsum = 0.000324737
Probability of good choice for population of 192=47.1246
prob - prevprob = 0.000150543
Convergence test: (sum - prevsum)/prevsum = 0.000319559
Probability of good choice for population of 194=47.1394
prob - prevprob = 0.000148215
Convergence test: (sum - prevsum)/prevsum = 0.000314517
Probability of good choice for population of 196=47.154
prob - prevprob = 0.000145946
Convergence test: (sum - prevsum)/prevsum = 0.000309606
Probability of good choice for population of 198=47.1684
prob - prevprob = 0.000143735
Convergence test: (sum - prevsum)/prevsum = 0.00030482
Probability of good choice for population of 200=47.1826
prob - prevprob = 0.000141579
Convergence test: (sum - prevsum)/prevsum = 0.000300157
Probability of good choice for population of 202=47.1965
prob - prevprob = 0.000139476
Convergence test: (sum - prevsum)/prevsum = 0.00029561
Probability of good choice for population of 204=47.2103
prob - prevprob = 0.000137425
Convergence test: (sum - prevsum)/prevsum = 0.000291177
Probability of good choice for population of 206=47.2238
prob - prevprob = 0.000135424
Convergence test: (sum - prevsum)/prevsum = 0.000286853
Probability of good choice for population of 208=47.2372
prob - prevprob = 0.000133471
Convergence test: (sum - prevsum)/prevsum = 0.000282634
Probability of good choice for population of 210=47.2503
prob - prevprob = 0.000131564
Convergence test: (sum - prevsum)/prevsum = 0.000278518
Probability of good choice for population of 212=47.2633
prob - prevprob = 0.000129702
Convergence test: (sum - prevsum)/prevsum = 0.0002745
Probability of good choice for population of 214=47.2761
```

```
prob - prevprob = 0.000127884
Convergence test: (sum - prevsum)/prevsum = 0.000270578
Probability of good choice for population of 216=47.2887
prob - prevprob = 0.000126108
Convergence test: (sum - prevsum)/prevsum = 0.000266748
Probability of good choice for population of 218=47.3011
prob - prevprob = 0.000124372
Convergence test: (sum - prevsum)/prevsum = 0.000263007
Probability of good choice for population of 220=47.3134
prob - prevprob = 0.000122676
Convergence test: (sum - prevsum)/prevsum = 0.000259352
Probability of good choice for population of 222=47.3255
prob - prevprob = 0.000121019
Convergence test: (sum - prevsum)/prevsum = 0.000255781
Probability of good choice for population of 224=47.3374
prob - prevprob = 0.000119398
Convergence test: (sum - prevsum)/prevsum = 0.000252291
Probability of good choice for population of 226=47.3492
prob - prevprob = 0.000117813
Convergence test: (sum - prevsum)/prevsum = 0.000248879
Probability of good choice for population of 228=47.3608
prob - prevprob = 0.000116263
Convergence test: (sum - prevsum)/prevsum = 0.000245543
Probability of good choice for population of 230=47.3723
prob - prevprob = 0.000114746
Convergence test: (sum - prevsum)/prevsum = 0.000242281
Probability of good choice for population of 232=47.3836
prob - prevprob = 0.000113262
Convergence test: (sum - prevsum)/prevsum = 0.00023909
Probability of good choice for population of 234=47.3948
prob - prevprob = 0.00011181
Convergence test: (sum - prevsum)/prevsum = 0.000235968
Probability of good choice for population of 236=47.4059
prob - prevprob = 0.000110389
Convergence test: (sum - prevsum)/prevsum = 0.000232914
Probability of good choice for population of 238=47.4168
prob - prevprob = 0.000108998
Convergence test: (sum - prevsum)/prevsum = 0.000229924
Probability of good choice for population of 240=47.4275
prob - prevprob = 0.000107635
Convergence test: (sum - prevsum)/prevsum = 0.000226998
Probability of good choice for population of 242=47.4381
prob - prevprob = 0.000106301
Convergence test: (sum - prevsum)/prevsum = 0.000224133
Probability of good choice for population of 244=47.4486
prob - prevprob = 0.000104994
Convergence test: (sum - prevsum)/prevsum = 0.000221328
Probability of good choice for population of 246=47.459
```

```
prob - prevprob = 0.000103713
Convergence test: (sum - prevsum)/prevsum = 0.00021858
Probability of good choice for population of 248=47.4693
prob - prevprob = 0.000102459
Convergence test: (sum - prevsum)/prevsum = 0.000215889
Probability of good choice for population of 250=47.4794
prob - prevprob = 0.000101229
Convergence test: (sum - prevsum)/prevsum = 0.000213252
Probability of good choice for population of 252=47.4894
prob - prevprob = 0.000100024
Convergence test: (sum - prevsum)/prevsum = 0.000210669
Probability of good choice for population of 254=47.4993
prob - prevprob = 9.88428e-05
Convergence test: (sum - prevsum)/prevsum = 0.000208137
Probability of good choice for population of 256=47.509
prob - prevprob = 9.76845e-05
Convergence test: (sum - prevsum)/prevsum = 0.000205655
Probability of good choice for population of 258=47.5187
prob - prevprob = 9.65487e-05
Convergence test: (sum - prevsum)/prevsum = 0.000203222
Probability of good choice for population of 260=47.5282
prob - prevprob = 9.54346e-05
Convergence test: (sum - prevsum)/prevsum = 0.000200836
Probability of good choice for population of 262=47.5377
prob - prevprob = 9.43419e-05
Convergence test: (sum - prevsum)/prevsum = 0.000198496
Probability of good choice for population of 264=47.547
prob - prevprob = 9.32698e-05
Convergence test: (sum - prevsum)/prevsum = 0.000196202
Probability of good choice for population of 266=47.5562
prob - prevprob = 9.22179e-05
Convergence test: (sum - prevsum)/prevsum = 0.000193951
Probability of good choice for population of 268=47.5653
prob - prevprob = 9.11856e-05
Convergence test: (sum - prevsum)/prevsum = 0.000191743
Probability of good choice for population of 270=47.5744
prob - prevprob = 9.01724e-05
Convergence test: (sum - prevsum)/prevsum = 0.000189576
Probability of good choice for population of 272=47.5833
prob - prevprob = 8.91779e-05
Convergence test: (sum - prevsum)/prevsum = 0.000187449
Probability of good choice for population of 274=47.5921
prob - prevprob = 8.82015e-05
Convergence test: (sum - prevsum)/prevsum = 0.000185362
Probability of good choice for population of 276=47.6008
prob - prevprob = 8.72428e-05
Convergence test: (sum - prevsum)/prevsum = 0.000183314
Probability of good choice for population of 278=47.6095
```

prob - prevprob = 8.63013e-05
Convergence test: (sum - prevsum)/prevsum = 0.000181302
Probability of good choice for population of 280=47.618
prob - prevprob = 8.53766e-05
Convergence test: (sum - prevsum)/prevsum = 0.000179327
Probability of good choice for population of 282=47.6264
prob - prevprob = 8.44684e-05
Convergence test: (sum - prevsum)/prevsum = 0.000177388
Probability of good choice for population of 284=47.6348
prob - prevprob = 8.35761e-05
Convergence test: (sum - prevsum)/prevsum = 0.000175483
Probability of good choice for population of 286=47.6431
prob - prevprob = 8.26994e-05
Convergence test: (sum - prevsum)/prevsum = 0.000173611
Probability of good choice for population of 288=47.6512
prob - prevprob = 8.1838e-05
Convergence test: (sum - prevsum)/prevsum = 0.000171773
Probability of good choice for population of 290=47.6593
prob - prevprob = 8.09914e-05
Convergence test: (sum - prevsum)/prevsum = 0.000169967
Probability of good choice for population of 292=47.6674
prob - prevprob = 8.01593e-05
Convergence test: (sum - prevsum)/prevsum = 0.000168192
Probability of good choice for population of 294=47.6753
prob - prevprob = 7.93413e-05
Convergence test: (sum - prevsum)/prevsum = 0.000166448
Probability of good choice for population of 296=47.6832
prob - prevprob = 7.85372e-05
Convergence test: (sum - prevsum)/prevsum = 0.000164734
Probability of good choice for population of 298=47.6909
prob - prevprob = 7.77466e-05
Convergence test: (sum - prevsum)/prevsum = 0.000163048
Probability of good choice for population of 300=47.6986
prob - prevprob = 7.69691e-05
Convergence test: (sum - prevsum)/prevsum = 0.000161391
Probability of good choice for population of 302=47.7062
prob - prevprob = 7.62045e-05
Convergence test: (sum - prevsum)/prevsum = 0.000159762
Probability of good choice for population of 304=47.7138
prob - prevprob = 7.54525e-05
Convergence test: (sum - prevsum)/prevsum = 0.000158161
Probability of good choice for population of 306=47.7213
prob - prevprob = 7.47127e-05
Convergence test: (sum - prevsum)/prevsum = 0.000156585
Probability of good choice for population of 308=47.7287
prob - prevprob = 7.3985e-05
Convergence test: (sum - prevsum)/prevsum = 0.000155036
Probability of good choice for population of 310=47.736

prob - prevprob = 7.3269e-05
Convergence test: (sum - prevsum)/prevsum = 0.000153512
Probability of good choice for population of 312=47.7432
prob - prevprob = 7.25645e-05
Convergence test: (sum - prevsum)/prevsum = 0.000152012
Probability of good choice for population of 314=47.7504
prob - prevprob = 7.18712e-05
Convergence test: (sum - prevsum)/prevsum = 0.000150537
Probability of good choice for population of 316=47.7575
prob - prevprob = 7.11889e-05
Convergence test: (sum - prevsum)/prevsum = 0.000149085
Probability of good choice for population of 318=47.7646
prob - prevprob = 7.05173e-05
Convergence test: (sum - prevsum)/prevsum = 0.000147657
Probability of good choice for population of 320=47.7716
prob - prevprob = 6.98562e-05
Convergence test: (sum - prevsum)/prevsum = 0.000146251
Probability of good choice for population of 322=47.7785
prob - prevprob = 6.92054e-05
Convergence test: (sum - prevsum)/prevsum = 0.000144867
Probability of good choice for population of 324=47.7854
prob - prevprob = 6.85646e-05
Convergence test: (sum - prevsum)/prevsum = 0.000143505
Probability of good choice for population of 326=47.7922
prob - prevprob = 6.79336e-05
Convergence test: (sum - prevsum)/prevsum = 0.000142164
Probability of good choice for population of 328=47.7989
prob - prevprob = 6.73123e-05
Convergence test: (sum - prevsum)/prevsum = 0.000140844
Probability of good choice for population of 330=47.8056
prob - prevprob = 6.67004e-05
Convergence test: (sum - prevsum)/prevsum = 0.000139544
Probability of good choice for population of 332=47.8122
prob - prevprob = 6.60976e-05
Convergence test: (sum - prevsum)/prevsum = 0.000138264
Probability of good choice for population of 334=47.8187
prob - prevprob = 6.5504e-05
Convergence test: (sum - prevsum)/prevsum = 0.000137003
Probability of good choice for population of 336=47.8252
prob - prevprob = 6.49191e-05
Convergence test: (sum - prevsum)/prevsum = 0.000135761
Probability of good choice for population of 338=47.8316
prob - prevprob = 6.43429e-05
Convergence test: (sum - prevsum)/prevsum = 0.000134538
Probability of good choice for population of 340=47.838
prob - prevprob = 6.37752e-05
Convergence test: (sum - prevsum)/prevsum = 0.000133333
Probability of good choice for population of 342=47.8443

```
prob - prevprob = 6.32157e-05
Convergence test: (sum - prevsum)/prevsum = 0.000132145
Probability of good choice for population of 344=47.8506
prob - prevprob = 6.26644e-05
Convergence test: (sum - prevsum)/prevsum = 0.000130976
Probability of good choice for population of 346=47.8568
prob - prevprob = 6.21211e-05
Convergence test: (sum - prevsum)/prevsum = 0.000129823
Probability of good choice for population of 348=47.863
prob - prevprob = 6.15856e-05
Convergence test: (sum - prevsum)/prevsum = 0.000128687
Probability of good choice for population of 350=47.8691
prob - prevprob = 6.10577e-05
Convergence test: (sum - prevsum)/prevsum = 0.000127568
Probability of good choice for population of 352=47.8751
prob - prevprob = 6.05373e-05
Convergence test: (sum - prevsum)/prevsum = 0.000126464
Probability of good choice for population of 354=47.8811
prob - prevprob = 6.00243e-05
Convergence test: (sum - prevsum)/prevsum = 0.000125377
Probability of good choice for population of 356=47.8871
prob - prevprob = 5.95185e-05
Convergence test: (sum - prevsum)/prevsum = 0.000124305
Probability of good choice for population of 358=47.893
prob - prevprob = 5.90197e-05
Convergence test: (sum - prevsum)/prevsum = 0.000123248
Probability of good choice for population of 360=47.8988
prob - prevprob = 5.85279e-05
Convergence test: (sum - prevsum)/prevsum = 0.000122206
Probability of good choice for population of 362=47.9047
prob - prevprob = 5.80428e-05
Convergence test: (sum - prevsum)/prevsum = 0.000121178
Probability of good choice for population of 364=47.9104
prob - prevprob = 5.75645e-05
Convergence test: (sum - prevsum)/prevsum = 0.000120165
Probability of good choice for population of 366=47.9161
prob - prevprob = 5.70926e-05
Convergence test: (sum - prevsum)/prevsum = 0.000119165
Probability of good choice for population of 368=47.9218
prob - prevprob = 5.66272e-05
Convergence test: (sum - prevsum)/prevsum = 0.00011818
Probability of good choice for population of 370=47.9274
prob - prevprob = 5.61681e-05
Convergence test: (sum - prevsum)/prevsum = 0.000117208
Probability of good choice for population of 372=47.933
prob - prevprob = 5.57151e-05
Convergence test: (sum - prevsum)/prevsum = 0.000116249
Probability of good choice for population of 374=47.9385
```

```
prob - prevprob = 5.52682e-05
Convergence test: (sum - prevsum)/prevsum = 0.000115303
Probability of good choice for population of 376=47.944
prob - prevprob = 5.48272e-05
Convergence test: (sum - prevsum)/prevsum = 0.00011437
Probability of good choice for population of 378=47.9494
prob - prevprob = 5.43921e-05
Convergence test: (sum - prevsum)/prevsum = 0.000113449
Probability of good choice for population of 380=47.9548
prob - prevprob = 5.39627e-05
Convergence test: (sum - prevsum)/prevsum = 0.000112541
Probability of good choice for population of 382=47.9602
prob - prevprob = 5.35389e-05
Convergence test: (sum - prevsum)/prevsum = 0.000111644
Probability of good choice for population of 384=47.9655
prob - prevprob = 5.31206e-05
Convergence test: (sum - prevsum)/prevsum = 0.00011076
Probability of good choice for population of 386=47.9708
prob - prevprob = 5.27077e-05
Convergence test: (sum - prevsum)/prevsum = 0.000109887
Probability of good choice for population of 388=47.976
prob - prevprob = 5.23002e-05
Convergence test: (sum - prevsum)/prevsum = 0.000109025
Probability of good choice for population of 390=47.9812
prob - prevprob = 5.18979e-05
Convergence test: (sum - prevsum)/prevsum = 0.000108175
Probability of good choice for population of 392=47.9863
prob - prevprob = 5.15007e-05
Convergence test: (sum - prevsum)/prevsum = 0.000107335
Probability of good choice for population of 394=47.9914
prob - prevprob = 5.11086e-05
Convergence test: (sum - prevsum)/prevsum = 0.000106507
Probability of good choice for population of 396=47.9965
prob - prevprob = 5.07214e-05
Convergence test: (sum - prevsum)/prevsum = 0.000105688
Probability of good choice for population of 398=48.0015
prob - prevprob = 5.03391e-05
Convergence test: (sum - prevsum)/prevsum = 0.000104881
Probability of good choice for population of 400=48.0065
prob - prevprob = 4.99615e-05
Convergence test: (sum - prevsum)/prevsum = 0.000104083
Probability of good choice for population of 402=48.0115
prob - prevprob = 4.95887e-05
Convergence test: (sum - prevsum)/prevsum = 0.000103296
Probability of good choice for population of 404=48.0164
prob - prevprob = 4.92205e-05
Convergence test: (sum - prevsum)/prevsum = 0.000102518
Probability of good choice for population of 406=48.0213
```

prob - prevprob = 4.88568e-05
Convergence test: (sum - prevsum)/prevsum = 0.00010175
Probability of good choice for population of 408=48.0262
prob - prevprob = 4.84975e-05
Convergence test: (sum - prevsum)/prevsum = 0.000100992
Probability of good choice for population of 410=48.031
prob - prevprob = 4.81427e-05
Convergence test: (sum - prevsum)/prevsum = 0.000100243
Probability of good choice for population of 412=48.0357
prob - prevprob = 4.77921e-05
Convergence test: (sum - prevsum)/prevsum = 9.95027e-05
Probability of good choice for population of 414=48.0405
prob - prevprob = 4.74458e-05
Convergence test: (sum - prevsum)/prevsum = 9.87718e-05
Probability of good choice for population of 416=48.0452
prob - prevprob = 4.71036e-05
Convergence test: (sum - prevsum)/prevsum = 9.80498e-05
Probability of good choice for population of 418=48.0499
prob - prevprob = 4.67656e-05
Convergence test: (sum - prevsum)/prevsum = 9.73366e-05
Probability of good choice for population of 420=48.0545
prob - prevprob = 4.64315e-05
Convergence test: (sum - prevsum)/prevsum = 9.66319e-05
Probability of good choice for population of 422=48.0591
prob - prevprob = 4.61014e-05
Convergence test: (sum - prevsum)/prevsum = 9.59357e-05
Probability of good choice for population of 424=48.0637
prob - prevprob = 4.57752e-05
Convergence test: (sum - prevsum)/prevsum = 9.52478e-05
Probability of good choice for population of 426=48.0683
prob - prevprob = 4.54529e-05
Convergence test: (sum - prevsum)/prevsum = 9.4568e-05
Probability of good choice for population of 428=48.0728
prob - prevprob = 4.51343e-05
Convergence test: (sum - prevsum)/prevsum = 9.38963e-05
Probability of good choice for population of 430=48.0772
prob - prevprob = 4.48194e-05
Convergence test: (sum - prevsum)/prevsum = 9.32324e-05
Probability of good choice for population of 432=48.0817
prob - prevprob = 4.45082e-05
Convergence test: (sum - prevsum)/prevsum = 9.25763e-05
Probability of good choice for population of 434=48.0861
prob - prevprob = 4.42005e-05
Convergence test: (sum - prevsum)/prevsum = 9.19279e-05
Probability of good choice for population of 436=48.0905
prob - prevprob = 4.38964e-05
Convergence test: (sum - prevsum)/prevsum = 9.1287e-05
Probability of good choice for population of 438=48.0949

```
prob - prevprob = 4.35957e-05
Convergence test: (sum - prevsum)/prevsum = 9.06534e-05
Probability of good choice for population of 440=48.0992
prob - prevprob = 4.32985e-05
Convergence test: (sum - prevsum)/prevsum = 9.00272e-05
Probability of good choice for population of 442=48.1035
prob - prevprob = 4.30046e-05
Convergence test: (sum - prevsum)/prevsum = 8.94081e-05
Probability of good choice for population of 444=48.1078
prob - prevprob = 4.2714e-05
Convergence test: (sum - prevsum)/prevsum = 8.87961e-05
Probability of good choice for population of 446=48.112
prob - prevprob = 4.24267e-05
Convergence test: (sum - prevsum)/prevsum = 8.81909e-05
Probability of good choice for population of 448=48.1162
prob - prevprob = 4.21426e-05
Convergence test: (sum - prevsum)/prevsum = 8.75926e-05
Probability of good choice for population of 450=48.1204
prob - prevprob = 4.18616e-05
Convergence test: (sum - prevsum)/prevsum = 8.70011e-05
Probability of good choice for population of 452=48.1246
prob - prevprob = 4.15838e-05
Convergence test: (sum - prevsum)/prevsum = 8.64161e-05
Probability of good choice for population of 454=48.1287
prob - prevprob = 4.1309e-05
Convergence test: (sum - prevsum)/prevsum = 8.58377e-05
Probability of good choice for population of 456=48.1328
prob - prevprob = 4.10372e-05
Convergence test: (sum - prevsum)/prevsum = 8.52656e-05
Probability of good choice for population of 458=48.1369
prob - prevprob = 4.07684e-05
Convergence test: (sum - prevsum)/prevsum = 8.46999e-05
Probability of good choice for population of 460=48.1409
prob - prevprob = 4.05026e-05
Convergence test: (sum - prevsum)/prevsum = 8.41404e-05
Probability of good choice for population of 462=48.145
prob - prevprob = 4.02396e-05
Convergence test: (sum - prevsum)/prevsum = 8.3587e-05
Probability of good choice for population of 464=48.149
prob - prevprob = 3.99794e-05
Convergence test: (sum - prevsum)/prevsum = 8.30396e-05
Probability of good choice for population of 466=48.1529
prob - prevprob = 3.9722e-05
Convergence test: (sum - prevsum)/prevsum = 8.24982e-05
Probability of good choice for population of 468=48.1569
prob - prevprob = 3.94674e-05
Convergence test: (sum - prevsum)/prevsum = 8.19626e-05
Probability of good choice for population of 470=48.1608
```

```
prob - prevprob = 3.92155e-05
Convergence test: (sum - prevsum)/prevsum = 8.14327e-05
Probability of good choice for population of 472=48.1647
prob - prevprob = 3.89662e-05
Convergence test: (sum - prevsum)/prevsum = 8.09086e-05
Probability of good choice for population of 474=48.1686
prob - prevprob = 3.87196e-05
Convergence test: (sum - prevsum)/prevsum = 8.039e-05
Probability of good choice for population of 476=48.1724
prob - prevprob = 3.84756e-05
Convergence test: (sum - prevsum)/prevsum = 7.98769e-05
Probability of good choice for population of 478=48.1762
prob - prevprob = 3.82341e-05
Convergence test: (sum - prevsum)/prevsum = 7.93692e-05
Probability of good choice for population of 480=48.18
prob - prevprob = 3.79951e-05
Convergence test: (sum - prevsum)/prevsum = 7.88669e-05
Probability of good choice for population of 482=48.1838
prob - prevprob = 3.77586e-05
Convergence test: (sum - prevsum)/prevsum = 7.83699e-05
Probability of good choice for population of 484=48.1876
prob - prevprob = 3.75246e-05
Convergence test: (sum - prevsum)/prevsum = 7.7878e-05
Probability of good choice for population of 486=48.1913
prob - prevprob = 3.7293e-05
Convergence test: (sum - prevsum)/prevsum = 7.73912e-05
Probability of good choice for population of 488=48.195
prob - prevprob = 3.70637e-05
Convergence test: (sum - prevsum)/prevsum = 7.69095e-05
Probability of good choice for population of 490=48.1987
prob - prevprob = 3.68368e-05
Convergence test: (sum - prevsum)/prevsum = 7.64328e-05
Probability of good choice for population of 492=48.2023
prob - prevprob = 3.66122e-05
Convergence test: (sum - prevsum)/prevsum = 7.59609e-05
Probability of good choice for population of 494=48.206
prob - prevprob = 3.63898e-05
Convergence test: (sum - prevsum)/prevsum = 7.54939e-05
Probability of good choice for population of 496=48.2096
prob - prevprob = 3.61697e-05
Convergence test: (sum - prevsum)/prevsum = 7.50316e-05
Probability of good choice for population of 498=48.2132
prob - prevprob = 3.59518e-05
Convergence test: (sum - prevsum)/prevsum = 7.4574e-05
Probability of good choice for population of 500=48.2168
prob - prevprob = 3.57361e-05
Convergence test: (sum - prevsum)/prevsum = 7.4121e-05
Probability of good choice for population of 502=48.2203
```

```
prob - prevprob = 3.55226e-05
Convergence test: (sum - prevsum)/prevsum = 7.36726e-05
Probability of good choice for population of 504=48.2239
prob - prevprob = 3.53111e-05
Convergence test: (sum - prevsum)/prevsum = 7.32287e-05
Probability of good choice for population of 506=48.2274
prob - prevprob = 3.51018e-05
Convergence test: (sum - prevsum)/prevsum = 7.27892e-05
Probability of good choice for population of 508=48.2309
prob - prevprob = 3.48945e-05
Convergence test: (sum - prevsum)/prevsum = 7.23541e-05
Probability of good choice for population of 510=48.2343
prob - prevprob = 3.46892e-05
Convergence test: (sum - prevsum)/prevsum = 7.19233e-05
Probability of good choice for population of 512=48.2378
prob - prevprob = 3.44859e-05
Convergence test: (sum - prevsum)/prevsum = 7.14967e-05
Probability of good choice for population of 514=48.2412
prob - prevprob = 3.42847e-05
Convergence test: (sum - prevsum)/prevsum = 7.10743e-05
Probability of good choice for population of 516=48.2446
prob - prevprob = 3.40853e-05
Convergence test: (sum - prevsum)/prevsum = 7.06561e-05
Probability of good choice for population of 518=48.248
prob - prevprob = 3.38879e-05
Convergence test: (sum - prevsum)/prevsum = 7.02419e-05
Probability of good choice for population of 520=48.2514
prob - prevprob = 3.36924e-05
Convergence test: (sum - prevsum)/prevsum = 6.98318e-05
Probability of good choice for population of 522=48.2547
prob - prevprob = 3.34988e-05
Convergence test: (sum - prevsum)/prevsum = 6.94256e-05
Probability of good choice for population of 524=48.258
prob - prevprob = 3.3307e-05
Convergence test: (sum - prevsum)/prevsum = 6.90233e-05
Probability of good choice for population of 526=48.2614
prob - prevprob = 3.3117e-05
Convergence test: (sum - prevsum)/prevsum = 6.86249e-05
Probability of good choice for population of 528=48.2646
prob - prevprob = 3.29289e-05
Convergence test: (sum - prevsum)/prevsum = 6.82303e-05
Probability of good choice for population of 530=48.2679
prob - prevprob = 3.27425e-05
Convergence test: (sum - prevsum)/prevsum = 6.78395e-05
Probability of good choice for population of 532=48.2712
prob - prevprob = 3.25578e-05
Convergence test: (sum - prevsum)/prevsum = 6.74523e-05
Probability of good choice for population of 534=48.2744
```

```
prob - prevprob = 3.23749e-05
Convergence test: (sum - prevsum)/prevsum = 6.70689e-05
Probability of good choice for population of 536=48.2776
prob - prevprob = 3.21937e-05
Convergence test: (sum - prevsum)/prevsum = 6.6689e-05
Probability of good choice for population of 538=48.2808
prob - prevprob = 3.20142e-05
Convergence test: (sum - prevsum)/prevsum = 6.63127e-05
Probability of good choice for population of 540=48.284
prob - prevprob = 3.18364e-05
Convergence test: (sum - prevsum)/prevsum = 6.59399e-05
Probability of good choice for population of 542=48.2872
prob - prevprob = 3.16601e-05
Convergence test: (sum - prevsum)/prevsum = 6.55706e-05
Probability of good choice for population of 544=48.2903
prob - prevprob = 3.14855e-05
Convergence test: (sum - prevsum)/prevsum = 6.52048e-05
Probability of good choice for population of 546=48.2935
prob - prevprob = 3.13125e-05
Convergence test: (sum - prevsum)/prevsum = 6.48423e-05
Probability of good choice for population of 548=48.2966
prob - prevprob = 3.11411e-05
Convergence test: (sum - prevsum)/prevsum = 6.44831e-05
Probability of good choice for population of 550=48.2997
prob - prevprob = 3.09713e-05
Convergence test: (sum - prevsum)/prevsum = 6.41272e-05
Probability of good choice for population of 552=48.3028
prob - prevprob = 3.08029e-05
Convergence test: (sum - prevsum)/prevsum = 6.37746e-05
Probability of good choice for population of 554=48.3058
prob - prevprob = 3.06361e-05
Convergence test: (sum - prevsum)/prevsum = 6.34252e-05
Probability of good choice for population of 556=48.3089
prob - prevprob = 3.04708e-05
Convergence test: (sum - prevsum)/prevsum = 6.3079e-05
Probability of good choice for population of 558=48.3119
prob - prevprob = 3.0307e-05
Convergence test: (sum - prevsum)/prevsum = 6.27359e-05
Probability of good choice for population of 560=48.3149
prob - prevprob = 3.01447e-05
Convergence test: (sum - prevsum)/prevsum = 6.23959e-05
Probability of good choice for population of 562=48.3179
prob - prevprob = 2.99837e-05
Convergence test: (sum - prevsum)/prevsum = 6.2059e-05
Probability of good choice for population of 564=48.3209
prob - prevprob = 2.98243e-05
Convergence test: (sum - prevsum)/prevsum = 6.17251e-05
Probability of good choice for population of 566=48.3239
```

```
prob - prevprob = 2.96662e-05
Convergence test: (sum - prevsum)/prevsum = 6.13941e-05
Probability of good choice for population of 568=48.3268
prob - prevprob = 2.95095e-05
Convergence test: (sum - prevsum)/prevsum = 6.10661e-05
Probability of good choice for population of 570=48.3297
prob - prevprob = 2.93542e-05
Convergence test: (sum - prevsum)/prevsum = 6.0741e-05
Probability of good choice for population of 572=48.3327
prob - prevprob = 2.92002e-05
Convergence test: (sum - prevsum)/prevsum = 6.04187e-05
Probability of good choice for population of 574=48.3356
prob - prevprob = 2.90476e-05
Convergence test: (sum - prevsum)/prevsum = 6.00993e-05
Probability of good choice for population of 576=48.3385
prob - prevprob = 2.88963e-05
Convergence test: (sum - prevsum)/prevsum = 5.97827e-05
Probability of good choice for population of 578=48.3413
prob - prevprob = 2.87463e-05
Convergence test: (sum - prevsum)/prevsum = 5.94689e-05
Probability of good choice for population of 580=48.3442
prob - prevprob = 2.85976e-05
Convergence test: (sum - prevsum)/prevsum = 5.91578e-05
Probability of good choice for population of 582=48.347
prob - prevprob = 2.84502e-05
Convergence test: (sum - prevsum)/prevsum = 5.88493e-05
Probability of good choice for population of 584=48.3499
prob - prevprob = 2.83041e-05
Convergence test: (sum - prevsum)/prevsum = 5.85436e-05
Probability of good choice for population of 586=48.3527
prob - prevprob = 2.81592e-05
Convergence test: (sum - prevsum)/prevsum = 5.82405e-05
Probability of good choice for population of 588=48.3555
prob - prevprob = 2.80155e-05
Convergence test: (sum - prevsum)/prevsum = 5.79399e-05
Probability of good choice for population of 590=48.3583
prob - prevprob = 2.78731e-05
Convergence test: (sum - prevsum)/prevsum = 5.7642e-05
Probability of good choice for population of 592=48.361
prob - prevprob = 2.77318e-05
Convergence test: (sum - prevsum)/prevsum = 5.73466e-05
Probability of good choice for population of 594=48.3638
prob - prevprob = 2.75918e-05
Convergence test: (sum - prevsum)/prevsum = 5.70537e-05
Probability of good choice for population of 596=48.3666
prob - prevprob = 2.74529e-05
Convergence test: (sum - prevsum)/prevsum = 5.67633e-05
Probability of good choice for population of 598=48.3693
```

```
prob - prevprob = 2.73151e-05
Convergence test: (sum - prevsum)/prevsum = 5.64753e-05
Probability of good choice for population of 600=48.372
prob - prevprob = 2.71786e-05
Convergence test: (sum - prevsum)/prevsum = 5.61897e-05
Probability of good choice for population of 602=48.3747
prob - prevprob = 2.70431e-05
Convergence test: (sum - prevsum)/prevsum = 5.59066e-05
Probability of good choice for population of 604=48.3774
prob - prevprob = 2.69088e-05
Convergence test: (sum - prevsum)/prevsum = 5.56258e-05
Probability of good choice for population of 606=48.3801
prob - prevprob = 2.67756e-05
Convergence test: (sum - prevsum)/prevsum = 5.53473e-05
Probability of good choice for population of 608=48.3827
prob - prevprob = 2.66435e-05
Convergence test: (sum - prevsum)/prevsum = 5.50712e-05
Probability of good choice for population of 610=48.3854
prob - prevprob = 2.65125e-05
Convergence test: (sum - prevsum)/prevsum = 5.47973e-05
Probability of good choice for population of 612=48.388
prob - prevprob = 2.63825e-05
Convergence test: (sum - prevsum)/prevsum = 5.45257e-05
Probability of good choice for population of 614=48.3907
prob - prevprob = 2.62536e-05
Convergence test: (sum - prevsum)/prevsum = 5.42564e-05
Probability of good choice for population of 616=48.3933
prob - prevprob = 2.61257e-05
Convergence test: (sum - prevsum)/prevsum = 5.39892e-05
Probability of good choice for population of 618=48.3959
prob - prevprob = 2.59989e-05
Convergence test: (sum - prevsum)/prevsum = 5.37242e-05
Probability of good choice for population of 620=48.3985
prob - prevprob = 2.58731e-05
Convergence test: (sum - prevsum)/prevsum = 5.34614e-05
Probability of good choice for population of 622=48.401
prob - prevprob = 2.57483e-05
Convergence test: (sum - prevsum)/prevsum = 5.32007e-05
Probability of good choice for population of 624=48.4036
prob - prevprob = 2.56245e-05
Convergence test: (sum - prevsum)/prevsum = 5.29421e-05
Probability of good choice for population of 626=48.4061
prob - prevprob = 2.55017e-05
Convergence test: (sum - prevsum)/prevsum = 5.26856e-05
Probability of good choice for population of 628=48.4087
prob - prevprob = 2.53799e-05
Convergence test: (sum - prevsum)/prevsum = 5.24311e-05
Probability of good choice for population of 630=48.4112
```

```
prob - prevprob = 2.5259e-05
Convergence test: (sum - prevsum)/prevsum = 5.21787e-05
Probability of good choice for population of 632=48.4137
prob - prevprob = 2.51391e-05
Convergence test: (sum - prevsum)/prevsum = 5.19283e-05
Probability of good choice for population of 634=48.4162
prob - prevprob = 2.50202e-05
Convergence test: (sum - prevsum)/prevsum = 5.16799e-05
Probability of good choice for population of 636=48.4187
prob - prevprob = 2.49022e-05
Convergence test: (sum - prevsum)/prevsum = 5.14335e-05
Probability of good choice for population of 638=48.4212
prob - prevprob = 2.47851e-05
Convergence test: (sum - prevsum)/prevsum = 5.1189e-05
Probability of good choice for population of 640=48.4237
prob - prevprob = 2.46689e-05
Convergence test: (sum - prevsum)/prevsum = 5.09465e-05
Probability of good choice for population of 642=48.4261
prob - prevprob = 2.45536e-05
Convergence test: (sum - prevsum)/prevsum = 5.07058e-05
Probability of good choice for population of 644=48.4286
prob - prevprob = 2.44392e-05
Convergence test: (sum - prevsum)/prevsum = 5.04671e-05
Probability of good choice for population of 646=48.431
prob - prevprob = 2.43257e-05
Convergence test: (sum - prevsum)/prevsum = 5.02302e-05
Probability of good choice for population of 648=48.4334
prob - prevprob = 2.42131e-05
Convergence test: (sum - prevsum)/prevsum = 4.99951e-05
Probability of good choice for population of 650=48.4358
prob - prevprob = 2.41014e-05
Convergence test: (sum - prevsum)/prevsum = 4.97619e-05
Probability of good choice for population of 652=48.4382
prob - prevprob = 2.39905e-05
Convergence test: (sum - prevsum)/prevsum = 4.95304e-05
Probability of good choice for population of 654=48.4406
prob - prevprob = 2.38804e-05
Convergence test: (sum - prevsum)/prevsum = 4.93008e-05
Probability of good choice for population of 656=48.443
prob - prevprob = 2.37712e-05
Convergence test: (sum - prevsum)/prevsum = 4.90729e-05
Probability of good choice for population of 658=48.4454
prob - prevprob = 2.36628e-05
Convergence test: (sum - prevsum)/prevsum = 4.88468e-05
Probability of good choice for population of 660=48.4477
prob - prevprob = 2.35553e-05
Convergence test: (sum - prevsum)/prevsum = 4.86224e-05
Probability of good choice for population of 662=48.4501
```

```
prob - prevprob = 2.34485e-05
Convergence test: (sum - prevsum)/prevsum = 4.83997e-05
Probability of good choice for population of 664=48.4524
prob - prevprob = 2.33426e-05
Convergence test: (sum - prevsum)/prevsum = 4.81787e-05
Probability of good choice for population of 666=48.4547
prob - prevprob = 2.32374e-05
Convergence test: (sum - prevsum)/prevsum = 4.79593e-05
Probability of good choice for population of 668=48.457
prob - prevprob = 2.31331e-05
Convergence test: (sum - prevsum)/prevsum = 4.77417e-05
Probability of good choice for population of 670=48.4593
prob - prevprob = 2.30295e-05
Convergence test: (sum - prevsum)/prevsum = 4.75256e-05
Probability of good choice for population of 672=48.4616
prob - prevprob = 2.29267e-05
Convergence test: (sum - prevsum)/prevsum = 4.73112e-05
Probability of good choice for population of 674=48.4639
prob - prevprob = 2.28246e-05
Convergence test: (sum - prevsum)/prevsum = 4.70984e-05
Probability of good choice for population of 676=48.4662
prob - prevprob = 2.27233e-05
Convergence test: (sum - prevsum)/prevsum = 4.68872e-05
Probability of good choice for population of 678=48.4684
prob - prevprob = 2.26228e-05
Convergence test: (sum - prevsum)/prevsum = 4.66775e-05
Probability of good choice for population of 680=48.4707
prob - prevprob = 2.2523e-05
Convergence test: (sum - prevsum)/prevsum = 4.64694e-05
Probability of good choice for population of 682=48.4729
prob - prevprob = 2.24239e-05
Convergence test: (sum - prevsum)/prevsum = 4.62628e-05
Probability of good choice for population of 684=48.4752
prob - prevprob = 2.23256e-05
Convergence test: (sum - prevsum)/prevsum = 4.60578e-05
Probability of good choice for population of 686=48.4774
prob - prevprob = 2.22279e-05
Convergence test: (sum - prevsum)/prevsum = 4.58543e-05
Probability of good choice for population of 688=48.4796
prob - prevprob = 2.2131e-05
Convergence test: (sum - prevsum)/prevsum = 4.56522e-05
Probability of good choice for population of 690=48.4818
prob - prevprob = 2.20348e-05
Convergence test: (sum - prevsum)/prevsum = 4.54517e-05
Probability of good choice for population of 692=48.484
prob - prevprob = 2.19393e-05
Convergence test: (sum - prevsum)/prevsum = 4.52526e-05
Probability of good choice for population of 694=48.4862
```

```
prob - prevprob = 2.18444e-05
Convergence test: (sum - prevsum)/prevsum = 4.50549e-05
Probability of good choice for population of 696=48.4884
prob - prevprob = 2.17503e-05
Convergence test: (sum - prevsum)/prevsum = 4.48587e-05
Probability of good choice for population of 698=48.4905
prob - prevprob = 2.16568e-05
Convergence test: (sum - prevsum)/prevsum = 4.46639e-05
Probability of good choice for population of 700=48.4927
prob - prevprob = 2.1564e-05
Convergence test: (sum - prevsum)/prevsum = 4.44705e-05
Probability of good choice for population of 702=48.4948
prob - prevprob = 2.14718e-05
Convergence test: (sum - prevsum)/prevsum = 4.42785e-05
Probability of good choice for population of 704=48.497
prob - prevprob = 2.13803e-05
Convergence test: (sum - prevsum)/prevsum = 4.40878e-05
Probability of good choice for population of 706=48.4991
prob - prevprob = 2.12895e-05
Convergence test: (sum - prevsum)/prevsum = 4.38986e-05
Probability of good choice for population of 708=48.5012
prob - prevprob = 2.11993e-05
Convergence test: (sum - prevsum)/prevsum = 4.37106e-05
Probability of good choice for population of 710=48.5033
prob - prevprob = 2.11097e-05
Convergence test: (sum - prevsum)/prevsum = 4.3524e-05
Probability of good choice for population of 712=48.5054
prob - prevprob = 2.10207e-05
Convergence test: (sum - prevsum)/prevsum = 4.33388e-05
Probability of good choice for population of 714=48.5075
prob - prevprob = 2.09324e-05
Convergence test: (sum - prevsum)/prevsum = 4.31548e-05
Probability of good choice for population of 716=48.5096
prob - prevprob = 2.08447e-05
Convergence test: (sum - prevsum)/prevsum = 4.29721e-05
Probability of good choice for population of 718=48.5117
prob - prevprob = 2.07576e-05
Convergence test: (sum - prevsum)/prevsum = 4.27907e-05
Probability of good choice for population of 720=48.5137
prob - prevprob = 2.06711e-05
Convergence test: (sum - prevsum)/prevsum = 4.26106e-05
Probability of good choice for population of 722=48.5158
prob - prevprob = 2.05852e-05
Convergence test: (sum - prevsum)/prevsum = 4.24318e-05
Probability of good choice for population of 724=48.5179
prob - prevprob = 2.04999e-05
Convergence test: (sum - prevsum)/prevsum = 4.22541e-05
Probability of good choice for population of 726=48.5199
```

```
prob - prevprob = 2.04152e-05
Convergence test: (sum - prevsum)/prevsum = 4.20778e-05
Probability of good choice for population of 728=48.5219
prob - prevprob = 2.03311e-05
Convergence test: (sum - prevsum)/prevsum = 4.19026e-05
Probability of good choice for population of 730=48.524
prob - prevprob = 2.02475e-05
Convergence test: (sum - prevsum)/prevsum = 4.17286e-05
Probability of good choice for population of 732=48.526
prob - prevprob = 2.01646e-05
Convergence test: (sum - prevsum)/prevsum = 4.15559e-05
Probability of good choice for population of 734=48.528
prob - prevprob = 2.00821e-05
Convergence test: (sum - prevsum)/prevsum = 4.13843e-05
Probability of good choice for population of 736=48.53
prob - prevprob = 2.00003e-05
Convergence test: (sum - prevsum)/prevsum = 4.12139e-05
Probability of good choice for population of 738=48.532
prob - prevprob = 1.9919e-05
Convergence test: (sum - prevsum)/prevsum = 4.10447e-05
Probability of good choice for population of 740=48.534
prob - prevprob = 1.98382e-05
Convergence test: (sum - prevsum)/prevsum = 4.08766e-05
Probability of good choice for population of 742=48.5359
prob - prevprob = 1.9758e-05
Convergence test: (sum - prevsum)/prevsum = 4.07097e-05
Probability of good choice for population of 744=48.5379
prob - prevprob = 1.96784e-05
Convergence test: (sum - prevsum)/prevsum = 4.05439e-05
Probability of good choice for population of 746=48.5399
prob - prevprob = 1.95992e-05
Convergence test: (sum - prevsum)/prevsum = 4.03792e-05
Probability of good choice for population of 748=48.5418
prob - prevprob = 1.95206e-05
Convergence test: (sum - prevsum)/prevsum = 4.02156e-05
Probability of good choice for population of 750=48.5438
prob - prevprob = 1.94425e-05
Convergence test: (sum - prevsum)/prevsum = 4.00532e-05
Probability of good choice for population of 752=48.5457
prob - prevprob = 1.9365e-05
Convergence test: (sum - prevsum)/prevsum = 3.98918e-05
Probability of good choice for population of 754=48.5476
prob - prevprob = 1.92879e-05
Convergence test: (sum - prevsum)/prevsum = 3.97315e-05
Probability of good choice for population of 756=48.5495
prob - prevprob = 1.92114e-05
Convergence test: (sum - prevsum)/prevsum = 3.95722e-05
Probability of good choice for population of 758=48.5515
```

```
prob - prevprob = 1.91353e-05
Convergence test: (sum - prevsum)/prevsum = 3.94141e-05
Probability of good choice for population of 760=48.5534
prob - prevprob = 1.90598e-05
Convergence test: (sum - prevsum)/prevsum = 3.92569e-05
Probability of good choice for population of 762=48.5553
prob - prevprob = 1.89848e-05
Convergence test: (sum - prevsum)/prevsum = 3.91008e-05
Probability of good choice for population of 764=48.5571
prob - prevprob = 1.89102e-05
Convergence test: (sum - prevsum)/prevsum = 3.89458e-05
Probability of good choice for population of 766=48.559
prob - prevprob = 1.88362e-05
Convergence test: (sum - prevsum)/prevsum = 3.87917e-05
Probability of good choice for population of 768=48.5609
prob - prevprob = 1.87626e-05
Convergence test: (sum - prevsum)/prevsum = 3.86387e-05
Probability of good choice for population of 770=48.5628
prob - prevprob = 1.86895e-05
Convergence test: (sum - prevsum)/prevsum = 3.84867e-05
Probability of good choice for population of 772=48.5646
prob - prevprob = 1.86169e-05
Convergence test: (sum - prevsum)/prevsum = 3.83357e-05
Probability of good choice for population of 774=48.5665
prob - prevprob = 1.85447e-05
Convergence test: (sum - prevsum)/prevsum = 3.81856e-05
Probability of good choice for population of 776=48.5683
prob - prevprob = 1.8473e-05
Convergence test: (sum - prevsum)/prevsum = 3.80365e-05
Probability of good choice for population of 778=48.5702
prob - prevprob = 1.84018e-05
Convergence test: (sum - prevsum)/prevsum = 3.78884e-05
Probability of good choice for population of 780=48.572
prob - prevprob = 1.8331e-05
Convergence test: (sum - prevsum)/prevsum = 3.77413e-05
Probability of good choice for population of 782=48.5738
prob - prevprob = 1.82607e-05
Convergence test: (sum - prevsum)/prevsum = 3.75951e-05
Probability of good choice for population of 784=48.5757
prob - prevprob = 1.81908e-05
Convergence test: (sum - prevsum)/prevsum = 3.74498e-05
Probability of good choice for population of 786=48.5775
prob - prevprob = 1.81214e-05
Convergence test: (sum - prevsum)/prevsum = 3.73055e-05
Probability of good choice for population of 788=48.5793
prob - prevprob = 1.80524e-05
Convergence test: (sum - prevsum)/prevsum = 3.7162e-05
Probability of good choice for population of 790=48.5811
```

```
prob - prevprob = 1.79838e-05
Convergence test: (sum - prevsum)/prevsum = 3.70195e-05
Probability of good choice for population of 792=48.5829
prob - prevprob = 1.79157e-05
Convergence test: (sum - prevsum)/prevsum = 3.6878e-05
Probability of good choice for population of 794=48.5847
prob - prevprob = 1.7848e-05
Convergence test: (sum - prevsum)/prevsum = 3.67373e-05
Probability of good choice for population of 796=48.5864
prob - prevprob = 1.77807e-05
Convergence test: (sum - prevsum)/prevsum = 3.65975e-05
Probability of good choice for population of 798=48.5882
prob - prevprob = 1.77139e-05
Convergence test: (sum - prevsum)/prevsum = 3.64585e-05
Probability of good choice for population of 800=48.59
prob - prevprob = 1.76475e-05
Convergence test: (sum - prevsum)/prevsum = 3.63205e-05
Probability of good choice for population of 802=48.5917
prob - prevprob = 1.75815e-05
Convergence test: (sum - prevsum)/prevsum = 3.61833e-05
Probability of good choice for population of 804=48.5935
prob - prevprob = 1.75159e-05
Convergence test: (sum - prevsum)/prevsum = 3.6047e-05
Probability of good choice for population of 806=48.5952
prob - prevprob = 1.74507e-05
Convergence test: (sum - prevsum)/prevsum = 3.59115e-05
Probability of good choice for population of 808=48.597
prob - prevprob = 1.73859e-05
Convergence test: (sum - prevsum)/prevsum = 3.57769e-05
Probability of good choice for population of 810=48.5987
prob - prevprob = 1.73215e-05
Convergence test: (sum - prevsum)/prevsum = 3.56431e-05
Probability of good choice for population of 812=48.6004
prob - prevprob = 1.72575e-05
Convergence test: (sum - prevsum)/prevsum = 3.55102e-05
Probability of good choice for population of 814=48.6021
prob - prevprob = 1.71939e-05
Convergence test: (sum - prevsum)/prevsum = 3.53781e-05
Probability of good choice for population of 816=48.6039
prob - prevprob = 1.71307e-05
Convergence test: (sum - prevsum)/prevsum = 3.52467e-05
Probability of good choice for population of 818=48.6056
prob - prevprob = 1.70678e-05
Convergence test: (sum - prevsum)/prevsum = 3.51162e-05
Probability of good choice for population of 820=48.6073
prob - prevprob = 1.70054e-05
Convergence test: (sum - prevsum)/prevsum = 3.49865e-05
Probability of good choice for population of 822=48.609
```

```
prob - prevprob = 1.69433e-05
Convergence test: (sum - prevsum)/prevsum = 3.48576e-05
Probability of good choice for population of 824=48.6106
prob - prevprob = 1.68816e-05
Convergence test: (sum - prevsum)/prevsum = 3.47295e-05
Probability of good choice for population of 826=48.6123
prob - prevprob = 1.68203e-05
Convergence test: (sum - prevsum)/prevsum = 3.46022e-05
Probability of good choice for population of 828=48.614
prob - prevprob = 1.67594e-05
Convergence test: (sum - prevsum)/prevsum = 3.44756e-05
Probability of good choice for population of 830=48.6157
prob - prevprob = 1.66988e-05
Convergence test: (sum - prevsum)/prevsum = 3.43498e-05
Probability of good choice for population of 832=48.6173
prob - prevprob = 1.66386e-05
Convergence test: (sum - prevsum)/prevsum = 3.42248e-05
Probability of good choice for population of 834=48.619
prob - prevprob = 1.65788e-05
Convergence test: (sum - prevsum)/prevsum = 3.41005e-05
Probability of good choice for population of 836=48.6206
prob - prevprob = 1.65193e-05
Convergence test: (sum - prevsum)/prevsum = 3.3977e-05
Probability of good choice for population of 838=48.6223
prob - prevprob = 1.64601e-05
Convergence test: (sum - prevsum)/prevsum = 3.38542e-05
Probability of good choice for population of 840=48.6239
prob - prevprob = 1.64013e-05
Convergence test: (sum - prevsum)/prevsum = 3.37321e-05
Probability of good choice for population of 842=48.6256
prob - prevprob = 1.63429e-05
Convergence test: (sum - prevsum)/prevsum = 3.36108e-05
Probability of good choice for population of 844=48.6272
prob - prevprob = 1.62848e-05
Convergence test: (sum - prevsum)/prevsum = 3.34902e-05
Probability of good choice for population of 846=48.6288
prob - prevprob = 1.62271e-05
Convergence test: (sum - prevsum)/prevsum = 3.33703e-05
Probability of good choice for population of 848=48.6304
prob - prevprob = 1.61697e-05
Convergence test: (sum - prevsum)/prevsum = 3.32512e-05
Probability of good choice for population of 850=48.632
prob - prevprob = 1.61126e-05
Convergence test: (sum - prevsum)/prevsum = 3.31327e-05
Probability of good choice for population of 852=48.6336
prob - prevprob = 1.60558e-05
Convergence test: (sum - prevsum)/prevsum = 3.3015e-05
Probability of good choice for population of 854=48.6352
```

```
prob - prevprob = 1.59994e-05
Convergence test: (sum - prevsum)/prevsum = 3.28979e-05
Probability of good choice for population of 856=48.6368
prob - prevprob = 1.59434e-05
Convergence test: (sum - prevsum)/prevsum = 3.27815e-05
Probability of good choice for population of 858=48.6384
prob - prevprob = 1.58876e-05
Convergence test: (sum - prevsum)/prevsum = 3.26658e-05
Probability of good choice for population of 860=48.64
prob - prevprob = 1.58322e-05
Convergence test: (sum - prevsum)/prevsum = 3.25508e-05
Probability of good choice for population of 862=48.6416
prob - prevprob = 1.57771e-05
Convergence test: (sum - prevsum)/prevsum = 3.24365e-05
Probability of good choice for population of 864=48.6432
prob - prevprob = 1.57223e-05
Convergence test: (sum - prevsum)/prevsum = 3.23228e-05
Probability of good choice for population of 866=48.6447
prob - prevprob = 1.56679e-05
Convergence test: (sum - prevsum)/prevsum = 3.22098e-05
Probability of good choice for population of 868=48.6463
prob - prevprob = 1.56137e-05
Convergence test: (sum - prevsum)/prevsum = 3.20974e-05
Probability of good choice for population of 870=48.6478
prob - prevprob = 1.55599e-05
Convergence test: (sum - prevsum)/prevsum = 3.19857e-05
Probability of good choice for population of 872=48.6494
prob - prevprob = 1.55063e-05
Convergence test: (sum - prevsum)/prevsum = 3.18747e-05
Probability of good choice for population of 874=48.6509
prob - prevprob = 1.54531e-05
Convergence test: (sum - prevsum)/prevsum = 3.17642e-05
Probability of good choice for population of 876=48.6525
prob - prevprob = 1.54002e-05
Convergence test: (sum - prevsum)/prevsum = 3.16544e-05
Probability of good choice for population of 878=48.654
prob - prevprob = 1.53476e-05
Convergence test: (sum - prevsum)/prevsum = 3.15453e-05
Probability of good choice for population of 880=48.6555
prob - prevprob = 1.52952e-05
Convergence test: (sum - prevsum)/prevsum = 3.14368e-05
Probability of good choice for population of 882=48.6571
prob - prevprob = 1.52432e-05
Convergence test: (sum - prevsum)/prevsum = 3.13288e-05
Probability of good choice for population of 884=48.6586
prob - prevprob = 1.51915e-05
Convergence test: (sum - prevsum)/prevsum = 3.12215e-05
Probability of good choice for population of 886=48.6601
```

```
prob - prevprob = 1.51401e-05
Convergence test: (sum - prevsum)/prevsum = 3.11149e-05
Probability of good choice for population of 888=48.6616
prob - prevprob = 1.50889e-05
Convergence test: (sum - prevsum)/prevsum = 3.10088e-05
Probability of good choice for population of 890=48.6631
prob - prevprob = 1.5038e-05
Convergence test: (sum - prevsum)/prevsum = 3.09033e-05
Probability of good choice for population of 892=48.6646
prob - prevprob = 1.49875e-05
Convergence test: (sum - prevsum)/prevsum = 3.07984e-05
Probability of good choice for population of 894=48.6661
prob - prevprob = 1.49372e-05
Convergence test: (sum - prevsum)/prevsum = 3.06941e-05
Probability of good choice for population of 896=48.6676
prob - prevprob = 1.48872e-05
Convergence test: (sum - prevsum)/prevsum = 3.05904e-05
Probability of good choice for population of 898=48.6691
prob - prevprob = 1.48374e-05
Convergence test: (sum - prevsum)/prevsum = 3.04873e-05
Probability of good choice for population of 900=48.6706
prob - prevprob = 1.4788e-05
Convergence test: (sum - prevsum)/prevsum = 3.03847e-05
Probability of good choice for population of 902=48.672
prob - prevprob = 1.47388e-05
Convergence test: (sum - prevsum)/prevsum = 3.02827e-05
Probability of good choice for population of 904=48.6735
prob - prevprob = 1.46899e-05
Convergence test: (sum - prevsum)/prevsum = 3.01813e-05
Probability of good choice for population of 906=48.675
prob - prevprob = 1.46412e-05
Convergence test: (sum - prevsum)/prevsum = 3.00805e-05
Probability of good choice for population of 908=48.6764
prob - prevprob = 1.45929e-05
Convergence test: (sum - prevsum)/prevsum = 2.99802e-05
Probability of good choice for population of 910=48.6779
prob - prevprob = 1.45447e-05
Convergence test: (sum - prevsum)/prevsum = 2.98805e-05
Probability of good choice for population of 912=48.6793
prob - prevprob = 1.44969e-05
Convergence test: (sum - prevsum)/prevsum = 2.97813e-05
Probability of good choice for population of 914=48.6808
prob - prevprob = 1.44493e-05
Convergence test: (sum - prevsum)/prevsum = 2.96827e-05
Probability of good choice for population of 916=48.6822
prob - prevprob = 1.4402e-05
Convergence test: (sum - prevsum)/prevsum = 2.95846e-05
Probability of good choice for population of 918=48.6837
```

```
prob - prevprob = 1.43549e-05
Convergence test: (sum - prevsum)/prevsum = 2.9487e-05
Probability of good choice for population of 920=48.6851
prob - prevprob = 1.43081e-05
Convergence test: (sum - prevsum)/prevsum = 2.939e-05
Probability of good choice for population of 922=48.6865
prob - prevprob = 1.42616e-05
Convergence test: (sum - prevsum)/prevsum = 2.92935e-05
Probability of good choice for population of 924=48.6879
prob - prevprob = 1.42153e-05
Convergence test: (sum - prevsum)/prevsum = 2.91975e-05
Probability of good choice for population of 926=48.6893
prob - prevprob = 1.41692e-05
Convergence test: (sum - prevsum)/prevsum = 2.91021e-05
Probability of good choice for population of 928=48.6908
prob - prevprob = 1.41234e-05
Convergence test: (sum - prevsum)/prevsum = 2.90072e-05
Probability of good choice for population of 930=48.6922
prob - prevprob = 1.40778e-05
Convergence test: (sum - prevsum)/prevsum = 2.89128e-05
Probability of good choice for population of 932=48.6936
prob - prevprob = 1.40325e-05
Convergence test: (sum - prevsum)/prevsum = 2.88189e-05
Probability of good choice for population of 934=48.695
prob - prevprob = 1.39875e-05
Convergence test: (sum - prevsum)/prevsum = 2.87255e-05
Probability of good choice for population of 936=48.6964
prob - prevprob = 1.39426e-05
Convergence test: (sum - prevsum)/prevsum = 2.86326e-05
Probability of good choice for population of 938=48.6978
prob - prevprob = 1.3898e-05
Convergence test: (sum - prevsum)/prevsum = 2.85402e-05
Probability of good choice for population of 940=48.6991
prob - prevprob = 1.38537e-05
Convergence test: (sum - prevsum)/prevsum = 2.84483e-05
Probability of good choice for population of 942=48.7005
prob - prevprob = 1.38096e-05
Convergence test: (sum - prevsum)/prevsum = 2.83569e-05
Probability of good choice for population of 944=48.7019
prob - prevprob = 1.37657e-05
Convergence test: (sum - prevsum)/prevsum = 2.8266e-05
Probability of good choice for population of 946=48.7033
prob - prevprob = 1.3722e-05
Convergence test: (sum - prevsum)/prevsum = 2.81755e-05
Probability of good choice for population of 948=48.7046
prob - prevprob = 1.36786e-05
Convergence test: (sum - prevsum)/prevsum = 2.80856e-05
Probability of good choice for population of 950=48.706
```

```
prob - prevprob = 1.36354e-05
Convergence test: (sum - prevsum)/prevsum = 2.79961e-05
Probability of good choice for population of 952=48.7074
prob - prevprob = 1.35924e-05
Convergence test: (sum - prevsum)/prevsum = 2.79071e-05
Probability of good choice for population of 954=48.7087
prob - prevprob = 1.35497e-05
Convergence test: (sum - prevsum)/prevsum = 2.78186e-05
Probability of good choice for population of 956=48.7101
prob - prevprob = 1.35072e-05
Convergence test: (sum - prevsum)/prevsum = 2.77305e-05
Probability of good choice for population of 958=48.7114
prob - prevprob = 1.34649e-05
Convergence test: (sum - prevsum)/prevsum = 2.76429e-05
Probability of good choice for population of 960=48.7128
prob - prevprob = 1.34228e-05
Convergence test: (sum - prevsum)/prevsum = 2.75557e-05
Probability of good choice for population of 962=48.7141
prob - prevprob = 1.33809e-05
Convergence test: (sum - prevsum)/prevsum = 2.7469e-05
Probability of good choice for population of 964=48.7154
prob - prevprob = 1.33393e-05
Convergence test: (sum - prevsum)/prevsum = 2.73828e-05
Probability of good choice for population of 966=48.7168
prob - prevprob = 1.32979e-05
Convergence test: (sum - prevsum)/prevsum = 2.7297e-05
Probability of good choice for population of 968=48.7181
prob - prevprob = 1.32566e-05
Convergence test: (sum - prevsum)/prevsum = 2.72117e-05
Probability of good choice for population of 970=48.7194
prob - prevprob = 1.32156e-05
Convergence test: (sum - prevsum)/prevsum = 2.71268e-05
Probability of good choice for population of 972=48.7207
prob - prevprob = 1.31749e-05
Convergence test: (sum - prevsum)/prevsum = 2.70423e-05
Probability of good choice for population of 974=48.722
prob - prevprob = 1.31343e-05
Convergence test: (sum - prevsum)/prevsum = 2.69583e-05
Probability of good choice for population of 976=48.7233
prob - prevprob = 1.30939e-05
Convergence test: (sum - prevsum)/prevsum = 2.68747e-05
Probability of good choice for population of 978=48.7246
prob - prevprob = 1.30537e-05
Convergence test: (sum - prevsum)/prevsum = 2.67916e-05
Probability of good choice for population of 980=48.726
prob - prevprob = 1.30138e-05
Convergence test: (sum - prevsum)/prevsum = 2.67088e-05
Probability of good choice for population of 982=48.7272
```

```
prob - prevprob = 1.2974e-05
Convergence test: (sum - prevsum)/prevsum = 2.66265e-05
Probability of good choice for population of 984=48.7285
prob - prevprob = 1.29345e-05
Convergence test: (sum - prevsum)/prevsum = 2.65446e-05
Probability of good choice for population of 986=48.7298
prob - prevprob = 1.28951e-05
Convergence test: (sum - prevsum)/prevsum = 2.64632e-05
Probability of good choice for population of 988=48.7311
prob - prevprob = 1.2856e-05
Convergence test: (sum - prevsum)/prevsum = 2.63821e-05
Probability of good choice for population of 990=48.7324
prob - prevprob = 1.2817e-05
Convergence test: (sum - prevsum)/prevsum = 2.63015e-05
Probability of good choice for population of 992=48.7337
prob - prevprob = 1.27782e-05
Convergence test: (sum - prevsum)/prevsum = 2.62212e-05
Probability of good choice for population of 994=48.735
prob - prevprob = 1.27397e-05
Convergence test: (sum - prevsum)/prevsum = 2.61414e-05
Probability of good choice for population of 996=48.7362
prob - prevprob = 1.27013e-05
Convergence test: (sum - prevsum)/prevsum = 2.6062e-05
Probability of good choice for population of 998=48.7375
prob - prevprob = 1.26631e-05
Convergence test: (sum - prevsum)/prevsum = 2.5983e-05
Probability of good choice for population of 1000=48.7387
prob - prevprob = 1.26251e-05
Convergence test: (sum - prevsum)/prevsum = 2.59044e-05
Probability of good choice for population of 1002=48.74
prob - prevprob = 1.25873e-05
Convergence test: (sum - prevsum)/prevsum = 2.58261e-05
Probability of good choice for population of 1004=48.7413
prob - prevprob = 1.25497e-05
Convergence test: (sum - prevsum)/prevsum = 2.57483e-05
Probability of good choice for population of 1006=48.7425
prob - prevprob = 1.25123e-05
Convergence test: (sum - prevsum)/prevsum = 2.56709e-05
Probability of good choice for population of 1008=48.7438
prob - prevprob = 1.24751e-05
Convergence test: (sum - prevsum)/prevsum = 2.55938e-05
Probability of good choice for population of 1010=48.745
prob - prevprob = 1.2438e-05
Convergence test: (sum - prevsum)/prevsum = 2.55171e-05
Probability of good choice for population of 1012=48.7462
prob - prevprob = 1.24011e-05
Convergence test: (sum - prevsum)/prevsum = 2.54408e-05
Probability of good choice for population of 1014=48.7475
```

```
prob - prevprob = 1.23644e-05
Convergence test: (sum - prevsum)/prevsum = 2.53649e-05
Probability of good choice for population of 1016=48.7487
prob - prevprob = 1.23279e-05
Convergence test: (sum - prevsum)/prevsum = 2.52894e-05
Probability of good choice for population of 1018=48.7499
prob - prevprob = 1.22916e-05
Convergence test: (sum - prevsum)/prevsum = 2.52142e-05
Probability of good choice for population of 1020=48.7512
prob - prevprob = 1.22555e-05
Convergence test: (sum - prevsum)/prevsum = 2.51394e-05
Probability of good choice for population of 1022=48.7524
prob - prevprob = 1.22195e-05
Convergence test: (sum - prevsum)/prevsum = 2.5065e-05
Probability of good choice for population of 1024=48.7536
prob - prevprob = 1.21837e-05
Convergence test: (sum - prevsum)/prevsum = 2.49909e-05
Probability of good choice for population of 1026=48.7548
prob - prevprob = 1.21481e-05
Convergence test: (sum - prevsum)/prevsum = 2.49172e-05
Probability of good choice for population of 1028=48.756
prob - prevprob = 1.21126e-05
Convergence test: (sum - prevsum)/prevsum = 2.48439e-05
Probability of good choice for population of 1030=48.7572
prob - prevprob = 1.20773e-05
Convergence test: (sum - prevsum)/prevsum = 2.47709e-05
Probability of good choice for population of 1032=48.7584
prob - prevprob = 1.20422e-05
Convergence test: (sum - prevsum)/prevsum = 2.46983e-05
Probability of good choice for population of 1034=48.7596
prob - prevprob = 1.20073e-05
Convergence test: (sum - prevsum)/prevsum = 2.4626e-05
Probability of good choice for population of 1036=48.7608
prob - prevprob = 1.19725e-05
Convergence test: (sum - prevsum)/prevsum = 2.45541e-05
Probability of good choice for population of 1038=48.762
prob - prevprob = 1.19379e-05
Convergence test: (sum - prevsum)/prevsum = 2.44826e-05
Probability of good choice for population of 1040=48.7632
prob - prevprob = 1.19035e-05
Convergence test: (sum - prevsum)/prevsum = 2.44113e-05
Probability of good choice for population of 1042=48.7644
prob - prevprob = 1.18692e-05
Convergence test: (sum - prevsum)/prevsum = 2.43405e-05
Probability of good choice for population of 1044=48.7656
prob - prevprob = 1.18351e-05
Convergence test: (sum - prevsum)/prevsum = 2.42699e-05
Probability of good choice for population of 1046=48.7668
```

```
prob - prevprob = 1.18011e-05
Convergence test: (sum - prevsum)/prevsum = 2.41997e-05
Probability of good choice for population of 1048=48.768
prob - prevprob = 1.17674e-05
Convergence test: (sum - prevsum)/prevsum = 2.41299e-05
Probability of good choice for population of 1050=48.7691
prob - prevprob = 1.17337e-05
Convergence test: (sum - prevsum)/prevsum = 2.40604e-05
Probability of good choice for population of 1052=48.7703
prob - prevprob = 1.17003e-05
Convergence test: (sum - prevsum)/prevsum = 2.39912e-05
Probability of good choice for population of 1054=48.7715
prob - prevprob = 1.1667e-05
Convergence test: (sum - prevsum)/prevsum = 2.39223e-05
Probability of good choice for population of 1056=48.7726
prob - prevprob = 1.16338e-05
Convergence test: (sum - prevsum)/prevsum = 2.38538e-05
Probability of good choice for population of 1058=48.7738
prob - prevprob = 1.16008e-05
Convergence test: (sum - prevsum)/prevsum = 2.37856e-05
Probability of good choice for population of 1060=48.7749
prob - prevprob = 1.1568e-05
Convergence test: (sum - prevsum)/prevsum = 2.37177e-05
Probability of good choice for population of 1062=48.7761
prob - prevprob = 1.15353e-05
Convergence test: (sum - prevsum)/prevsum = 2.36501e-05
Probability of good choice for population of 1064=48.7773
prob - prevprob = 1.15028e-05
Convergence test: (sum - prevsum)/prevsum = 2.35829e-05
Probability of good choice for population of 1066=48.7784
prob - prevprob = 1.14704e-05
Convergence test: (sum - prevsum)/prevsum = 2.3516e-05
Probability of good choice for population of 1068=48.7795
prob - prevprob = 1.14382e-05
Convergence test: (sum - prevsum)/prevsum = 2.34494e-05
Probability of good choice for population of 1070=48.7807
prob - prevprob = 1.14061e-05
Convergence test: (sum - prevsum)/prevsum = 2.33831e-05
Probability of good choice for population of 1072=48.7818
prob - prevprob = 1.13742e-05
Convergence test: (sum - prevsum)/prevsum = 2.33171e-05
Probability of good choice for population of 1074=48.783
prob - prevprob = 1.13425e-05
Convergence test: (sum - prevsum)/prevsum = 2.32514e-05
Probability of good choice for population of 1076=48.7841
prob - prevprob = 1.13108e-05
Convergence test: (sum - prevsum)/prevsum = 2.3186e-05
Probability of good choice for population of 1078=48.7852
```

```
prob - prevprob = 1.12794e-05
Convergence test: (sum - prevsum)/prevsum = 2.3121e-05
Probability of good choice for population of 1080=48.7863
prob - prevprob = 1.1248e-05
Convergence test: (sum - prevsum)/prevsum = 2.30562e-05
Probability of good choice for population of 1082=48.7875
prob - prevprob = 1.12168e-05
Convergence test: (sum - prevsum)/prevsum = 2.29918e-05
Probability of good choice for population of 1084=48.7886
prob - prevprob = 1.11858e-05
Convergence test: (sum - prevsum)/prevsum = 2.29276e-05
Probability of good choice for population of 1086=48.7897
prob - prevprob = 1.11549e-05
Convergence test: (sum - prevsum)/prevsum = 2.28637e-05
Probability of good choice for population of 1088=48.7908
prob - prevprob = 1.11241e-05
Convergence test: (sum - prevsum)/prevsum = 2.28002e-05
Probability of good choice for population of 1090=48.7919
prob - prevprob = 1.10935e-05
Convergence test: (sum - prevsum)/prevsum = 2.27369e-05
Probability of good choice for population of 1092=48.793
prob - prevprob = 1.1063e-05
Convergence test: (sum - prevsum)/prevsum = 2.26739e-05
Probability of good choice for population of 1094=48.7941
prob - prevprob = 1.10327e-05
Convergence test: (sum - prevsum)/prevsum = 2.26112e-05
Probability of good choice for population of 1096=48.7952
prob - prevprob = 1.10025e-05
Convergence test: (sum - prevsum)/prevsum = 2.25488e-05
Probability of good choice for population of 1098=48.7963
prob - prevprob = 1.09724e-05
Convergence test: (sum - prevsum)/prevsum = 2.24867e-05
Probability of good choice for population of 1100=48.7974
prob - prevprob = 1.09425e-05
Convergence test: (sum - prevsum)/prevsum = 2.24249e-05
Probability of good choice for population of 1102=48.7985
prob - prevprob = 1.09127e-05
Convergence test: (sum - prevsum)/prevsum = 2.23633e-05
Probability of good choice for population of 1104=48.7996
prob - prevprob = 1.08831e-05
Convergence test: (sum - prevsum)/prevsum = 2.23021e-05
Probability of good choice for population of 1106=48.8007
prob - prevprob = 1.08536e-05
Convergence test: (sum - prevsum)/prevsum = 2.22411e-05
Probability of good choice for population of 1108=48.8018
prob - prevprob = 1.08242e-05
Convergence test: (sum - prevsum)/prevsum = 2.21804e-05
Probability of good choice for population of 1110=48.8028
```

```
prob - prevprob = 1.07949e-05
Convergence test: (sum - prevsum)/prevsum = 2.21199e-05
Probability of good choice for population of 1112=48.8039
prob - prevprob = 1.07658e-05
Convergence test: (sum - prevsum)/prevsum = 2.20598e-05
Probability of good choice for population of 1114=48.805
prob - prevprob = 1.07368e-05
Convergence test: (sum - prevsum)/prevsum = 2.19999e-05
Probability of good choice for population of 1116=48.8061
prob - prevprob = 1.07079e-05
Convergence test: (sum - prevsum)/prevsum = 2.19402e-05
Probability of good choice for population of 1118=48.8071
prob - prevprob = 1.06792e-05
Convergence test: (sum - prevsum)/prevsum = 2.18809e-05
Probability of good choice for population of 1120=48.8082
prob - prevprob = 1.06506e-05
Convergence test: (sum - prevsum)/prevsum = 2.18218e-05
Probability of good choice for population of 1122=48.8093
prob - prevprob = 1.06221e-05
Convergence test: (sum - prevsum)/prevsum = 2.1763e-05
Probability of good choice for population of 1124=48.8103
prob - prevprob = 1.05938e-05
Convergence test: (sum - prevsum)/prevsum = 2.17044e-05
Probability of good choice for population of 1126=48.8114
prob - prevprob = 1.05655e-05
Convergence test: (sum - prevsum)/prevsum = 2.16461e-05
Probability of good choice for population of 1128=48.8124
prob - prevprob = 1.05374e-05
Convergence test: (sum - prevsum)/prevsum = 2.15881e-05
Probability of good choice for population of 1130=48.8135
prob - prevprob = 1.05095e-05
Convergence test: (sum - prevsum)/prevsum = 2.15303e-05
Probability of good choice for population of 1132=48.8145
prob - prevprob = 1.04816e-05
Convergence test: (sum - prevsum)/prevsum = 2.14728e-05
Probability of good choice for population of 1134=48.8156
prob - prevprob = 1.04539e-05
Convergence test: (sum - prevsum)/prevsum = 2.14155e-05
Probability of good choice for population of 1136=48.8166
prob - prevprob = 1.04263e-05
Convergence test: (sum - prevsum)/prevsum = 2.13585e-05
Probability of good choice for population of 1138=48.8177
prob - prevprob = 1.03988e-05
Convergence test: (sum - prevsum)/prevsum = 2.13018e-05
Probability of good choice for population of 1140=48.8187
prob - prevprob = 1.03714e-05
Convergence test: (sum - prevsum)/prevsum = 2.12452e-05
Probability of good choice for population of 1142=48.8197
```

```
prob - prevprob = 1.03442e-05
Convergence test: (sum - prevsum)/prevsum = 2.1189e-05
Probability of good choice for population of 1144=48.8208
prob - prevprob = 1.03171e-05
Convergence test: (sum - prevsum)/prevsum = 2.1133e-05
Probability of good choice for population of 1146=48.8218
prob - prevprob = 1.02901e-05
Convergence test: (sum - prevsum)/prevsum = 2.10772e-05
Probability of good choice for population of 1148=48.8228
prob - prevprob = 1.02632e-05
Convergence test: (sum - prevsum)/prevsum = 2.10217e-05
Probability of good choice for population of 1150=48.8238
prob - prevprob = 1.02364e-05
Convergence test: (sum - prevsum)/prevsum = 2.09664e-05
Probability of good choice for population of 1152=48.8249
prob - prevprob = 1.02097e-05
Convergence test: (sum - prevsum)/prevsum = 2.09114e-05
Probability of good choice for population of 1154=48.8259
prob - prevprob = 1.01832e-05
Convergence test: (sum - prevsum)/prevsum = 2.08566e-05
Probability of good choice for population of 1156=48.8269
prob - prevprob = 1.01568e-05
Convergence test: (sum - prevsum)/prevsum = 2.0802e-05
Probability of good choice for population of 1158=48.8279
prob - prevprob = 1.01304e-05
Convergence test: (sum - prevsum)/prevsum = 2.07477e-05
Probability of good choice for population of 1160=48.8289
prob - prevprob = 1.01042e-05
Convergence test: (sum - prevsum)/prevsum = 2.06936e-05
Probability of good choice for population of 1162=48.8299
prob - prevprob = 1.00782e-05
Convergence test: (sum - prevsum)/prevsum = 2.06397e-05
Probability of good choice for population of 1164=48.8309
prob - prevprob = 1.00522e-05
Convergence test: (sum - prevsum)/prevsum = 2.05861e-05
Probability of good choice for population of 1166=48.8319
prob - prevprob = 1.00263e-05
Convergence test: (sum - prevsum)/prevsum = 2.05327e-05
Probability of good choice for population of 1168=48.8329
prob - prevprob = 1.00006e-05
Convergence test: (sum - prevsum)/prevsum = 2.04796e-05
Probability of good choice for population of 1170=48.8339
prob - prevprob = 9.97493e-06
Convergence test: (sum - prevsum)/prevsum = 2.04266e-05
Probability of good choice for population of 1172=48.8349
prob - prevprob = 9.9494e-06
Convergence test: (sum - prevsum)/prevsum = 2.03739e-05
Probability of good choice for population of 1174=48.8359
```

```
prob - prevprob = 9.92397e-06
Convergence test: (sum - prevsum)/prevsum = 2.03215e-05
Probability of good choice for population of 1176=48.8369
prob - prevprob = 9.89866e-06
Convergence test: (sum - prevsum)/prevsum = 2.02692e-05
Probability of good choice for population of 1178=48.8379
prob - prevprob = 9.87345e-06
Convergence test: (sum - prevsum)/prevsum = 2.02172e-05
Probability of good choice for population of 1180=48.8389
prob - prevprob = 9.84834e-06
Convergence test: (sum - prevsum)/prevsum = 2.01654e-05
Probability of good choice for population of 1182=48.8399
prob - prevprob = 9.82335e-06
Convergence test: (sum - prevsum)/prevsum = 2.01138e-05
Probability of good choice for population of 1184=48.8408
prob - prevprob = 9.79846e-06
Convergence test: (sum - prevsum)/prevsum = 2.00624e-05
Probability of good choice for population of 1186=48.8418
prob - prevprob = 9.77367e-06
Convergence test: (sum - prevsum)/prevsum = 2.00113e-05
Probability of good choice for population of 1188=48.8428
prob - prevprob = 9.74899e-06
Convergence test: (sum - prevsum)/prevsum = 1.99603e-05
Probability of good choice for population of 1190=48.8438
prob - prevprob = 9.72442e-06
Convergence test: (sum - prevsum)/prevsum = 1.99096e-05
Probability of good choice for population of 1192=48.8447
prob - prevprob = 9.69994e-06
Convergence test: (sum - prevsum)/prevsum = 1.98591e-05
Probability of good choice for population of 1194=48.8457
prob - prevprob = 9.67557e-06
Convergence test: (sum - prevsum)/prevsum = 1.98088e-05
Probability of good choice for population of 1196=48.8467
prob - prevprob = 9.6513e-06
Convergence test: (sum - prevsum)/prevsum = 1.97587e-05
Probability of good choice for population of 1198=48.8476
prob - prevprob = 9.62713e-06
Convergence test: (sum - prevsum)/prevsum = 1.97089e-05
Probability of good choice for population of 1200=48.8486
prob - prevprob = 9.60306e-06
Convergence test: (sum - prevsum)/prevsum = 1.96592e-05
Probability of good choice for population of 1202=48.8496
prob - prevprob = 9.5791e-06
Convergence test: (sum - prevsum)/prevsum = 1.96098e-05
Probability of good choice for population of 1204=48.8505
prob - prevprob = 9.55523e-06
Convergence test: (sum - prevsum)/prevsum = 1.95605e-05
Probability of good choice for population of 1206=48.8515
```

```
prob - prevprob = 9.53146e-06
Convergence test: (sum - prevsum)/prevsum = 1.95115e-05
Probability of good choice for population of 1208=48.8524
prob - prevprob = 9.50779e-06
Convergence test: (sum - prevsum)/prevsum = 1.94626e-05
Probability of good choice for population of 1210=48.8534
prob - prevprob = 9.48421e-06
Convergence test: (sum - prevsum)/prevsum = 1.9414e-05
Probability of good choice for population of 1212=48.8543
prob - prevprob = 9.46074e-06
Convergence test: (sum - prevsum)/prevsum = 1.93656e-05
Probability of good choice for population of 1214=48.8552
prob - prevprob = 9.43736e-06
Convergence test: (sum - prevsum)/prevsum = 1.93174e-05
Probability of good choice for population of 1216=48.8562
prob - prevprob = 9.41408e-06
Convergence test: (sum - prevsum)/prevsum = 1.92693e-05
Probability of good choice for population of 1218=48.8571
prob - prevprob = 9.39089e-06
Convergence test: (sum - prevsum)/prevsum = 1.92215e-05
Probability of good choice for population of 1220=48.8581
prob - prevprob = 9.3678e-06
Convergence test: (sum - prevsum)/prevsum = 1.91739e-05
Probability of good choice for population of 1222=48.859
prob - prevprob = 9.3448e-06
Convergence test: (sum - prevsum)/prevsum = 1.91264e-05
Probability of good choice for population of 1224=48.8599
prob - prevprob = 9.32189e-06
Convergence test: (sum - prevsum)/prevsum = 1.90792e-05
Probability of good choice for population of 1226=48.8609
prob - prevprob = 9.29908e-06
Convergence test: (sum - prevsum)/prevsum = 1.90321e-05
Probability of good choice for population of 1228=48.8618
prob - prevprob = 9.27637e-06
Convergence test: (sum - prevsum)/prevsum = 1.89853e-05
Probability of good choice for population of 1230=48.8627
prob - prevprob = 9.25374e-06
Convergence test: (sum - prevsum)/prevsum = 1.89386e-05
Probability of good choice for population of 1232=48.8636
prob - prevprob = 9.23121e-06
Convergence test: (sum - prevsum)/prevsum = 1.88921e-05
Probability of good choice for population of 1234=48.8646
prob - prevprob = 9.20877e-06
Convergence test: (sum - prevsum)/prevsum = 1.88458e-05
Probability of good choice for population of 1236=48.8655
prob - prevprob = 9.18641e-06
Convergence test: (sum - prevsum)/prevsum = 1.87997e-05
Probability of good choice for population of 1238=48.8664
```

```
prob - prevprob = 9.16415e-06
Convergence test: (sum - prevsum)/prevsum = 1.87538e-05
Probability of good choice for population of 1240=48.8673
prob - prevprob = 9.14198e-06
Convergence test: (sum - prevsum)/prevsum = 1.87081e-05
Probability of good choice for population of 1242=48.8682
prob - prevprob = 9.1199e-06
Convergence test: (sum - prevsum)/prevsum = 1.86626e-05
Probability of good choice for population of 1244=48.8691
prob - prevprob = 9.09791e-06
Convergence test: (sum - prevsum)/prevsum = 1.86172e-05
Probability of good choice for population of 1246=48.87
prob - prevprob = 9.076e-06
Convergence test: (sum - prevsum)/prevsum = 1.85721e-05
Probability of good choice for population of 1248=48.8709
prob - prevprob = 9.05418e-06
Convergence test: (sum - prevsum)/prevsum = 1.85271e-05
Probability of good choice for population of 1250=48.8718
prob - prevprob = 9.03245e-06
Convergence test: (sum - prevsum)/prevsum = 1.84823e-05
Probability of good choice for population of 1252=48.8727
prob - prevprob = 9.01081e-06
Convergence test: (sum - prevsum)/prevsum = 1.84376e-05
Probability of good choice for population of 1254=48.8736
prob - prevprob = 8.98925e-06
Convergence test: (sum - prevsum)/prevsum = 1.83932e-05
Probability of good choice for population of 1256=48.8745
prob - prevprob = 8.96778e-06
Convergence test: (sum - prevsum)/prevsum = 1.83489e-05
Probability of good choice for population of 1258=48.8754
prob - prevprob = 8.9464e-06
Convergence test: (sum - prevsum)/prevsum = 1.83048e-05
Probability of good choice for population of 1260=48.8763
prob - prevprob = 8.9251e-06
Convergence test: (sum - prevsum)/prevsum = 1.82609e-05
Probability of good choice for population of 1262=48.8772
prob - prevprob = 8.90388e-06
Convergence test: (sum - prevsum)/prevsum = 1.82172e-05
Probability of good choice for population of 1264=48.8781
prob - prevprob = 8.88275e-06
Convergence test: (sum - prevsum)/prevsum = 1.81736e-05
Probability of good choice for population of 1266=48.879
prob - prevprob = 8.8617e-06
Convergence test: (sum - prevsum)/prevsum = 1.81302e-05
Probability of good choice for population of 1268=48.8799
prob - prevprob = 8.84073e-06
Convergence test: (sum - prevsum)/prevsum = 1.8087e-05
Probability of good choice for population of 1270=48.8808
```

```
prob - prevprob = 8.81985e-06
Convergence test: (sum - prevsum)/prevsum = 1.80439e-05
Probability of good choice for population of 1272=48.8816
prob - prevprob = 8.79905e-06
Convergence test: (sum - prevsum)/prevsum = 1.8001e-05
Probability of good choice for population of 1274=48.8825
prob - prevprob = 8.77833e-06
Convergence test: (sum - prevsum)/prevsum = 1.79583e-05
Probability of good choice for population of 1276=48.8834
prob - prevprob = 8.75769e-06
Convergence test: (sum - prevsum)/prevsum = 1.79158e-05
Probability of good choice for population of 1278=48.8843
prob - prevprob = 8.73713e-06
Convergence test: (sum - prevsum)/prevsum = 1.78734e-05
Probability of good choice for population of 1280=48.8851
prob - prevprob = 8.71665e-06
Convergence test: (sum - prevsum)/prevsum = 1.78312e-05
Probability of good choice for population of 1282=48.886
prob - prevprob = 8.69625e-06
Convergence test: (sum - prevsum)/prevsum = 1.77892e-05
Probability of good choice for population of 1284=48.8869
prob - prevprob = 8.67594e-06
Convergence test: (sum - prevsum)/prevsum = 1.77473e-05
Probability of good choice for population of 1286=48.8877
prob - prevprob = 8.6557e-06
Convergence test: (sum - prevsum)/prevsum = 1.77056e-05
Probability of good choice for population of 1288=48.8886
prob - prevprob = 8.63554e-06
Convergence test: (sum - prevsum)/prevsum = 1.7664e-05
Probability of good choice for population of 1290=48.8895
prob - prevprob = 8.61545e-06
Convergence test: (sum - prevsum)/prevsum = 1.76226e-05
Probability of good choice for population of 1292=48.8903
prob - prevprob = 8.59545e-06
Convergence test: (sum - prevsum)/prevsum = 1.75814e-05
Probability of good choice for population of 1294=48.8912
prob - prevprob = 8.57552e-06
Convergence test: (sum - prevsum)/prevsum = 1.75403e-05
Probability of good choice for population of 1296=48.892
prob - prevprob = 8.55567e-06
Convergence test: (sum - prevsum)/prevsum = 1.74994e-05
Probability of good choice for population of 1298=48.8929
prob - prevprob = 8.5359e-06
Convergence test: (sum - prevsum)/prevsum = 1.74587e-05
Probability of good choice for population of 1300=48.8937
prob - prevprob = 8.5162e-06
Convergence test: (sum - prevsum)/prevsum = 1.74181e-05
Probability of good choice for population of 1302=48.8946
```

```
prob - prevprob = 8.49657e-06
Convergence test: (sum - prevsum)/prevsum = 1.73776e-05
Probability of good choice for population of 1304=48.8954
prob - prevprob = 8.47703e-06
Convergence test: (sum - prevsum)/prevsum = 1.73374e-05
Probability of good choice for population of 1306=48.8963
prob - prevprob = 8.45756e-06
Convergence test: (sum - prevsum)/prevsum = 1.72972e-05
Probability of good choice for population of 1308=48.8971
prob - prevprob = 8.43816e-06
Convergence test: (sum - prevsum)/prevsum = 1.72573e-05
Probability of good choice for population of 1310=48.898
prob - prevprob = 8.41883e-06
Convergence test: (sum - prevsum)/prevsum = 1.72174e-05
Probability of good choice for population of 1312=48.8988
prob - prevprob = 8.39958e-06
Convergence test: (sum - prevsum)/prevsum = 1.71778e-05
Probability of good choice for population of 1314=48.8997
prob - prevprob = 8.38041e-06
Convergence test: (sum - prevsum)/prevsum = 1.71383e-05
Probability of good choice for population of 1316=48.9005
prob - prevprob = 8.3613e-06
Convergence test: (sum - prevsum)/prevsum = 1.70989e-05
Probability of good choice for population of 1318=48.9013
prob - prevprob = 8.34227e-06
Convergence test: (sum - prevsum)/prevsum = 1.70597e-05
Probability of good choice for population of 1320=48.9022
prob - prevprob = 8.32331e-06
Convergence test: (sum - prevsum)/prevsum = 1.70206e-05
Probability of good choice for population of 1322=48.903
prob - prevprob = 8.30442e-06
Convergence test: (sum - prevsum)/prevsum = 1.69817e-05
Probability of good choice for population of 1324=48.9038
prob - prevprob = 8.2856e-06
Convergence test: (sum - prevsum)/prevsum = 1.69429e-05
Probability of good choice for population of 1326=48.9046
prob - prevprob = 8.26686e-06
Convergence test: (sum - prevsum)/prevsum = 1.69043e-05
Probability of good choice for population of 1328=48.9055
prob - prevprob = 8.24818e-06
Convergence test: (sum - prevsum)/prevsum = 1.68659e-05
Probability of good choice for population of 1330=48.9063
prob - prevprob = 8.22958e-06
Convergence test: (sum - prevsum)/prevsum = 1.68275e-05
Probability of good choice for population of 1332=48.9071
prob - prevprob = 8.21104e-06
Convergence test: (sum - prevsum)/prevsum = 1.67893e-05
Probability of good choice for population of 1334=48.9079
```

```
prob - prevprob = 8.19258e-06
Convergence test: (sum - prevsum)/prevsum = 1.67513e-05
Probability of good choice for population of 1336=48.9087
prob - prevprob = 8.17418e-06
Convergence test: (sum - prevsum)/prevsum = 1.67134e-05
Probability of good choice for population of 1338=48.9096
prob - prevprob = 8.15585e-06
Convergence test: (sum - prevsum)/prevsum = 1.66757e-05
Probability of good choice for population of 1340=48.9104
prob - prevprob = 8.13759e-06
Convergence test: (sum - prevsum)/prevsum = 1.6638e-05
Probability of good choice for population of 1342=48.9112
prob - prevprob = 8.1194e-06
Convergence test: (sum - prevsum)/prevsum = 1.66006e-05
Probability of good choice for population of 1344=48.912
prob - prevprob = 8.10128e-06
Convergence test: (sum - prevsum)/prevsum = 1.65632e-05
Probability of good choice for population of 1346=48.9128
prob - prevprob = 8.08322e-06
Convergence test: (sum - prevsum)/prevsum = 1.65261e-05
Probability of good choice for population of 1348=48.9136
prob - prevprob = 8.06523e-06
Convergence test: (sum - prevsum)/prevsum = 1.6489e-05
Probability of good choice for population of 1350=48.9144
prob - prevprob = 8.04731e-06
Convergence test: (sum - prevsum)/prevsum = 1.64521e-05
Probability of good choice for population of 1352=48.9152
prob - prevprob = 8.02945e-06
Convergence test: (sum - prevsum)/prevsum = 1.64153e-05
Probability of good choice for population of 1354=48.916
prob - prevprob = 8.01166e-06
Convergence test: (sum - prevsum)/prevsum = 1.63787e-05
Probability of good choice for population of 1356=48.9168
prob - prevprob = 7.99394e-06
Convergence test: (sum - prevsum)/prevsum = 1.63422e-05
Probability of good choice for population of 1358=48.9176
prob - prevprob = 7.97628e-06
Convergence test: (sum - prevsum)/prevsum = 1.63058e-05
Probability of good choice for population of 1360=48.9184
prob - prevprob = 7.95869e-06
Convergence test: (sum - prevsum)/prevsum = 1.62696e-05
Probability of good choice for population of 1362=48.9192
prob - prevprob = 7.94115e-06
Convergence test: (sum - prevsum)/prevsum = 1.62335e-05
Probability of good choice for population of 1364=48.92
prob - prevprob = 7.92369e-06
Convergence test: (sum - prevsum)/prevsum = 1.61975e-05
Probability of good choice for population of 1366=48.9208
```

```
prob - prevprob = 7.90629e-06
Convergence test: (sum - prevsum)/prevsum = 1.61617e-05
Probability of good choice for population of 1368=48.9216
prob - prevprob = 7.88895e-06
Convergence test: (sum - prevsum)/prevsum = 1.6126e-05
Probability of good choice for population of 1370=48.9224
prob - prevprob = 7.87167e-06
Convergence test: (sum - prevsum)/prevsum = 1.60904e-05
Probability of good choice for population of 1372=48.9232
prob - prevprob = 7.85446e-06
Convergence test: (sum - prevsum)/prevsum = 1.60549e-05
Probability of good choice for population of 1374=48.9239
prob - prevprob = 7.83731e-06
Convergence test: (sum - prevsum)/prevsum = 1.60196e-05
Probability of good choice for population of 1376=48.9247
prob - prevprob = 7.82022e-06
Convergence test: (sum - prevsum)/prevsum = 1.59845e-05
Probability of good choice for population of 1378=48.9255
prob - prevprob = 7.8032e-06
Convergence test: (sum - prevsum)/prevsum = 1.59494e-05
Probability of good choice for population of 1380=48.9263
prob - prevprob = 7.78624e-06
Convergence test: (sum - prevsum)/prevsum = 1.59145e-05
Probability of good choice for population of 1382=48.9271
prob - prevprob = 7.76933e-06
Convergence test: (sum - prevsum)/prevsum = 1.58797e-05
Probability of good choice for population of 1384=48.9278
prob - prevprob = 7.75249e-06
Convergence test: (sum - prevsum)/prevsum = 1.5845e-05
Probability of good choice for population of 1386=48.9286
prob - prevprob = 7.73571e-06
Convergence test: (sum - prevsum)/prevsum = 1.58105e-05
Probability of good choice for population of 1388=48.9294
prob - prevprob = 7.71899e-06
Convergence test: (sum - prevsum)/prevsum = 1.5776e-05
Probability of good choice for population of 1390=48.9301
prob - prevprob = 7.70233e-06
Convergence test: (sum - prevsum)/prevsum = 1.57417e-05
Probability of good choice for population of 1392=48.9309
prob - prevprob = 7.68573e-06
Convergence test: (sum - prevsum)/prevsum = 1.57076e-05
Probability of good choice for population of 1394=48.9317
prob - prevprob = 7.66919e-06
Convergence test: (sum - prevsum)/prevsum = 1.56735e-05
Probability of good choice for population of 1396=48.9324
prob - prevprob = 7.65271e-06
Convergence test: (sum - prevsum)/prevsum = 1.56396e-05
Probability of good choice for population of 1398=48.9332
```

```
prob - prevprob = 7.63629e-06
Convergence test: (sum - prevsum)/prevsum = 1.56058e-05
Probability of good choice for population of 1400=48.934
prob - prevprob = 7.61993e-06
Convergence test: (sum - prevsum)/prevsum = 1.55721e-05
Probability of good choice for population of 1402=48.9347
prob - prevprob = 7.60362e-06
Convergence test: (sum - prevsum)/prevsum = 1.55385e-05
Probability of good choice for population of 1404=48.9355
prob - prevprob = 7.58737e-06
Convergence test: (sum - prevsum)/prevsum = 1.55051e-05
Probability of good choice for population of 1406=48.9362
prob - prevprob = 7.57119e-06
Convergence test: (sum - prevsum)/prevsum = 1.54718e-05
Probability of good choice for population of 1408=48.937
prob - prevprob = 7.55505e-06
Convergence test: (sum - prevsum)/prevsum = 1.54386e-05
Probability of good choice for population of 1410=48.9378
prob - prevprob = 7.53898e-06
Convergence test: (sum - prevsum)/prevsum = 1.54055e-05
Probability of good choice for population of 1412=48.9385
prob - prevprob = 7.52296e-06
Convergence test: (sum - prevsum)/prevsum = 1.53725e-05
Probability of good choice for population of 1414=48.9393
prob - prevprob = 7.507e-06
Convergence test: (sum - prevsum)/prevsum = 1.53397e-05
Probability of good choice for population of 1416=48.94
prob - prevprob = 7.4911e-06
Convergence test: (sum - prevsum)/prevsum = 1.53069e-05
Probability of good choice for population of 1418=48.9408
prob - prevprob = 7.47525e-06
Convergence test: (sum - prevsum)/prevsum = 1.52743e-05
Probability of good choice for population of 1420=48.9415
prob - prevprob = 7.45945e-06
Convergence test: (sum - prevsum)/prevsum = 1.52418e-05
Probability of good choice for population of 1422=48.9422
prob - prevprob = 7.44372e-06
Convergence test: (sum - prevsum)/prevsum = 1.52094e-05
Probability of good choice for population of 1424=48.943
prob - prevprob = 7.42803e-06
Convergence test: (sum - prevsum)/prevsum = 1.51771e-05
Probability of good choice for population of 1426=48.9437
prob - prevprob = 7.41241e-06
Convergence test: (sum - prevsum)/prevsum = 1.5145e-05
Probability of good choice for population of 1428=48.9445
prob - prevprob = 7.39684e-06
Convergence test: (sum - prevsum)/prevsum = 1.51129e-05
Probability of good choice for population of 1430=48.9452
```

```
prob - prevprob = 7.38132e-06
Convergence test: (sum - prevsum)/prevsum = 1.5081e-05
Probability of good choice for population of 1432=48.9459
prob - prevprob = 7.36585e-06
Convergence test: (sum - prevsum)/prevsum = 1.50492e-05
Probability of good choice for population of 1434=48.9467
prob - prevprob = 7.35044e-06
Convergence test: (sum - prevsum)/prevsum = 1.50175e-05
Probability of good choice for population of 1436=48.9474
prob - prevprob = 7.33509e-06
Convergence test: (sum - prevsum)/prevsum = 1.49859e-05
Probability of good choice for population of 1438=48.9481
prob - prevprob = 7.31979e-06
Convergence test: (sum - prevsum)/prevsum = 1.49544e-05
Probability of good choice for population of 1440=48.9489
prob - prevprob = 7.30454e-06
Convergence test: (sum - prevsum)/prevsum = 1.4923e-05
Probability of good choice for population of 1442=48.9496
prob - prevprob = 7.28934e-06
Convergence test: (sum - prevsum)/prevsum = 1.48917e-05
Probability of good choice for population of 1444=48.9503
prob - prevprob = 7.2742e-06
Convergence test: (sum - prevsum)/prevsum = 1.48606e-05
Probability of good choice for population of 1446=48.9511
prob - prevprob = 7.2591e-06
Convergence test: (sum - prevsum)/prevsum = 1.48295e-05
Probability of good choice for population of 1448=48.9518
prob - prevprob = 7.24406e-06
Convergence test: (sum - prevsum)/prevsum = 1.47986e-05
Probability of good choice for population of 1450=48.9525
prob - prevprob = 7.22908e-06
Convergence test: (sum - prevsum)/prevsum = 1.47677e-05
Probability of good choice for population of 1452=48.9532
prob - prevprob = 7.21414e-06
Convergence test: (sum - prevsum)/prevsum = 1.4737e-05
Probability of good choice for population of 1454=48.9539
prob - prevprob = 7.19926e-06
Convergence test: (sum - prevsum)/prevsum = 1.47064e-05
Probability of good choice for population of 1456=48.9547
prob - prevprob = 7.18442e-06
Convergence test: (sum - prevsum)/prevsum = 1.46759e-05
Probability of good choice for population of 1458=48.9554
prob - prevprob = 7.16964e-06
Convergence test: (sum - prevsum)/prevsum = 1.46455e-05
Probability of good choice for population of 1460=48.9561
prob - prevprob = 7.15491e-06
Convergence test: (sum - prevsum)/prevsum = 1.46152e-05
Probability of good choice for population of 1462=48.9568
```

```
prob - prevprob = 7.14023e-06
Convergence test: (sum - prevsum)/prevsum = 1.4585e-05
Probability of good choice for population of 1464=48.9575
prob - prevprob = 7.12559e-06
Convergence test: (sum - prevsum)/prevsum = 1.45549e-05
Probability of good choice for population of 1466=48.9582
prob - prevprob = 7.11101e-06
Convergence test: (sum - prevsum)/prevsum = 1.45249e-05
Probability of good choice for population of 1468=48.9589
prob - prevprob = 7.09648e-06
Convergence test: (sum - prevsum)/prevsum = 1.4495e-05
Probability of good choice for population of 1470=48.9597
prob - prevprob = 7.082e-06
Convergence test: (sum - prevsum)/prevsum = 1.44652e-05
Probability of good choice for population of 1472=48.9604
prob - prevprob = 7.06756e-06
Convergence test: (sum - prevsum)/prevsum = 1.44355e-05
Probability of good choice for population of 1474=48.9611
prob - prevprob = 7.05318e-06
Convergence test: (sum - prevsum)/prevsum = 1.44059e-05
Probability of good choice for population of 1476=48.9618
prob - prevprob = 7.03884e-06
Convergence test: (sum - prevsum)/prevsum = 1.43764e-05
Probability of good choice for population of 1478=48.9625
prob - prevprob = 7.02456e-06
Convergence test: (sum - prevsum)/prevsum = 1.4347e-05
Probability of good choice for population of 1480=48.9632
prob - prevprob = 7.01032e-06
Convergence test: (sum - prevsum)/prevsum = 1.43177e-05
Probability of good choice for population of 1482=48.9639
prob - prevprob = 6.99613e-06
Convergence test: (sum - prevsum)/prevsum = 1.42885e-05
Probability of good choice for population of 1484=48.9646
prob - prevprob = 6.98198e-06
Convergence test: (sum - prevsum)/prevsum = 1.42595e-05
Probability of good choice for population of 1486=48.9653
prob - prevprob = 6.96789e-06
Convergence test: (sum - prevsum)/prevsum = 1.42305e-05
Probability of good choice for population of 1488=48.966
prob - prevprob = 6.95384e-06
Convergence test: (sum - prevsum)/prevsum = 1.42016e-05
Probability of good choice for population of 1490=48.9667
prob - prevprob = 6.93984e-06
Convergence test: (sum - prevsum)/prevsum = 1.41728e-05
Probability of good choice for population of 1492=48.9674
prob - prevprob = 6.92588e-06
Convergence test: (sum - prevsum)/prevsum = 1.41441e-05
Probability of good choice for population of 1494=48.968
```

```
prob - prevprob = 6.91198e-06
Convergence test: (sum - prevsum)/prevsum = 1.41155e-05
Probability of good choice for population of 1496=48.9687
prob - prevprob = 6.89812e-06
Convergence test: (sum - prevsum)/prevsum = 1.4087e-05
Probability of good choice for population of 1498=48.9694
prob - prevprob = 6.8843e-06
Convergence test: (sum - prevsum)/prevsum = 1.40586e-05
Probability of good choice for population of 1500=48.9701
prob - prevprob = 6.87053e-06
Convergence test: (sum - prevsum)/prevsum = 1.40303e-05
Probability of good choice for population of 1502=48.9708
prob - prevprob = 6.85681e-06
Convergence test: (sum - prevsum)/prevsum = 1.4002e-05
Probability of good choice for population of 1504=48.9715
prob - prevprob = 6.84313e-06
Convergence test: (sum - prevsum)/prevsum = 1.39739e-05
Probability of good choice for population of 1506=48.9722
prob - prevprob = 6.8295e-06
Convergence test: (sum - prevsum)/prevsum = 1.39459e-05
Probability of good choice for population of 1508=48.9728
prob - prevprob = 6.81591e-06
Convergence test: (sum - prevsum)/prevsum = 1.39179e-05
Probability of good choice for population of 1510=48.9735
prob - prevprob = 6.80237e-06
Convergence test: (sum - prevsum)/prevsum = 1.38901e-05
Probability of good choice for population of 1512=48.9742
prob - prevprob = 6.78888e-06
Convergence test: (sum - prevsum)/prevsum = 1.38623e-05
Probability of good choice for population of 1514=48.9749
prob - prevprob = 6.77542e-06
Convergence test: (sum - prevsum)/prevsum = 1.38347e-05
Probability of good choice for population of 1516=48.9756
prob - prevprob = 6.76202e-06
Convergence test: (sum - prevsum)/prevsum = 1.38071e-05
Probability of good choice for population of 1518=48.9762
prob - prevprob = 6.74865e-06
Convergence test: (sum - prevsum)/prevsum = 1.37796e-05
Probability of good choice for population of 1520=48.9769
prob - prevprob = 6.73533e-06
Convergence test: (sum - prevsum)/prevsum = 1.37522e-05
Probability of good choice for population of 1522=48.9776
prob - prevprob = 6.72206e-06
Convergence test: (sum - prevsum)/prevsum = 1.3725e-05
Probability of good choice for population of 1524=48.9782
prob - prevprob = 6.70882e-06
Convergence test: (sum - prevsum)/prevsum = 1.36977e-05
Probability of good choice for population of 1526=48.9789
```

```
prob - prevprob = 6.69564e-06
Convergence test: (sum - prevsum)/prevsum = 1.36706e-05
Probability of good choice for population of 1528=48.9796
prob - prevprob = 6.68249e-06
Convergence test: (sum - prevsum)/prevsum = 1.36436e-05
Probability of good choice for population of 1530=48.9803
prob - prevprob = 6.66939e-06
Convergence test: (sum - prevsum)/prevsum = 1.36167e-05
Probability of good choice for population of 1532=48.9809
prob - prevprob = 6.65633e-06
Convergence test: (sum - prevsum)/prevsum = 1.35898e-05
Probability of good choice for population of 1534=48.9816
prob - prevprob = 6.64331e-06
Convergence test: (sum - prevsum)/prevsum = 1.35631e-05
Probability of good choice for population of 1536=48.9822
prob - prevprob = 6.63033e-06
Convergence test: (sum - prevsum)/prevsum = 1.35364e-05
Probability of good choice for population of 1538=48.9829
prob - prevprob = 6.6174e-06
Convergence test: (sum - prevsum)/prevsum = 1.35098e-05
Probability of good choice for population of 1540=48.9836
prob - prevprob = 6.60451e-06
Convergence test: (sum - prevsum)/prevsum = 1.34833e-05
Probability of good choice for population of 1542=48.9842
prob - prevprob = 6.59166e-06
Convergence test: (sum - prevsum)/prevsum = 1.34569e-05
Probability of good choice for population of 1544=48.9849
prob - prevprob = 6.57885e-06
Convergence test: (sum - prevsum)/prevsum = 1.34306e-05
Probability of good choice for population of 1546=48.9855
prob - prevprob = 6.56609e-06
Convergence test: (sum - prevsum)/prevsum = 1.34043e-05
Probability of good choice for population of 1548=48.9862
prob - prevprob = 6.55336e-06
Convergence test: (sum - prevsum)/prevsum = 1.33782e-05
Probability of good choice for population of 1550=48.9868
prob - prevprob = 6.54068e-06
Convergence test: (sum - prevsum)/prevsum = 1.33521e-05
Probability of good choice for population of 1552=48.9875
prob - prevprob = 6.52803e-06
Convergence test: (sum - prevsum)/prevsum = 1.33261e-05
Probability of good choice for population of 1554=48.9882
prob - prevprob = 6.51543e-06
Convergence test: (sum - prevsum)/prevsum = 1.33002e-05
Probability of good choice for population of 1556=48.9888
prob - prevprob = 6.50287e-06
Convergence test: (sum - prevsum)/prevsum = 1.32744e-05
Probability of good choice for population of 1558=48.9895
```

```
prob - prevprob = 6.49035e-06
Convergence test: (sum - prevsum)/prevsum = 1.32486e-05
Probability of good choice for population of 1560=48.9901
prob - prevprob = 6.47787e-06
Convergence test: (sum - prevsum)/prevsum = 1.3223e-05
Probability of good choice for population of 1562=48.9907
prob - prevprob = 6.46543e-06
Convergence test: (sum - prevsum)/prevsum = 1.31974e-05
Probability of good choice for population of 1564=48.9914
prob - prevprob = 6.45302e-06
Convergence test: (sum - prevsum)/prevsum = 1.31719e-05
Probability of good choice for population of 1566=48.992
prob - prevprob = 6.44066e-06
Convergence test: (sum - prevsum)/prevsum = 1.31465e-05
Probability of good choice for population of 1568=48.9927
prob - prevprob = 6.42834e-06
Convergence test: (sum - prevsum)/prevsum = 1.31212e-05
Probability of good choice for population of 1570=48.9933
prob - prevprob = 6.41606e-06
Convergence test: (sum - prevsum)/prevsum = 1.30959e-05
Probability of good choice for population of 1572=48.994
prob - prevprob = 6.40381e-06
Convergence test: (sum - prevsum)/prevsum = 1.30708e-05
Probability of good choice for population of 1574=48.9946
prob - prevprob = 6.39161e-06
Convergence test: (sum - prevsum)/prevsum = 1.30457e-05
Probability of good choice for population of 1576=48.9952
prob - prevprob = 6.37944e-06
Convergence test: (sum - prevsum)/prevsum = 1.30207e-05
Probability of good choice for population of 1578=48.9959
prob - prevprob = 6.36731e-06
Convergence test: (sum - prevsum)/prevsum = 1.29958e-05
Probability of good choice for population of 1580=48.9965
prob - prevprob = 6.35522e-06
Convergence test: (sum - prevsum)/prevsum = 1.29709e-05
Probability of good choice for population of 1582=48.9971
prob - prevprob = 6.34317e-06
Convergence test: (sum - prevsum)/prevsum = 1.29462e-05
Probability of good choice for population of 1584=48.9978
prob - prevprob = 6.33116e-06
Convergence test: (sum - prevsum)/prevsum = 1.29215e-05
Probability of good choice for population of 1586=48.9984
prob - prevprob = 6.31918e-06
Convergence test: (sum - prevsum)/prevsum = 1.28969e-05
Probability of good choice for population of 1588=48.999
prob - prevprob = 6.30724e-06
Convergence test: (sum - prevsum)/prevsum = 1.28723e-05
Probability of good choice for population of 1590=48.9997
```

```
prob - prevprob = 6.29534e-06
Convergence test: (sum - prevsum)/prevsum = 1.28479e-05
Probability of good choice for population of 1592=49.0003
prob - prevprob = 6.28348e-06
Convergence test: (sum - prevsum)/prevsum = 1.28235e-05
Probability of good choice for population of 1594=49.0009
prob - prevprob = 6.27165e-06
Convergence test: (sum - prevsum)/prevsum = 1.27992e-05
Probability of good choice for population of 1596=49.0016
prob - prevprob = 6.25986e-06
Convergence test: (sum - prevsum)/prevsum = 1.2775e-05
Probability of good choice for population of 1598=49.0022
prob - prevprob = 6.24811e-06
Convergence test: (sum - prevsum)/prevsum = 1.27508e-05
Probability of good choice for population of 1600=49.0028
prob - prevprob = 6.2364e-06
Convergence test: (sum - prevsum)/prevsum = 1.27268e-05
Probability of good choice for population of 1602=49.0034
prob - prevprob = 6.22472e-06
Convergence test: (sum - prevsum)/prevsum = 1.27028e-05
Probability of good choice for population of 1604=49.004
prob - prevprob = 6.21308e-06
Convergence test: (sum - prevsum)/prevsum = 1.26789e-05
Probability of good choice for population of 1606=49.0047
prob - prevprob = 6.20147e-06
Convergence test: (sum - prevsum)/prevsum = 1.2655e-05
Probability of good choice for population of 1608=49.0053
prob - prevprob = 6.1899e-06
Convergence test: (sum - prevsum)/prevsum = 1.26312e-05
Probability of good choice for population of 1610=49.0059
prob - prevprob = 6.17837e-06
Convergence test: (sum - prevsum)/prevsum = 1.26076e-05
Probability of good choice for population of 1612=49.0065
prob - prevprob = 6.16687e-06
Convergence test: (sum - prevsum)/prevsum = 1.25839e-05
Probability of good choice for population of 1614=49.0071
prob - prevprob = 6.15541e-06
Convergence test: (sum - prevsum)/prevsum = 1.25604e-05
Probability of good choice for population of 1616=49.0077
prob - prevprob = 6.14398e-06
Convergence test: (sum - prevsum)/prevsum = 1.25369e-05
Probability of good choice for population of 1618=49.0084
prob - prevprob = 6.13259e-06
Convergence test: (sum - prevsum)/prevsum = 1.25135e-05
Probability of good choice for population of 1620=49.009
prob - prevprob = 6.12123e-06
Convergence test: (sum - prevsum)/prevsum = 1.24902e-05
Probability of good choice for population of 1622=49.0096
```

```
prob - prevprob = 6.10991e-06
Convergence test: (sum - prevsum)/prevsum = 1.24669e-05
Probability of good choice for population of 1624=49.0102
prob - prevprob = 6.09862e-06
Convergence test: (sum - prevsum)/prevsum = 1.24437e-05
Probability of good choice for population of 1626=49.0108
prob - prevprob = 6.08737e-06
Convergence test: (sum - prevsum)/prevsum = 1.24206e-05
Probability of good choice for population of 1628=49.0114
prob - prevprob = 6.07615e-06
Convergence test: (sum - prevsum)/prevsum = 1.23976e-05
Probability of good choice for population of 1630=49.012
prob - prevprob = 6.06497e-06
Convergence test: (sum - prevsum)/prevsum = 1.23746e-05
Probability of good choice for population of 1632=49.0126
prob - prevprob = 6.05382e-06
Convergence test: (sum - prevsum)/prevsum = 1.23517e-05
Probability of good choice for population of 1634=49.0132
prob - prevprob = 6.04271e-06
Convergence test: (sum - prevsum)/prevsum = 1.23289e-05
Probability of good choice for population of 1636=49.0138
prob - prevprob = 6.03162e-06
Convergence test: (sum - prevsum)/prevsum = 1.23061e-05
Probability of good choice for population of 1638=49.0144
prob - prevprob = 6.02058e-06
Convergence test: (sum - prevsum)/prevsum = 1.22834e-05
Probability of good choice for population of 1640=49.015
prob - prevprob = 6.00956e-06
Convergence test: (sum - prevsum)/prevsum = 1.22608e-05
Probability of good choice for population of 1642=49.0156
prob - prevprob = 5.99858e-06
Convergence test: (sum - prevsum)/prevsum = 1.22383e-05
Probability of good choice for population of 1644=49.0162
prob - prevprob = 5.98764e-06
Convergence test: (sum - prevsum)/prevsum = 1.22158e-05
Probability of good choice for population of 1646=49.0168
prob - prevprob = 5.97673e-06
Convergence test: (sum - prevsum)/prevsum = 1.21934e-05
Probability of good choice for population of 1648=49.0174
prob - prevprob = 5.96585e-06
Convergence test: (sum - prevsum)/prevsum = 1.2171e-05
Probability of good choice for population of 1650=49.018
prob - prevprob = 5.955e-06
Convergence test: (sum - prevsum)/prevsum = 1.21487e-05
Probability of good choice for population of 1652=49.0186
prob - prevprob = 5.94418e-06
Convergence test: (sum - prevsum)/prevsum = 1.21265e-05
Probability of good choice for population of 1654=49.0192
```

```
prob - prevprob = 5.9334e-06
Convergence test: (sum - prevsum)/prevsum = 1.21044e-05
Probability of good choice for population of 1656=49.0198
prob - prevprob = 5.92265e-06
Convergence test: (sum - prevsum)/prevsum = 1.20823e-05
Probability of good choice for population of 1658=49.0204
prob - prevprob = 5.91194e-06
Convergence test: (sum - prevsum)/prevsum = 1.20603e-05
Probability of good choice for population of 1660=49.021
prob - prevprob = 5.90125e-06
Convergence test: (sum - prevsum)/prevsum = 1.20384e-05
Probability of good choice for population of 1662=49.0216
prob - prevprob = 5.8906e-06
Convergence test: (sum - prevsum)/prevsum = 1.20165e-05
Probability of good choice for population of 1664=49.0222
prob - prevprob = 5.87998e-06
Convergence test: (sum - prevsum)/prevsum = 1.19947e-05
Probability of good choice for population of 1666=49.0227
prob - prevprob = 5.86939e-06
Convergence test: (sum - prevsum)/prevsum = 1.19729e-05
Probability of good choice for population of 1668=49.0233
prob - prevprob = 5.85884e-06
Convergence test: (sum - prevsum)/prevsum = 1.19513e-05
Probability of good choice for population of 1670=49.0239
prob - prevprob = 5.84831e-06
Convergence test: (sum - prevsum)/prevsum = 1.19296e-05
Probability of good choice for population of 1672=49.0245
prob - prevprob = 5.83782e-06
Convergence test: (sum - prevsum)/prevsum = 1.19081e-05
Probability of good choice for population of 1674=49.0251
prob - prevprob = 5.82736e-06
Convergence test: (sum - prevsum)/prevsum = 1.18866e-05
Probability of good choice for population of 1676=49.0257
prob - prevprob = 5.81693e-06
Convergence test: (sum - prevsum)/prevsum = 1.18652e-05
Probability of good choice for population of 1678=49.0262
prob - prevprob = 5.80653e-06
Convergence test: (sum - prevsum)/prevsum = 1.18438e-05
Probability of good choice for population of 1680=49.0268
prob - prevprob = 5.79616e-06
Convergence test: (sum - prevsum)/prevsum = 1.18226e-05
Probability of good choice for population of 1682=49.0274
prob - prevprob = 5.78582e-06
Convergence test: (sum - prevsum)/prevsum = 1.18013e-05
Probability of good choice for population of 1684=49.028
prob - prevprob = 5.77551e-06
Convergence test: (sum - prevsum)/prevsum = 1.17802e-05
Probability of good choice for population of 1686=49.0286
```

```
prob - prevprob = 5.76523e-06
Convergence test: (sum - prevsum)/prevsum = 1.17591e-05
Probability of good choice for population of 1688=49.0291
prob - prevprob = 5.75499e-06
Convergence test: (sum - prevsum)/prevsum = 1.1738e-05
Probability of good choice for population of 1690=49.0297
prob - prevprob = 5.74477e-06
Convergence test: (sum - prevsum)/prevsum = 1.17171e-05
Probability of good choice for population of 1692=49.0303
prob - prevprob = 5.73459e-06
Convergence test: (sum - prevsum)/prevsum = 1.16961e-05
Probability of good choice for population of 1694=49.0309
prob - prevprob = 5.72443e-06
Convergence test: (sum - prevsum)/prevsum = 1.16753e-05
Probability of good choice for population of 1696=49.0314
prob - prevprob = 5.71431e-06
Convergence test: (sum - prevsum)/prevsum = 1.16545e-05
Probability of good choice for population of 1698=49.032
prob - prevprob = 5.70421e-06
Convergence test: (sum - prevsum)/prevsum = 1.16338e-05
Probability of good choice for population of 1700=49.0326
prob - prevprob = 5.69414e-06
Convergence test: (sum - prevsum)/prevsum = 1.16131e-05
Probability of good choice for population of 1702=49.0331
prob - prevprob = 5.68411e-06
Convergence test: (sum - prevsum)/prevsum = 1.15925e-05
Probability of good choice for population of 1704=49.0337
prob - prevprob = 5.6741e-06
Convergence test: (sum - prevsum)/prevsum = 1.1572e-05
Probability of good choice for population of 1706=49.0343
prob - prevprob = 5.66412e-06
Convergence test: (sum - prevsum)/prevsum = 1.15515e-05
Probability of good choice for population of 1708=49.0348
prob - prevprob = 5.65417e-06
Convergence test: (sum - prevsum)/prevsum = 1.15311e-05
Probability of good choice for population of 1710=49.0354
prob - prevprob = 5.64425e-06
Convergence test: (sum - prevsum)/prevsum = 1.15107e-05
Probability of good choice for population of 1712=49.036
prob - prevprob = 5.63436e-06
Convergence test: (sum - prevsum)/prevsum = 1.14904e-05
Probability of good choice for population of 1714=49.0365
prob - prevprob = 5.6245e-06
Convergence test: (sum - prevsum)/prevsum = 1.14702e-05
Probability of good choice for population of 1716=49.0371
prob - prevprob = 5.61467e-06
Convergence test: (sum - prevsum)/prevsum = 1.145e-05
Probability of good choice for population of 1718=49.0376
```

```
prob - prevprob = 5.60486e-06
Convergence test: (sum - prevsum)/prevsum = 1.14298e-05
Probability of good choice for population of 1720=49.0382
prob - prevprob = 5.59509e-06
Convergence test: (sum - prevsum)/prevsum = 1.14098e-05
Probability of good choice for population of 1722=49.0388
prob - prevprob = 5.58534e-06
Convergence test: (sum - prevsum)/prevsum = 1.13898e-05
Probability of good choice for population of 1724=49.0393
prob - prevprob = 5.57562e-06
Convergence test: (sum - prevsum)/prevsum = 1.13698e-05
Probability of good choice for population of 1726=49.0399
prob - prevprob = 5.56593e-06
Convergence test: (sum - prevsum)/prevsum = 1.13499e-05
Probability of good choice for population of 1728=49.0404
prob - prevprob = 5.55627e-06
Convergence test: (sum - prevsum)/prevsum = 1.13301e-05
Probability of good choice for population of 1730=49.041
prob - prevprob = 5.54663e-06
Convergence test: (sum - prevsum)/prevsum = 1.13103e-05
Probability of good choice for population of 1732=49.0415
prob - prevprob = 5.53702e-06
Convergence test: (sum - prevsum)/prevsum = 1.12906e-05
Probability of good choice for population of 1734=49.0421
prob - prevprob = 5.52744e-06
Convergence test: (sum - prevsum)/prevsum = 1.12709e-05
Probability of good choice for population of 1736=49.0426
prob - prevprob = 5.51789e-06
Convergence test: (sum - prevsum)/prevsum = 1.12513e-05
Probability of good choice for population of 1738=49.0432
prob - prevprob = 5.50837e-06
Convergence test: (sum - prevsum)/prevsum = 1.12318e-05
Probability of good choice for population of 1740=49.0437
prob - prevprob = 5.49887e-06
Convergence test: (sum - prevsum)/prevsum = 1.12123e-05
Probability of good choice for population of 1742=49.0443
prob - prevprob = 5.4894e-06
Convergence test: (sum - prevsum)/prevsum = 1.11929e-05
Probability of good choice for population of 1744=49.0448
prob - prevprob = 5.47996e-06
Convergence test: (sum - prevsum)/prevsum = 1.11735e-05
Probability of good choice for population of 1746=49.0454
prob - prevprob = 5.47054e-06
Convergence test: (sum - prevsum)/prevsum = 1.11542e-05
Probability of good choice for population of 1748=49.0459
prob - prevprob = 5.46115e-06
Convergence test: (sum - prevsum)/prevsum = 1.11349e-05
Probability of good choice for population of 1750=49.0465
```

```
prob - prevprob = 5.45179e-06
Convergence test: (sum - prevsum)/prevsum = 1.11157e-05
Probability of good choice for population of 1752=49.047
prob - prevprob = 5.44246e-06
Convergence test: (sum - prevsum)/prevsum = 1.10965e-05
Probability of good choice for population of 1754=49.0476
prob - prevprob = 5.43315e-06
Convergence test: (sum - prevsum)/prevsum = 1.10774e-05
Probability of good choice for population of 1756=inf
prob - prevprob = inf
Convergence test: (sum - prevsum)/prevsum = inf
Probability of good choice for population of 1758=inf
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1760=inf
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1762=inf
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1764=inf
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1766=inf
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1768=inf
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1770=inf
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1772=inf
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1774=inf
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1776=inf
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1778=inf
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1780=inf
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1782=inf
```

```
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1784=inf
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1786=inf
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1788=inf
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1790=inf
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1792=inf
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1794=inf
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1796=inf
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1798=inf
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1800=inf
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1802=inf
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1804=inf
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1806=inf
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1808=inf
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1810=inf
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1812=inf
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1814=inf
```

```
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1816=inf
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1818=inf
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1820=inf
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1822=inf
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1824=inf
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1826=inf
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1828=inf
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1830=inf
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1832=inf
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1834=inf
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1836=inf
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1838=inf
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1840=inf
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1842=inf
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1844=inf
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1846=inf
```

```
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1848=inf
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1850=inf
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1852=inf
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1854=inf
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1856=inf
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1858=inf
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1860=inf
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1862=inf
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1864=inf
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1866=inf
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1868=inf
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1870=inf
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1872=inf
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1874=inf
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1876=inf
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1878=inf
```

```
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1880=inf
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1882=inf
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1884=inf
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1886=inf
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1888=inf
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1890=inf
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1892=inf
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1894=inf
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1896=inf
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1898=inf
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1900=inf
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1902=inf
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1904=inf
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1906=inf
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1908=inf
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1910=inf
```

```
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1912=inf
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1914=inf
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1916=inf
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1918=inf
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1920=inf
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1922=inf
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1924=inf
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1926=inf
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1928=inf
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1930=inf
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1932=inf
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1934=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1936=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1938=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1940=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1942=-nan
```

```
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1944=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1946=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1948=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1950=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1952=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1954=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1956=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1958=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1960=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1962=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1964=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1966=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1968=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1970=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1972=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1974=-nan
```

```
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1976=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1978=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1980=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1982=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1984=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1986=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1988=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1990=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1992=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1994=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1996=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 1998=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2000=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2002=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2004=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2006=-nan
```

```
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2008=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2010=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2012=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2014=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2016=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2018=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2020=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2022=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2024=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2026=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2028=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2030=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2032=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2034=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2036=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2038=-nan
```

```
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2040=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2042=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2044=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2046=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2048=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2050=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2052=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2054=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2056=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2058=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2060=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2062=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2064=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2066=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2068=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2070=-nan
```

```
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2072=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2074=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2076=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2078=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2080=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2082=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2084=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2086=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2088=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2090=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2092=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2094=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2096=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2098=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2100=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2102=-nan
```

```
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2104=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2106=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2108=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2110=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2112=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2114=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2116=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2118=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2120=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2122=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2124=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2126=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2128=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2130=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2132=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2134=-nan
```

```
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2136=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2138=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2140=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2142=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2144=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2146=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2148=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2150=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2152=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2154=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2156=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2158=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2160=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2162=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2164=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2166=-nan
```

```
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2168=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2170=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2172=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2174=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2176=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2178=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2180=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2182=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2184=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2186=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2188=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2190=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2192=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2194=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2196=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2198=-nan
```

```
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2200=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2202=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2204=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2206=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2208=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2210=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2212=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2214=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2216=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2218=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2220=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2222=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2224=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2226=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2228=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2230=-nan
```

```
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2232=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2234=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2236=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2238=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2240=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2242=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2244=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2246=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2248=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2250=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2252=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2254=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2256=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2258=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2260=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2262=-nan
```

```
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2264=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2266=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2268=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2270=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2272=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2274=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2276=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2278=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2280=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2282=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2284=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2286=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2288=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2290=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2292=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2294=-nan
```

```
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2296=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2298=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2300=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2302=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2304=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2306=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2308=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2310=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2312=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2314=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2316=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2318=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2320=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2322=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2324=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2326=-nan
```

```
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2328=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2330=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2332=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2334=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2336=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2338=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2340=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2342=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2344=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2346=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2348=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2350=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2352=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2354=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2356=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2358=-nan
```

```
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2360=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2362=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2364=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2366=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2368=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2370=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2372=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2374=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2376=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2378=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2380=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2382=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2384=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2386=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2388=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2390=-nan
```

```
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2392=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2394=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2396=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2398=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2400=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2402=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2404=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2406=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2408=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2410=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2412=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2414=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2416=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2418=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2420=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2422=-nan
```

```
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2424=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2426=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2428=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2430=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2432=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2434=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2436=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2438=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2440=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2442=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2444=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
Probability of good choice for population of 2446=-nan
prob - prevprob = -nan
Convergence test: (sum - prevsum)/prevsum = -nan
```

6 Acknowledgement

I dedicate this article to God.

7 Bibliography

References

- [1] Few Algorithms for ascertaining merit of a document - <http://arxiv.org/pdf/1006.4458.pdf>
- [2] TAC 2010 Update summarization by Interview Algorithm (http://www.nist.gov/tac/publications/2010/appendices/Summarization/guided/CMI_IT.pdf)
- [3] Complexity theory, Sanjeev arora and Boaz Barak
- [4] Majority function in non-uniform NC1, Mix Barrington
- [5] Sorting networks for majority function, Ajtai ,Komlos, Szemerredi
- [6] Valiant's non-constructive majority function circuit
- [7] <http://sourceforge.net/p/asfer/code/HEAD/tree/cpp-src/miscellaneous/pgood.cpp>
- [8] Binomial coefficients, Pascal triangle - (http://en.wikipedia.org/wiki/Binomial_coefficient)
- [9] Hypergeometric functions for binomial coefficient summations - <http://www.math.upenn.edu/wilf/AeqB.pdf>.
- [10] Majority voting related publication drafts in <https://sites.google.com/site/kuja27/>
- [11] Output logs - http://sourceforge.net/p/asfer/code/HEAD/tree/cpp-src/miscellaneous/pgood_output_excerpts.with_series_convergence_test.26November2013

Circuit For Computing Error Probability of Majority Voting

Ka.Shrinivaasan (ka.shrinivaasan@gmail.com)

March 8, 2013

Abstract

In this article a DC uniform circuit for computation of error probability in majority voting is constructed and thus shown to be in PH. This error probability in majority voting is nothing but the RHS of the P(good) equation published earlier as mentioned in bibliography.

1 Derivation of P(good) expression

In a distributed systems (e.g cloud computing environments) with $2n$ nodes, the leader is elected by a majority voting if it obtains $n + 1$ or more votes. We would like to bound the probability that such an outcome of a majority vote is good. This is precisely the probability that atleast $n + 1$ nodes have made a good decision. Assumption here is that a good majority decision by majority of the nodes with high probability also results in a good outcome in the majority voting (rather it is trivial and is accepted axiomatically without proof). Thus by probability union bound,

$\Pr(\text{leader elected by majority voting is good}) =$
 $\Pr(\text{atleast } n+1 \text{ nodes have made a good decision})$

$= \Pr(n+1 \text{ nodes have made good decision}) +$
 $\Pr(n+2 \text{ nodes have made good decision}) +$
 $\dots +$
 $\Pr(2n \text{ nodes have made good decision})$
 $- (0 \text{ for intersection probability})$

If $\Pr(\text{good decision by a node}) = 0.5$
(assuming a uniform distribution and thus good and bad decisions are equally probable) then LHS of the P(Good) equation is 0.5 and the RHS is obtained through the series distribution as,

$$\begin{aligned} &= 2nC(n+1) (0.5)^{n+1} (0.5)^{n-1} + \dots + 2nC2n (0.5)^{2n} \\ &= (0.25)^n [2nC(n+1) + 2nC(n+2) + \dots + 2nC(2n)] \\ &= (2n)!/(4^n) [1/(n+1)!(n-1)! + 1/(n-2)!(n+2)! + \dots + 1/(2n)!] \end{aligned}$$

This series tends to 0.5 which can be confirmed by simple convergence test. The fraction $[n^{th} \text{ term} - (n-1)^{th} \text{ term}]/n$ tends to zero as n tends to infinity. Output of a computer program which computes the above is in appendix. For odd number of nodes, probability can be analogously derived by rounding off to nearest integer to get past the halfway point. For odd number of the voter population, say m , the halfway point is $\text{ceiling}(m/2)$. Let $x = \text{ceiling}(m/2)$. Thus above series becomes,

$$\begin{aligned}
&= mC(x+1) (0.5)^m + mC(x+2) (0.5)^m + \dots + mCm (0.5)^m \\
&= (0.5)^m [mC(x) + mC(x+1) + \dots + mCm] \\
&= m!/(2^m) [1/x!(m-x)! + \dots + 1/m!]
\end{aligned}$$

Assumption here is that a good majority decision with high probability also results in a good outcome in the majority voting (rather it is trivial and is stated without proof).

2 A simple majority voting example with 5 voters

Following are the 32 possible voting patterns in terms of nature of individual decision for 5 voters to elect a leader (0 means voter has made bad decision and 1 means voter has made good decision):

00000, 10000
00001, 10001
00010, 10010
00011, 10011
00100, 10100
00101, 10101
00110, 10110
00111, 10111
01000, 11000
01001, 11001
01010, 11010
01011, 11011
01100, 11100
01101, 11101
01110, 11110
01111, 11111

From the above the probability that atleast $ceiling(5/2)$ (or more than or equal to 3 in above example) voters have made a good decision, can be computed easily by glancing. Out of 32 patterns, 16 have 3 or more 1s (good decisions). Thus the probability that elected leader is good is 16/32. This can be derived from above series also as,

$$\begin{aligned}
&= 1/32 [5C3 + 5C4 + 5C5] \\
&= 16/32 = 0.5
\end{aligned}$$

3 Circuit for computing P(Good) error probability from voter decision patterns

Complexity of computing above series in RHS of P(Good) equation is exponential in n because of computation of factorials (can be approximated by Stirling's formula). P(Good) series implies that any leader election algorithm that involves majority voting (under zero bias space where $Pr[decision = 1]Pr[decision = 0] = 0$) is no better than a (pseudo)random choice. Translating P(good) series into circuit requires computation of majority function on 2^{2n} possible inputs corresponding to all possible voting patters by the $2n$ voters.

Following is the algorithm for drawing the above circuit:

1. Have 2^{2n} majority circuits. Each of these majority circuit takes as input one voting voting decision bit pattern each with each bit as a decision(good or bad) input for corresponding voter(as explained in example above). Majority function can be computed in polynomial size by sorting networks (Ajtai et al) or through non-uniform NC1 circuit (Barrington) or Valiant's non-constructive majority circuit of size $n^5 \cdot 3$. Thus each majority circuit computes the majority of the voting pattern and outputs 1 or 0 depending on which of the bits are in majority (1 means majority decision is good and 0 means bad for that voting pattern). Thus 2^{2n} majority circuits compute the majority of each of the 2^{2n} voting bit patterns and output 0 or 1.
2. An addition circuit then adds up the 2^{2n} bits output by all of the above majority circuits. This addition circuit has exponential fan-in and thus exponential in n and thus NC1 cannot compute this addition which requires bounded fanin ,logdepth and polysize circuit. Exponential sized circuits are in DC-uniform family characterized by Polynomial Hierarchy(PH) which is defined as circuit having AND,OR and NOT gates and size 2^{2n} with unbounded fanin.
3. Output of the above addition circuit is the numerator of the P(Good) fraction. Thus a division circuit is needed to divide this numerator by denominator which is 2^{2n} . Division can be performed in TC0. Thus summing up we have a 3 step circuit ($NC1 + DC - uniform + TC0$). Subsuming NC1 and TC0 in DC-uniform gives a DC-uniform exponential sized circuit to compute the P(Good) RHS probability.

4 Conclusion

Thus a series expression for the error probability in majority voting has been derived and a DC-uniform circuit has been constructed for it.

5 Appendix for P(Good) computation with even number voter population output by a computer program (probability is in percentage)

```

Probability of good choice for population of 0=0
prob - prevprob = 0
sumdiff - prevsumdiff = 0
Probability of good choice for population of 2=25
prob - prevprob = 0.25
sumdiff - prevsumdiff = 0.25
Probability of good choice for population of 4=31.25
prob - prevprob = 0.0625
sumdiff - prevsumdiff = -0.1875
Probability of good choice for population of 6=34.375
prob - prevprob = 0.03125
sumdiff - prevsumdiff = -0.03125
Probability of good choice for population of 8=36.3281
prob - prevprob = 0.0195312
sumdiff - prevsumdiff = -0.0117188
Probability of good choice for population of 10=37.6953

```

```
prob - prevprob = 0.0136719
sumdiff - prevsumdiff = -0.00585938
Probability of good choice for population of 12=38.7207
prob - prevprob = 0.0102539
sumdiff - prevsumdiff = -0.00341797
Probability of good choice for population of 14=39.5264
prob - prevprob = 0.00805664
sumdiff - prevsumdiff = -0.00219727
Probability of good choice for population of 16=40.181
prob - prevprob = 0.00654602
sumdiff - prevsumdiff = -0.00151062
Probability of good choice for population of 18=40.7265
prob - prevprob = 0.00545502
sumdiff - prevsumdiff = -0.001091
Probability of good choice for population of 20=41.1901
prob - prevprob = 0.00463676
sumdiff - prevsumdiff = -0.000818253
Probability of good choice for population of 22=41.5906
prob - prevprob = 0.00400448
sumdiff - prevsumdiff = -0.000632286
Probability of good choice for population of 24=41.941
prob - prevprob = 0.00350392
sumdiff - prevsumdiff = -0.00050056
Probability of good choice for population of 26=42.2509
prob - prevprob = 0.00309962
sumdiff - prevsumdiff = -0.000404298
Probability of good choice for population of 28=42.5277
prob - prevprob = 0.00276752
sumdiff - prevsumdiff = -0.000332102
Probability of good choice for population of 30=42.7768
prob - prevprob = 0.00249077
sumdiff - prevsumdiff = -0.000276752
Probability of good choice for population of 32=43.0025
prob - prevprob = 0.00225726
sumdiff - prevsumdiff = -0.000233509
Probability of good choice for population of 34=43.2083
prob - prevprob = 0.00205809
sumdiff - prevsumdiff = -0.00019917
Probability of good choice for population of 36=43.397
prob - prevprob = 0.00188658
sumdiff - prevsumdiff = -0.000171507
Probability of good choice for population of 38=43.5707
prob - prevprob = 0.00173764
sumdiff - prevsumdiff = -0.000148941
Probability of good choice for population of 40=43.7315
prob - prevprob = 0.00160732
sumdiff - prevsumdiff = -0.000130323
Probability of good choice for population of 42=43.8807
```

```
prob - prevprob = 0.00149251
sumdiff - prevsumdiff = -0.000114808
Probability of good choice for population of 44=44.0198
prob - prevprob = 0.00139075
sumdiff - prevsumdiff = -0.000101762
Probability of good choice for population of 46=44.1498
prob - prevprob = 0.00130005
sumdiff - prevsumdiff = -9.07008e-005
Probability of good choice for population of 48=44.2717
prob - prevprob = 0.00121879
sumdiff - prevsumdiff = -8.12528e-005
Probability of good choice for population of 50=44.3862
prob - prevprob = 0.00114567
sumdiff - prevsumdiff = -7.31276e-005
Probability of good choice for population of 52=44.4942
prob - prevprob = 0.00107957
sumdiff - prevsumdiff = -6.60961e-005
Probability of good choice for population of 54=44.5962
prob - prevprob = 0.00101959
sumdiff - prevsumdiff = -5.99761e-005
Probability of good choice for population of 56=44.6927
prob - prevprob = 0.000964972
sumdiff - prevsumdiff = -5.4621e-005
Probability of good choice for population of 58=44.7842
prob - prevprob = 0.00091506
sumdiff - prevsumdiff = -4.99123e-005
Probability of good choice for population of 60=44.8711
prob - prevprob = 0.000869307
sumdiff - prevsumdiff = -4.5753e-005
Probability of good choice for population of 62=44.9538
prob - prevprob = 0.000827243
sumdiff - prevsumdiff = -4.20632e-005
Probability of good choice for population of 64=45.0327
prob - prevprob = 0.000788466
sumdiff - prevsumdiff = -3.8777e-005
Probability of good choice for population of 66=45.1079
prob - prevprob = 0.000752627
sumdiff - prevsumdiff = -3.58394e-005
Probability of good choice for population of 68=45.1799
prob - prevprob = 0.000719423
sumdiff - prevsumdiff = -3.32041e-005
Probability of good choice for population of 70=45.2487
prob - prevprob = 0.00068859
sumdiff - prevsumdiff = -3.08324e-005
Probability of good choice for population of 72=45.3147
prob - prevprob = 0.000659899
sumdiff - prevsumdiff = -2.86913e-005
Probability of good choice for population of 74=45.378
```

```
prob - prevprob = 0.000633146
sumdiff - prevsumdiff = -2.67527e-005
Probability of good choice for population of 76=45.4388
prob - prevprob = 0.000608154
sumdiff - prevsumdiff = -2.49926e-005
Probability of good choice for population of 78=45.4973
prob - prevprob = 0.000584763
sumdiff - prevsumdiff = -2.33905e-005
Probability of good choice for population of 80=45.5536
prob - prevprob = 0.000562835
sumdiff - prevsumdiff = -2.19286e-005
Probability of good choice for population of 82=45.6078
prob - prevprob = 0.000542243
sumdiff - prevsumdiff = -2.05915e-005
Probability of good choice for population of 84=45.6601
prob - prevprob = 0.000522877
sumdiff - prevsumdiff = -1.93658e-005
Probability of good choice for population of 86=45.7106
prob - prevprob = 0.000504637
sumdiff - prevsumdiff = -1.82399e-005
Probability of good choice for population of 88=45.7593
prob - prevprob = 0.000487434
sumdiff - prevsumdiff = -1.72035e-005
Probability of good choice for population of 90=45.8064
prob - prevprob = 0.000471186
sumdiff - prevsumdiff = -1.62478e-005
Probability of good choice for population of 92=45.852
prob - prevprob = 0.000455821
sumdiff - prevsumdiff = -1.53648e-005
Probability of good choice for population of 94=45.8962
prob - prevprob = 0.000441274
sumdiff - prevsumdiff = -1.45475e-005
Probability of good choice for population of 96=45.9389
prob - prevprob = 0.000427484
sumdiff - prevsumdiff = -1.37898e-005
Probability of good choice for population of 98=45.9803
prob - prevprob = 0.000414398
sumdiff - prevsumdiff = -1.30862e-005
Probability of good choice for population of 100=46.0205
prob - prevprob = 0.000401966
sumdiff - prevsumdiff = -1.24319e-005
Probability of good choice for population of 102=46.0596
prob - prevprob = 0.000390143
sumdiff - prevsumdiff = -1.18225e-005
Probability of good choice for population of 104=46.0974
prob - prevprob = 0.000378889
sumdiff - prevsumdiff = -1.12541e-005
Probability of good choice for population of 106=46.1343
```

```
prob - prevprob = 0.000368166
sumdiff - prevsumdiff = -1.07233e-005
Probability of good choice for population of 108=46.1701
prob - prevprob = 0.000357939
sumdiff - prevsumdiff = -1.02268e-005
Probability of good choice for population of 110=46.2049
prob - prevprob = 0.000348177
sumdiff - prevsumdiff = -9.76197e-006
Probability of good choice for population of 112=46.2388
prob - prevprob = 0.000338851
sumdiff - prevsumdiff = -9.32617e-006
Probability of good choice for population of 114=46.2717
prob - prevprob = 0.000329934
sumdiff - prevsumdiff = -8.91713e-006
Probability of good choice for population of 116=46.3039
prob - prevprob = 0.000321401
sumdiff - prevsumdiff = -8.53277e-006
Probability of good choice for population of 118=46.3352
prob - prevprob = 0.00031323
sumdiff - prevsumdiff = -8.17121e-006
Probability of good choice for population of 120=46.3658
prob - prevprob = 0.000305399
sumdiff - prevsumdiff = -7.83075e-006
Probability of good choice for population of 122=46.3955
prob - prevprob = 0.000297889
sumdiff - prevsumdiff = -7.50981e-006
Probability of good choice for population of 124=46.4246
prob - prevprob = 0.000290682
sumdiff - prevsumdiff = -7.207e-006
Probability of good choice for population of 126=46.453
prob - prevprob = 0.000283761
sumdiff - prevsumdiff = -6.92101e-006
Probability of good choice for population of 128=46.4807
prob - prevprob = 0.000277111
sumdiff - prevsumdiff = -6.65065e-006
Probability of good choice for population of 130=46.5078
prob - prevprob = 0.000270716
sumdiff - prevsumdiff = -6.39486e-006
Probability of good choice for population of 132=46.5342
prob - prevprob = 0.000264563
sumdiff - prevsumdiff = -6.15263e-006
Probability of good choice for population of 134=46.5601
prob - prevprob = 0.00025864
sumdiff - prevsumdiff = -5.92305e-006
Probability of good choice for population of 136=46.5854
prob - prevprob = 0.000252935
sumdiff - prevsumdiff = -5.7053e-006
Probability of good choice for population of 138=46.6101
```

```
prob - prevprob = 0.000247436
sumdiff - prevsumdiff = -5.49858e-006
Probability of good choice for population of 140=46.6343
prob - prevprob = 0.000242134
sumdiff - prevsumdiff = -5.3022e-006
Probability of good choice for population of 142=46.658
prob - prevprob = 0.000237018
sumdiff - prevsumdiff = -5.11551e-006
Probability of good choice for population of 144=46.6812
prob - prevprob = 0.000232081
sumdiff - prevsumdiff = -4.93788e-006
Probability of good choice for population of 146=46.704
prob - prevprob = 0.000227312
sumdiff - prevsumdiff = -4.76878e-006
Probability of good choice for population of 148=46.7262
prob - prevprob = 0.000222704
sumdiff - prevsumdiff = -4.60767e-006
Probability of good choice for population of 150=46.7481
prob - prevprob = 0.00021825
sumdiff - prevsumdiff = -4.45408e-006
Probability of good choice for population of 152=46.7695
prob - prevprob = 0.000213942
sumdiff - prevsumdiff = -4.30757e-006
Probability of good choice for population of 154=46.7904
prob - prevprob = 0.000209775
sumdiff - prevsumdiff = -4.16771e-006
Probability of good choice for population of 156=46.811
prob - prevprob = 0.000205741
sumdiff - prevsumdiff = -4.03413e-006
Probability of good choice for population of 158=46.8312
prob - prevprob = 0.000201834
sumdiff - prevsumdiff = -3.90647e-006
Probability of good choice for population of 160=46.851
prob - prevprob = 0.00019805
sumdiff - prevsumdiff = -3.78439e-006
Probability of good choice for population of 162=46.8704
prob - prevprob = 0.000194382
sumdiff - prevsumdiff = -3.66759e-006
Probability of good choice for population of 164=46.8895
prob - prevprob = 0.000190826
sumdiff - prevsumdiff = -3.55577e-006
Probability of good choice for population of 166=46.9083
prob - prevprob = 0.000187378
sumdiff - prevsumdiff = -3.44867e-006
Probability of good choice for population of 168=46.9267
prob - prevprob = 0.000184032
sumdiff - prevsumdiff = -3.34603e-006
Probability of good choice for population of 170=46.9447
```

```
prob - prevprob = 0.000180784
sumdiff - prevsumdiff = -3.24762e-006
Probability of good choice for population of 172=46.9625
prob - prevprob = 0.000177631
sumdiff - prevsumdiff = -3.15321e-006
Probability of good choice for population of 174=46.98
prob - prevprob = 0.000174568
sumdiff - prevsumdiff = -3.0626e-006
Probability of good choice for population of 176=46.9971
prob - prevprob = 0.000171593
sumdiff - prevsumdiff = -2.9756e-006
Probability of good choice for population of 178=47.014
prob - prevprob = 0.000168701
sumdiff - prevsumdiff = -2.89201e-006
Probability of good choice for population of 180=47.0306
prob - prevprob = 0.000165889
sumdiff - prevsumdiff = -2.81168e-006
Probability of good choice for population of 182=47.0469
prob - prevprob = 0.000163155
sumdiff - prevsumdiff = -2.73443e-006
Probability of good choice for population of 184=47.063
prob - prevprob = 0.000160494
sumdiff - prevsumdiff = -2.66013e-006
Probability of good choice for population of 186=47.0787
prob - prevprob = 0.000157906
sumdiff - prevsumdiff = -2.58862e-006
Probability of good choice for population of 188=47.0943
prob - prevprob = 0.000155386
sumdiff - prevsumdiff = -2.51977e-006
Probability of good choice for population of 190=47.1096
prob - prevprob = 0.000152933
sumdiff - prevsumdiff = -2.45346e-006
Probability of good choice for population of 192=47.1246
prob - prevprob = 0.000150543
sumdiff - prevsumdiff = -2.38957e-006
Probability of good choice for population of 194=47.1394
prob - prevprob = 0.000148215
sumdiff - prevsumdiff = -2.32798e-006
Probability of good choice for population of 196=47.154
prob - prevprob = 0.000145946
sumdiff - prevsumdiff = -2.2686e-006
Probability of good choice for population of 198=47.1684
prob - prevprob = 0.000143735
sumdiff - prevsumdiff = -2.21131e-006
Probability of good choice for population of 200=47.1826
prob - prevprob = 0.000141579
sumdiff - prevsumdiff = -2.15603e-006
Probability of good choice for population of 202=47.1965
```

```
prob - prevprob = 0.000139476
sumdiff - prevsumdiff = -2.10266e-006
Probability of good choice for population of 204=47.2103
prob - prevprob = 0.000137425
sumdiff - prevsumdiff = -2.05112e-006
Probability of good choice for population of 206=47.2238
prob - prevprob = 0.000135424
sumdiff - prevsumdiff = -2.00134e-006
Probability of good choice for population of 208=47.2372
prob - prevprob = 0.000133471
sumdiff - prevsumdiff = -1.95323e-006
Probability of good choice for population of 210=47.2503
prob - prevprob = 0.000131564
sumdiff - prevsumdiff = -1.90672e-006
Probability of good choice for population of 212=47.2633
prob - prevprob = 0.000129702
sumdiff - prevsumdiff = -1.86175e-006
Probability of good choice for population of 214=47.2761
prob - prevprob = 0.000127884
sumdiff - prevsumdiff = -1.81826e-006
Probability of good choice for population of 216=47.2887
prob - prevprob = 0.000126108
sumdiff - prevsumdiff = -1.77617e-006
Probability of good choice for population of 218=47.3011
prob - prevprob = 0.000124372
sumdiff - prevsumdiff = -1.73543e-006
Probability of good choice for population of 220=47.3134
prob - prevprob = 0.000122676
sumdiff - prevsumdiff = -1.69599e-006
Probability of good choice for population of 222=47.3255
prob - prevprob = 0.000121019
sumdiff - prevsumdiff = -1.65779e-006
Probability of good choice for population of 224=47.3374
prob - prevprob = 0.000119398
sumdiff - prevsumdiff = -1.62079e-006
Probability of good choice for population of 226=47.3492
prob - prevprob = 0.000117813
sumdiff - prevsumdiff = -1.58493e-006
Probability of good choice for population of 228=47.3608
prob - prevprob = 0.000116263
sumdiff - prevsumdiff = -1.55017e-006
Probability of good choice for population of 230=47.3723
prob - prevprob = 0.000114746
sumdiff - prevsumdiff = -1.51647e-006
Probability of good choice for population of 232=47.3836
prob - prevprob = 0.000113262
sumdiff - prevsumdiff = -1.48379e-006
Probability of good choice for population of 234=47.3948
```

```
prob - prevprob = 0.00011181
sumdiff - prevsumdiff = -1.45208e-006
Probability of good choice for population of 236=47.4059
prob - prevprob = 0.000110389
sumdiff - prevsumdiff = -1.42132e-006
Probability of good choice for population of 238=47.4168
prob - prevprob = 0.000108998
sumdiff - prevsumdiff = -1.39146e-006
Probability of good choice for population of 240=47.4275
prob - prevprob = 0.000107635
sumdiff - prevsumdiff = -1.36247e-006
Probability of good choice for population of 242=47.4381
prob - prevprob = 0.000106301
sumdiff - prevsumdiff = -1.33432e-006
Probability of good choice for population of 244=47.4486
prob - prevprob = 0.000104994
sumdiff - prevsumdiff = -1.30698e-006
Probability of good choice for population of 246=47.459
prob - prevprob = 0.000103713
sumdiff - prevsumdiff = -1.28041e-006
Probability of good choice for population of 248=47.4693
prob - prevprob = 0.000102459
sumdiff - prevsumdiff = -1.2546e-006
Probability of good choice for population of 250=47.4794
prob - prevprob = 0.000101229
sumdiff - prevsumdiff = -1.22951e-006
Probability of good choice for population of 252=47.4894
prob - prevprob = 0.000100024
sumdiff - prevsumdiff = -1.20511e-006
Probability of good choice for population of 254=47.4993
prob - prevprob = 9.88428e-005
sumdiff - prevsumdiff = -1.18139e-006
Probability of good choice for population of 256=47.509
prob - prevprob = 9.76845e-005
sumdiff - prevsumdiff = -1.15831e-006
Probability of good choice for population of 258=47.5187
prob - prevprob = 9.65487e-005
sumdiff - prevsumdiff = -1.13587e-006
Probability of good choice for population of 260=47.5282
prob - prevprob = 9.54346e-005
sumdiff - prevsumdiff = -1.11402e-006
Probability of good choice for population of 262=47.5377
prob - prevprob = 9.43419e-005
sumdiff - prevsumdiff = -1.09276e-006
Probability of good choice for population of 264=47.547
prob - prevprob = 9.32698e-005
sumdiff - prevsumdiff = -1.07207e-006
Probability of good choice for population of 266=47.5562
```

```
prob - prevprob = 9.22179e-005
sumdiff - prevsumdiff = -1.05192e-006
Probability of good choice for population of 268=47.5653
prob - prevprob = 9.11856e-005
sumdiff - prevsumdiff = -1.03229e-006
Probability of good choice for population of 270=47.5744
prob - prevprob = 9.01724e-005
sumdiff - prevsumdiff = -1.01317e-006
Probability of good choice for population of 272=47.5833
prob - prevprob = 8.91779e-005
sumdiff - prevsumdiff = -9.94549e-007
Probability of good choice for population of 274=47.5921
prob - prevprob = 8.82015e-005
sumdiff - prevsumdiff = -9.764e-007
Probability of good choice for population of 276=47.6008
prob - prevprob = 8.72428e-005
sumdiff - prevsumdiff = -9.58712e-007
Probability of good choice for population of 278=47.6095
prob - prevprob = 8.63013e-005
sumdiff - prevsumdiff = -9.41469e-007
Probability of good choice for population of 280=47.618
prob - prevprob = 8.53766e-005
sumdiff - prevsumdiff = -9.24657e-007
Probability of good choice for population of 282=47.6264
prob - prevprob = 8.44684e-005
sumdiff - prevsumdiff = -9.08262e-007
Probability of good choice for population of 284=47.6348
prob - prevprob = 8.35761e-005
sumdiff - prevsumdiff = -8.92272e-007
Probability of good choice for population of 286=47.6431
prob - prevprob = 8.26994e-005
sumdiff - prevsumdiff = -8.76672e-007
Probability of good choice for population of 288=47.6512
prob - prevprob = 8.1838e-005
sumdiff - prevsumdiff = -8.61452e-007
Probability of good choice for population of 290=47.6593
prob - prevprob = 8.09914e-005
sumdiff - prevsumdiff = -8.466e-007
Probability of good choice for population of 292=47.6674
prob - prevprob = 8.01593e-005
sumdiff - prevsumdiff = -8.32103e-007
Probability of good choice for population of 294=47.6753
prob - prevprob = 7.93413e-005
sumdiff - prevsumdiff = -8.17952e-007
Probability of good choice for population of 296=47.6832
prob - prevprob = 7.85372e-005
sumdiff - prevsumdiff = -8.04135e-007
Probability of good choice for population of 298=47.6909
```

```
prob - prevprob = 7.77466e-005
sumdiff - prevsumdiff = -7.90643e-007
Probability of good choice for population of 300=47.6986
prob - prevprob = 7.69691e-005
sumdiff - prevsumdiff = -7.77466e-007
Probability of good choice for population of 302=47.7062
prob - prevprob = 7.62045e-005
sumdiff - prevsumdiff = -7.64594e-007
Probability of good choice for population of 304=47.7138
prob - prevprob = 7.54525e-005
sumdiff - prevsumdiff = -7.52018e-007
Probability of good choice for population of 306=47.7213
prob - prevprob = 7.47127e-005
sumdiff - prevsumdiff = -7.3973e-007
Probability of good choice for population of 308=47.7287
prob - prevprob = 7.3985e-005
sumdiff - prevsumdiff = -7.27722e-007
Probability of good choice for population of 310=47.736
prob - prevprob = 7.3269e-005
sumdiff - prevsumdiff = -7.15984e-007
Probability of good choice for population of 312=47.7432
prob - prevprob = 7.25645e-005
sumdiff - prevsumdiff = -7.0451e-007
Probability of good choice for population of 314=47.7504
prob - prevprob = 7.18712e-005
sumdiff - prevsumdiff = -6.93292e-007
Probability of good choice for population of 316=47.7575
prob - prevprob = 7.11889e-005
sumdiff - prevsumdiff = -6.82322e-007
Probability of good choice for population of 318=47.7646
prob - prevprob = 7.05173e-005
sumdiff - prevsumdiff = -6.71594e-007
Probability of good choice for population of 320=47.7716
prob - prevprob = 6.98562e-005
sumdiff - prevsumdiff = -6.611e-007
Probability of good choice for population of 322=47.7785
prob - prevprob = 6.92054e-005
sumdiff - prevsumdiff = -6.50834e-007
Probability of good choice for population of 324=47.7854
prob - prevprob = 6.85646e-005
sumdiff - prevsumdiff = -6.40791e-007
Probability of good choice for population of 326=47.7922
prob - prevprob = 6.79336e-005
sumdiff - prevsumdiff = -6.30963e-007
Probability of good choice for population of 328=47.7989
prob - prevprob = 6.73123e-005
sumdiff - prevsumdiff = -6.21344e-007
Probability of good choice for population of 330=47.8056
```

```
prob - prevprob = 6.67004e-005
sumdiff - prevsumdiff = -6.1193e-007
Probability of good choice for population of 332=47.8122
prob - prevprob = 6.60976e-005
sumdiff - prevsumdiff = -6.02714e-007
Probability of good choice for population of 334=47.8187
prob - prevprob = 6.5504e-005
sumdiff - prevsumdiff = -5.93691e-007
Probability of good choice for population of 336=47.8252
prob - prevprob = 6.49191e-005
sumdiff - prevsumdiff = -5.84857e-007
Probability of good choice for population of 338=47.8316
prob - prevprob = 6.43429e-005
sumdiff - prevsumdiff = -5.76205e-007
Probability of good choice for population of 340=47.838
prob - prevprob = 6.37752e-005
sumdiff - prevsumdiff = -5.67731e-007
Probability of good choice for population of 342=47.8443
prob - prevprob = 6.32157e-005
sumdiff - prevsumdiff = -5.59431e-007
Probability of good choice for population of 344=47.8506
prob - prevprob = 6.26644e-005
sumdiff - prevsumdiff = -5.513e-007
Probability of good choice for population of 346=47.8568
prob - prevprob = 6.21211e-005
sumdiff - prevsumdiff = -5.43333e-007
Probability of good choice for population of 348=47.863
prob - prevprob = 6.15856e-005
sumdiff - prevsumdiff = -5.35527e-007
Probability of good choice for population of 350=47.8691
prob - prevprob = 6.10577e-005
sumdiff - prevsumdiff = -5.27876e-007
Probability of good choice for population of 352=47.8751
prob - prevprob = 6.05373e-005
sumdiff - prevsumdiff = -5.20378e-007
Probability of good choice for population of 354=47.8811
prob - prevprob = 6.00243e-005
sumdiff - prevsumdiff = -5.13028e-007
Probability of good choice for population of 356=47.8871
prob - prevprob = 5.95185e-005
sumdiff - prevsumdiff = -5.05823e-007
Probability of good choice for population of 358=47.893
prob - prevprob = 5.90197e-005
sumdiff - prevsumdiff = -4.98758e-007
Probability of good choice for population of 360=47.8988
prob - prevprob = 5.85279e-005
sumdiff - prevsumdiff = -4.91831e-007
Probability of good choice for population of 362=47.9047
```

```
prob - prevprob = 5.80428e-005
sumdiff - prevsumdiff = -4.85038e-007
Probability of good choice for population of 364=47.9104
prob - prevprob = 5.75645e-005
sumdiff - prevsumdiff = -4.78375e-007
Probability of good choice for population of 366=47.9161
prob - prevprob = 5.70926e-005
sumdiff - prevsumdiff = -4.7184e-007
Probability of good choice for population of 368=47.9218
prob - prevprob = 5.66272e-005
sumdiff - prevsumdiff = -4.65429e-007
Probability of good choice for population of 370=47.9274
prob - prevprob = 5.61681e-005
sumdiff - prevsumdiff = -4.59139e-007
Probability of good choice for population of 372=47.933
prob - prevprob = 5.57151e-005
sumdiff - prevsumdiff = -4.52968e-007
Probability of good choice for population of 374=47.9385
prob - prevprob = 5.52682e-005
sumdiff - prevsumdiff = -4.46912e-007
Probability of good choice for population of 376=47.944
prob - prevprob = 5.48272e-005
sumdiff - prevsumdiff = -4.40969e-007
Probability of good choice for population of 378=47.9494
prob - prevprob = 5.43921e-005
sumdiff - prevsumdiff = -4.35137e-007
Probability of good choice for population of 380=47.9548
prob - prevprob = 5.39627e-005
sumdiff - prevsumdiff = -4.29411e-007
Probability of good choice for population of 382=47.9602
prob - prevprob = 5.35389e-005
sumdiff - prevsumdiff = -4.23791e-007
Probability of good choice for population of 384=47.9655
prob - prevprob = 5.31206e-005
sumdiff - prevsumdiff = -4.18272e-007
Probability of good choice for population of 386=47.9708
prob - prevprob = 5.27077e-005
sumdiff - prevsumdiff = -4.12854e-007
Probability of good choice for population of 388=47.976
prob - prevprob = 5.23002e-005
sumdiff - prevsumdiff = -4.07534e-007
Probability of good choice for population of 390=47.9812
prob - prevprob = 5.18979e-005
sumdiff - prevsumdiff = -4.02309e-007
Probability of good choice for population of 392=47.9863
prob - prevprob = 5.15007e-005
sumdiff - prevsumdiff = -3.97178e-007
Probability of good choice for population of 394=47.9914
```

```
prob - prevprob = 5.11086e-005
sumdiff - prevsumdiff = -3.92137e-007
Probability of good choice for population of 396=47.9965
prob - prevprob = 5.07214e-005
sumdiff - prevsumdiff = -3.87186e-007
Probability of good choice for population of 398=48.0015
prob - prevprob = 5.03391e-005
sumdiff - prevsumdiff = -3.82322e-007
Probability of good choice for population of 400=48.0065
prob - prevprob = 4.99615e-005
sumdiff - prevsumdiff = -3.77543e-007
Probability of good choice for population of 402=48.0115
prob - prevprob = 4.95887e-005
sumdiff - prevsumdiff = -3.72847e-007
Probability of good choice for population of 404=48.0164
prob - prevprob = 4.92205e-005
sumdiff - prevsumdiff = -3.68233e-007
Probability of good choice for population of 406=48.0213
prob - prevprob = 4.88568e-005
sumdiff - prevsumdiff = -3.63698e-007
Probability of good choice for population of 408=48.0262
prob - prevprob = 4.84975e-005
sumdiff - prevsumdiff = -3.59241e-007
Probability of good choice for population of 410=48.031
prob - prevprob = 4.81427e-005
sumdiff - prevsumdiff = -3.5486e-007
Probability of good choice for population of 412=48.0357
prob - prevprob = 4.77921e-005
sumdiff - prevsumdiff = -3.50553e-007
Probability of good choice for population of 414=48.0405
prob - prevprob = 4.74458e-005
sumdiff - prevsumdiff = -3.4632e-007
Probability of good choice for population of 416=48.0452
prob - prevprob = 4.71036e-005
sumdiff - prevsumdiff = -3.42157e-007
Probability of good choice for population of 418=48.0499
prob - prevprob = 4.67656e-005
sumdiff - prevsumdiff = -3.38064e-007
Probability of good choice for population of 420=48.0545
prob - prevprob = 4.64315e-005
sumdiff - prevsumdiff = -3.3404e-007
Probability of good choice for population of 422=48.0591
prob - prevprob = 4.61014e-005
sumdiff - prevsumdiff = -3.30082e-007
Probability of good choice for population of 424=48.0637
prob - prevprob = 4.57752e-005
sumdiff - prevsumdiff = -3.26189e-007
Probability of good choice for population of 426=48.0683
```

```
prob - prevprob = 4.54529e-005
sumdiff - prevsumdiff = -3.22361e-007
Probability of good choice for population of 428=48.0728
prob - prevprob = 4.51343e-005
sumdiff - prevsumdiff = -3.18595e-007
Probability of good choice for population of 430=48.0772
prob - prevprob = 4.48194e-005
sumdiff - prevsumdiff = -3.1489e-007
Probability of good choice for population of 432=48.0817
prob - prevprob = 4.45082e-005
sumdiff - prevsumdiff = -3.11246e-007
Probability of good choice for population of 434=48.0861
prob - prevprob = 4.42005e-005
sumdiff - prevsumdiff = -3.0766e-007
Probability of good choice for population of 436=48.0905
prob - prevprob = 4.38964e-005
sumdiff - prevsumdiff = -3.04132e-007
Probability of good choice for population of 438=48.0949
prob - prevprob = 4.35957e-005
sumdiff - prevsumdiff = -3.0066e-007
Probability of good choice for population of 440=48.0992
prob - prevprob = 4.32985e-005
sumdiff - prevsumdiff = -2.97243e-007
Probability of good choice for population of 442=48.1035
prob - prevprob = 4.30046e-005
sumdiff - prevsumdiff = -2.93881e-007
Probability of good choice for population of 444=48.1078
prob - prevprob = 4.2714e-005
sumdiff - prevsumdiff = -2.90571e-007
Probability of good choice for population of 446=48.112
prob - prevprob = 4.24267e-005
sumdiff - prevsumdiff = -2.87314e-007
Probability of good choice for population of 448=48.1162
prob - prevprob = 4.21426e-005
sumdiff - prevsumdiff = -2.84107e-007
Probability of good choice for population of 450=48.1204
prob - prevprob = 4.18616e-005
sumdiff - prevsumdiff = -2.80951e-007
Probability of good choice for population of 452=48.1246
prob - prevprob = 4.15838e-005
sumdiff - prevsumdiff = -2.77843e-007
Probability of good choice for population of 454=48.1287
prob - prevprob = 4.1309e-005
sumdiff - prevsumdiff = -2.74783e-007
Probability of good choice for population of 456=48.1328
prob - prevprob = 4.10372e-005
sumdiff - prevsumdiff = -2.7177e-007
Probability of good choice for population of 458=48.1369
```

```
prob - prevprob = 4.07684e-005
sumdiff - prevsumdiff = -2.68803e-007
Probability of good choice for population of 460=48.1409
prob - prevprob = 4.05026e-005
sumdiff - prevsumdiff = -2.65881e-007
Probability of good choice for population of 462=48.145
prob - prevprob = 4.02396e-005
sumdiff - prevsumdiff = -2.63004e-007
Probability of good choice for population of 464=48.149
prob - prevprob = 3.99794e-005
sumdiff - prevsumdiff = -2.6017e-007
Probability of good choice for population of 466=48.1529
prob - prevprob = 3.9722e-005
sumdiff - prevsumdiff = -2.57378e-007
Probability of good choice for population of 468=48.1569
prob - prevprob = 3.94674e-005
sumdiff - prevsumdiff = -2.54628e-007
Probability of good choice for population of 470=48.1608
prob - prevprob = 3.92155e-005
sumdiff - prevsumdiff = -2.51919e-007
Probability of good choice for population of 472=48.1647
prob - prevprob = 3.89662e-005
sumdiff - prevsumdiff = -2.49251e-007
Probability of good choice for population of 474=48.1686
prob - prevprob = 3.87196e-005
sumdiff - prevsumdiff = -2.46622e-007
Probability of good choice for population of 476=48.1724
prob - prevprob = 3.84756e-005
sumdiff - prevsumdiff = -2.44031e-007
Probability of good choice for population of 478=48.1762
prob - prevprob = 3.82341e-005
sumdiff - prevsumdiff = -2.41478e-007
Probability of good choice for population of 480=48.18
prob - prevprob = 3.79951e-005
sumdiff - prevsumdiff = -2.38963e-007
Probability of good choice for population of 482=48.1838
prob - prevprob = 3.77586e-005
sumdiff - prevsumdiff = -2.36484e-007
Probability of good choice for population of 484=48.1876
prob - prevprob = 3.75246e-005
sumdiff - prevsumdiff = -2.34041e-007
Probability of good choice for population of 486=48.1913
prob - prevprob = 3.7293e-005
sumdiff - prevsumdiff = -2.31633e-007
Probability of good choice for population of 488=48.195
prob - prevprob = 3.70637e-005
sumdiff - prevsumdiff = -2.2926e-007
Probability of good choice for population of 490=48.1987
```

```
prob - prevprob = 3.68368e-005
sumdiff - prevsumdiff = -2.26921e-007
Probability of good choice for population of 492=48.2023
prob - prevprob = 3.66122e-005
sumdiff - prevsumdiff = -2.24614e-007
Probability of good choice for population of 494=48.206
prob - prevprob = 3.63898e-005
sumdiff - prevsumdiff = -2.22341e-007
Probability of good choice for population of 496=48.2096
prob - prevprob = 3.61697e-005
sumdiff - prevsumdiff = -2.201e-007
Probability of good choice for population of 498=48.2132
prob - prevprob = 3.59518e-005
sumdiff - prevsumdiff = -2.1789e-007
Probability of good choice for population of 500=48.2168
prob - prevprob = 3.57361e-005
sumdiff - prevsumdiff = -2.15711e-007
Probability of good choice for population of 502=48.2203
prob - prevprob = 3.55226e-005
sumdiff - prevsumdiff = -2.13562e-007
Probability of good choice for population of 504=48.2239
prob - prevprob = 3.53111e-005
sumdiff - prevsumdiff = -2.11444e-007
Probability of good choice for population of 506=48.2274
prob - prevprob = 3.51018e-005
sumdiff - prevsumdiff = -2.09354e-007
Probability of good choice for population of 508=48.2309
prob - prevprob = 3.48945e-005
sumdiff - prevsumdiff = -2.07294e-007
Probability of good choice for population of 510=48.2343
prob - prevprob = 3.46892e-005
sumdiff - prevsumdiff = -2.05262e-007
Probability of good choice for population of 512=48.2378
prob - prevprob = 3.44859e-005
sumdiff - prevsumdiff = -2.03257e-007
Probability of good choice for population of 514=48.2412
prob - prevprob = 3.42847e-005
sumdiff - prevsumdiff = -2.0128e-007
Probability of good choice for population of 516=48.2446
prob - prevprob = 3.40853e-005
sumdiff - prevsumdiff = -1.99329e-007
Probability of good choice for population of 518=48.248
prob - prevprob = 3.38879e-005
sumdiff - prevsumdiff = -1.97405e-007
Probability of good choice for population of 520=48.2514
prob - prevprob = 3.36924e-005
sumdiff - prevsumdiff = -1.95507e-007
Probability of good choice for population of 522=48.2547
```

```
prob - prevprob = 3.34988e-005
sumdiff - prevsumdiff = -1.93635e-007
Probability of good choice for population of 524=48.258
prob - prevprob = 3.3307e-005
sumdiff - prevsumdiff = -1.91787e-007
Probability of good choice for population of 526=48.2614
prob - prevprob = 3.3117e-005
sumdiff - prevsumdiff = -1.89964e-007
Probability of good choice for population of 528=48.2646
prob - prevprob = 3.29289e-005
sumdiff - prevsumdiff = -1.88165e-007
Probability of good choice for population of 530=48.2679
prob - prevprob = 3.27425e-005
sumdiff - prevsumdiff = -1.8639e-007
Probability of good choice for population of 532=48.2712
prob - prevprob = 3.25578e-005
sumdiff - prevsumdiff = -1.84638e-007
Probability of good choice for population of 534=48.2744
prob - prevprob = 3.23749e-005
sumdiff - prevsumdiff = -1.82909e-007
Probability of good choice for population of 536=48.2776
prob - prevprob = 3.21937e-005
sumdiff - prevsumdiff = -1.81203e-007
Probability of good choice for population of 538=48.2808
prob - prevprob = 3.20142e-005
sumdiff - prevsumdiff = -1.79519e-007
Probability of good choice for population of 540=48.284
prob - prevprob = 3.18364e-005
sumdiff - prevsumdiff = -1.77857e-007
Probability of good choice for population of 542=48.2872
prob - prevprob = 3.16601e-005
sumdiff - prevsumdiff = -1.76216e-007
Probability of good choice for population of 544=48.2903
prob - prevprob = 3.14855e-005
sumdiff - prevsumdiff = -1.74596e-007
Probability of good choice for population of 546=48.2935
prob - prevprob = 3.13125e-005
sumdiff - prevsumdiff = -1.72997e-007
Probability of good choice for population of 548=48.2966
prob - prevprob = 3.11411e-005
sumdiff - prevsumdiff = -1.71419e-007
Probability of good choice for population of 550=48.2997
prob - prevprob = 3.09713e-005
sumdiff - prevsumdiff = -1.69861e-007
Probability of good choice for population of 552=48.3028
prob - prevprob = 3.08029e-005
sumdiff - prevsumdiff = -1.68322e-007
Probability of good choice for population of 554=48.3058
```

```
prob - prevprob = 3.06361e-005
sumdiff - prevsumdiff = -1.66803e-007
Probability of good choice for population of 556=48.3089
prob - prevprob = 3.04708e-005
sumdiff - prevsumdiff = -1.65303e-007
Probability of good choice for population of 558=48.3119
prob - prevprob = 3.0307e-005
sumdiff - prevsumdiff = -1.63822e-007
Probability of good choice for population of 560=48.3149
prob - prevprob = 3.01447e-005
sumdiff - prevsumdiff = -1.62359e-007
Probability of good choice for population of 562=48.3179
prob - prevprob = 2.99837e-005
sumdiff - prevsumdiff = -1.60915e-007
Probability of good choice for population of 564=48.3209
prob - prevprob = 2.98243e-005
sumdiff - prevsumdiff = -1.59488e-007
Probability of good choice for population of 566=48.3239
prob - prevprob = 2.96662e-005
sumdiff - prevsumdiff = -1.58079e-007
Probability of good choice for population of 568=48.3268
prob - prevprob = 2.95095e-005
sumdiff - prevsumdiff = -1.56688e-007
Probability of good choice for population of 570=48.3297
prob - prevprob = 2.93542e-005
sumdiff - prevsumdiff = -1.55313e-007
Probability of good choice for population of 572=48.3327
prob - prevprob = 2.92002e-005
sumdiff - prevsumdiff = -1.53955e-007
Probability of good choice for population of 574=48.3356
prob - prevprob = 2.90476e-005
sumdiff - prevsumdiff = -1.52614e-007
Probability of good choice for population of 576=48.3385
prob - prevprob = 2.88963e-005
sumdiff - prevsumdiff = -1.5129e-007
Probability of good choice for population of 578=48.3413
prob - prevprob = 2.87463e-005
sumdiff - prevsumdiff = -1.49981e-007
Probability of good choice for population of 580=48.3442
prob - prevprob = 2.85976e-005
sumdiff - prevsumdiff = -1.48688e-007
Probability of good choice for population of 582=48.347
prob - prevprob = 2.84502e-005
sumdiff - prevsumdiff = -1.47411e-007
Probability of good choice for population of 584=48.3499
prob - prevprob = 2.83041e-005
sumdiff - prevsumdiff = -1.46148e-007
Probability of good choice for population of 586=48.3527
```

```
prob - prevprob = 2.81592e-005
sumdiff - prevsumdiff = -1.44901e-007
Probability of good choice for population of 588=48.3555
prob - prevprob = 2.80155e-005
sumdiff - prevsumdiff = -1.43669e-007
Probability of good choice for population of 590=48.3583
prob - prevprob = 2.78731e-005
sumdiff - prevsumdiff = -1.42452e-007
Probability of good choice for population of 592=48.361
prob - prevprob = 2.77318e-005
sumdiff - prevsumdiff = -1.41249e-007
Probability of good choice for population of 594=48.3638
prob - prevprob = 2.75918e-005
sumdiff - prevsumdiff = -1.4006e-007
Probability of good choice for population of 596=48.3666
prob - prevprob = 2.74529e-005
sumdiff - prevsumdiff = -1.38885e-007
Probability of good choice for population of 598=48.3693
prob - prevprob = 2.73151e-005
sumdiff - prevsumdiff = -1.37723e-007
Probability of good choice for population of 600=48.372
prob - prevprob = 2.71786e-005
sumdiff - prevsumdiff = -1.36576e-007
Probability of good choice for population of 602=48.3747
prob - prevprob = 2.70431e-005
sumdiff - prevsumdiff = -1.35441e-007
Probability of good choice for population of 604=48.3774
prob - prevprob = 2.69088e-005
sumdiff - prevsumdiff = -1.3432e-007
Probability of good choice for population of 606=48.3801
prob - prevprob = 2.67756e-005
sumdiff - prevsumdiff = -1.33212e-007
Probability of good choice for population of 608=48.3827
prob - prevprob = 2.66435e-005
sumdiff - prevsumdiff = -1.32116e-007
Probability of good choice for population of 610=48.3854
prob - prevprob = 2.65125e-005
sumdiff - prevsumdiff = -1.31034e-007
Probability of good choice for population of 612=48.388
prob - prevprob = 2.63825e-005
sumdiff - prevsumdiff = -1.29963e-007
Probability of good choice for population of 614=48.3907
prob - prevprob = 2.62536e-005
sumdiff - prevsumdiff = -1.28905e-007
Probability of good choice for population of 616=48.3933
prob - prevprob = 2.61257e-005
sumdiff - prevsumdiff = -1.27858e-007
Probability of good choice for population of 618=48.3959
```

```
prob - prevprob = 2.59989e-005
sumdiff - prevsumdiff = -1.26824e-007
Probability of good choice for population of 620=48.3985
prob - prevprob = 2.58731e-005
sumdiff - prevsumdiff = -1.25801e-007
Probability of good choice for population of 622=48.401
prob - prevprob = 2.57483e-005
sumdiff - prevsumdiff = -1.2479e-007
Probability of good choice for population of 624=48.4036
prob - prevprob = 2.56245e-005
sumdiff - prevsumdiff = -1.2379e-007
Probability of good choice for population of 626=48.4061
prob - prevprob = 2.55017e-005
sumdiff - prevsumdiff = -1.22801e-007
Probability of good choice for population of 628=48.4087
prob - prevprob = 2.53799e-005
sumdiff - prevsumdiff = -1.21823e-007
Probability of good choice for population of 630=48.4112
prob - prevprob = 2.5259e-005
sumdiff - prevsumdiff = -1.20857e-007
Probability of good choice for population of 632=48.4137
prob - prevprob = 2.51391e-005
sumdiff - prevsumdiff = -1.199e-007
Probability of good choice for population of 634=48.4162
prob - prevprob = 2.50202e-005
sumdiff - prevsumdiff = -1.18955e-007
Probability of good choice for population of 636=48.4187
prob - prevprob = 2.49022e-005
sumdiff - prevsumdiff = -1.1802e-007
Probability of good choice for population of 638=48.4212
prob - prevprob = 2.47851e-005
sumdiff - prevsumdiff = -1.17095e-007
Probability of good choice for population of 640=48.4237
prob - prevprob = 2.46689e-005
sumdiff - prevsumdiff = -1.1618e-007
Probability of good choice for population of 642=48.4261
prob - prevprob = 2.45536e-005
sumdiff - prevsumdiff = -1.15275e-007
Probability of good choice for population of 644=48.4286
prob - prevprob = 2.44392e-005
sumdiff - prevsumdiff = -1.1438e-007
Probability of good choice for population of 646=48.431
prob - prevprob = 2.43257e-005
sumdiff - prevsumdiff = -1.13495e-007
Probability of good choice for population of 648=48.4334
prob - prevprob = 2.42131e-005
sumdiff - prevsumdiff = -1.12619e-007
Probability of good choice for population of 650=48.4358
```

```
prob - prevprob = 2.41014e-005
sumdiff - prevsumdiff = -1.11753e-007
Probability of good choice for population of 652=48.4382
prob - prevprob = 2.39905e-005
sumdiff - prevsumdiff = -1.10896e-007
Probability of good choice for population of 654=48.4406
prob - prevprob = 2.38804e-005
sumdiff - prevsumdiff = -1.10048e-007
Probability of good choice for population of 656=48.443
prob - prevprob = 2.37712e-005
sumdiff - prevsumdiff = -1.09209e-007
Probability of good choice for population of 658=48.4454
prob - prevprob = 2.36628e-005
sumdiff - prevsumdiff = -1.08379e-007
Probability of good choice for population of 660=48.4477
prob - prevprob = 2.35553e-005
sumdiff - prevsumdiff = -1.07558e-007
Probability of good choice for population of 662=48.4501
prob - prevprob = 2.34485e-005
sumdiff - prevsumdiff = -1.06746e-007
Probability of good choice for population of 664=48.4524
prob - prevprob = 2.33426e-005
sumdiff - prevsumdiff = -1.05942e-007
Probability of good choice for population of 666=48.4547
prob - prevprob = 2.32374e-005
sumdiff - prevsumdiff = -1.05147e-007
Probability of good choice for population of 668=48.457
prob - prevprob = 2.31331e-005
sumdiff - prevsumdiff = -1.0436e-007
Probability of good choice for population of 670=48.4593
prob - prevprob = 2.30295e-005
sumdiff - prevsumdiff = -1.03581e-007
Probability of good choice for population of 672=48.4616
prob - prevprob = 2.29267e-005
sumdiff - prevsumdiff = -1.0281e-007
Probability of good choice for population of 674=48.4639
prob - prevprob = 2.28246e-005
sumdiff - prevsumdiff = -1.02048e-007
Probability of good choice for population of 676=48.4662
prob - prevprob = 2.27233e-005
sumdiff - prevsumdiff = -1.01293e-007
Probability of good choice for population of 678=48.4684
prob - prevprob = 2.26228e-005
sumdiff - prevsumdiff = -1.00546e-007
Probability of good choice for population of 680=48.4707
prob - prevprob = 2.2523e-005
sumdiff - prevsumdiff = -9.98065e-008
Probability of good choice for population of 682=48.4729
```

```
prob - prevprob = 2.24239e-005
sumdiff - prevsumdiff = -9.90748e-008
Probability of good choice for population of 684=48.4752
prob - prevprob = 2.23256e-005
sumdiff - prevsumdiff = -9.83505e-008
Probability of good choice for population of 686=48.4774
prob - prevprob = 2.22279e-005
sumdiff - prevsumdiff = -9.76337e-008
Probability of good choice for population of 688=48.4796
prob - prevprob = 2.2131e-005
sumdiff - prevsumdiff = -9.69241e-008
Probability of good choice for population of 690=48.4818
prob - prevprob = 2.20348e-005
sumdiff - prevsumdiff = -9.62218e-008
Probability of good choice for population of 692=48.484
prob - prevprob = 2.19393e-005
sumdiff - prevsumdiff = -9.55266e-008
Probability of good choice for population of 694=48.4862
prob - prevprob = 2.18444e-005
sumdiff - prevsumdiff = -9.48383e-008
Probability of good choice for population of 696=48.4884
prob - prevprob = 2.17503e-005
sumdiff - prevsumdiff = -9.4157e-008
Probability of good choice for population of 698=48.4905
prob - prevprob = 2.16568e-005
sumdiff - prevsumdiff = -9.34825e-008
Probability of good choice for population of 700=48.4927
prob - prevprob = 2.1564e-005
sumdiff - prevsumdiff = -9.28148e-008
Probability of good choice for population of 702=48.4948
prob - prevprob = 2.14718e-005
sumdiff - prevsumdiff = -9.21537e-008
Probability of good choice for population of 704=48.497
prob - prevprob = 2.13803e-005
sumdiff - prevsumdiff = -9.14992e-008
Probability of good choice for population of 706=48.4991
prob - prevprob = 2.12895e-005
sumdiff - prevsumdiff = -9.08512e-008
Probability of good choice for population of 708=48.5012
prob - prevprob = 2.11993e-005
sumdiff - prevsumdiff = -9.02096e-008
Probability of good choice for population of 710=48.5033
prob - prevprob = 2.11097e-005
sumdiff - prevsumdiff = -8.95743e-008
Probability of good choice for population of 712=48.5054
prob - prevprob = 2.10207e-005
sumdiff - prevsumdiff = -8.89453e-008
Probability of good choice for population of 714=48.5075
```

```
prob - prevprob = 2.09324e-005
sumdiff - prevsumdiff = -8.83224e-008
Probability of good choice for population of 716=48.5096
prob - prevprob = 2.08447e-005
sumdiff - prevsumdiff = -8.77057e-008
Probability of good choice for population of 718=48.5117
prob - prevprob = 2.07576e-005
sumdiff - prevsumdiff = -8.70949e-008
Probability of good choice for population of 720=48.5137
prob - prevprob = 2.06711e-005
sumdiff - prevsumdiff = -8.64901e-008
Probability of good choice for population of 722=48.5158
prob - prevprob = 2.05852e-005
sumdiff - prevsumdiff = -8.58911e-008
Probability of good choice for population of 724=48.5179
prob - prevprob = 2.04999e-005
sumdiff - prevsumdiff = -8.52979e-008
Probability of good choice for population of 726=48.5199
prob - prevprob = 2.04152e-005
sumdiff - prevsumdiff = -8.47105e-008
Probability of good choice for population of 728=48.5219
prob - prevprob = 2.03311e-005
sumdiff - prevsumdiff = -8.41287e-008
Probability of good choice for population of 730=48.524
prob - prevprob = 2.02475e-005
sumdiff - prevsumdiff = -8.35525e-008
Probability of good choice for population of 732=48.526
prob - prevprob = 2.01646e-005
sumdiff - prevsumdiff = -8.29817e-008
Probability of good choice for population of 734=48.528
prob - prevprob = 2.00821e-005
sumdiff - prevsumdiff = -8.24165e-008
Probability of good choice for population of 736=48.53
prob - prevprob = 2.00003e-005
sumdiff - prevsumdiff = -8.18566e-008
Probability of good choice for population of 738=48.532
prob - prevprob = 1.9919e-005
sumdiff - prevsumdiff = -8.1302e-008
Probability of good choice for population of 740=48.534
prob - prevprob = 1.98382e-005
sumdiff - prevsumdiff = -8.07527e-008
Probability of good choice for population of 742=48.5359
prob - prevprob = 1.9758e-005
sumdiff - prevsumdiff = -8.02085e-008
Probability of good choice for population of 744=48.5379
prob - prevprob = 1.96784e-005
sumdiff - prevsumdiff = -7.96695e-008
Probability of good choice for population of 746=48.5399
```

```
prob - prevprob = 1.95992e-005
sumdiff - prevsumdiff = -7.91355e-008
Probability of good choice for population of 748=48.5418
prob - prevprob = 1.95206e-005
sumdiff - prevsumdiff = -7.86065e-008
Probability of good choice for population of 750=48.5438
prob - prevprob = 1.94425e-005
sumdiff - prevsumdiff = -7.80825e-008
Probability of good choice for population of 752=48.5457
prob - prevprob = 1.9365e-005
sumdiff - prevsumdiff = -7.75633e-008
Probability of good choice for population of 754=48.5476
prob - prevprob = 1.92879e-005
sumdiff - prevsumdiff = -7.7049e-008
Probability of good choice for population of 756=48.5495
prob - prevprob = 1.92114e-005
sumdiff - prevsumdiff = -7.65394e-008
Probability of good choice for population of 758=48.5515
prob - prevprob = 1.91353e-005
sumdiff - prevsumdiff = -7.60345e-008
Probability of good choice for population of 760=48.5534
prob - prevprob = 1.90598e-005
sumdiff - prevsumdiff = -7.55343e-008
Probability of good choice for population of 762=48.5553
prob - prevprob = 1.89848e-005
sumdiff - prevsumdiff = -7.50386e-008
Probability of good choice for population of 764=48.5571
prob - prevprob = 1.89102e-005
sumdiff - prevsumdiff = -7.45475e-008
Probability of good choice for population of 766=48.559
prob - prevprob = 1.88362e-005
sumdiff - prevsumdiff = -7.40609e-008
Probability of good choice for population of 768=48.5609
prob - prevprob = 1.87626e-005
sumdiff - prevsumdiff = -7.35788e-008
Probability of good choice for population of 770=48.5628
prob - prevprob = 1.86895e-005
sumdiff - prevsumdiff = -7.3101e-008
Probability of good choice for population of 772=48.5646
prob - prevprob = 1.86169e-005
sumdiff - prevsumdiff = -7.26275e-008
Probability of good choice for population of 774=48.5665
prob - prevprob = 1.85447e-005
sumdiff - prevsumdiff = -7.21584e-008
Probability of good choice for population of 776=48.5683
prob - prevprob = 1.8473e-005
sumdiff - prevsumdiff = -7.16934e-008
Probability of good choice for population of 778=48.5702
```

```
prob - prevprob = 1.84018e-005
sumdiff - prevsumdiff = -7.12327e-008
Probability of good choice for population of 780=48.572
prob - prevprob = 1.8331e-005
sumdiff - prevsumdiff = -7.07761e-008
Probability of good choice for population of 782=48.5738
prob - prevprob = 1.82607e-005
sumdiff - prevsumdiff = -7.03235e-008
Probability of good choice for population of 784=48.5757
prob - prevprob = 1.81908e-005
sumdiff - prevsumdiff = -6.9875e-008
Probability of good choice for population of 786=48.5775
prob - prevprob = 1.81214e-005
sumdiff - prevsumdiff = -6.94305e-008
Probability of good choice for population of 788=48.5793
prob - prevprob = 1.80524e-005
sumdiff - prevsumdiff = -6.899e-008
Probability of good choice for population of 790=48.5811
prob - prevprob = 1.79838e-005
sumdiff - prevsumdiff = -6.85533e-008
Probability of good choice for population of 792=48.5829
prob - prevprob = 1.79157e-005
sumdiff - prevsumdiff = -6.81206e-008
Probability of good choice for population of 794=48.5847
prob - prevprob = 1.7848e-005
sumdiff - prevsumdiff = -6.76916e-008
Probability of good choice for population of 796=48.5864
prob - prevprob = 1.77807e-005
sumdiff - prevsumdiff = -6.72664e-008
Probability of good choice for population of 798=48.5882
prob - prevprob = 1.77139e-005
sumdiff - prevsumdiff = -6.68449e-008
Probability of good choice for population of 800=48.59
prob - prevprob = 1.76475e-005
sumdiff - prevsumdiff = -6.64271e-008
Probability of good choice for population of 802=48.5917
prob - prevprob = 1.75815e-005
sumdiff - prevsumdiff = -6.6013e-008
Probability of good choice for population of 804=48.5935
prob - prevprob = 1.75159e-005
sumdiff - prevsumdiff = -6.56025e-008
Probability of good choice for population of 806=48.5952
prob - prevprob = 1.74507e-005
sumdiff - prevsumdiff = -6.51955e-008
Probability of good choice for population of 808=48.597
prob - prevprob = 1.73859e-005
sumdiff - prevsumdiff = -6.47921e-008
Probability of good choice for population of 810=48.5987
```

```
prob - prevprob = 1.73215e-005
sumdiff - prevsumdiff = -6.43921e-008
Probability of good choice for population of 812=48.6004
prob - prevprob = 1.72575e-005
sumdiff - prevsumdiff = -6.39956e-008
Probability of good choice for population of 814=48.6021
prob - prevprob = 1.71939e-005
sumdiff - prevsumdiff = -6.36025e-008
Probability of good choice for population of 816=48.6039
prob - prevprob = 1.71307e-005
sumdiff - prevsumdiff = -6.32128e-008
Probability of good choice for population of 818=48.6056
prob - prevprob = 1.70678e-005
sumdiff - prevsumdiff = -6.28264e-008
Probability of good choice for population of 820=48.6073
prob - prevprob = 1.70054e-005
sumdiff - prevsumdiff = -6.24433e-008
Probability of good choice for population of 822=48.609
prob - prevprob = 1.69433e-005
sumdiff - prevsumdiff = -6.20635e-008
Probability of good choice for population of 824=48.6106
prob - prevprob = 1.68816e-005
sumdiff - prevsumdiff = -6.16869e-008
Probability of good choice for population of 826=48.6123
prob - prevprob = 1.68203e-005
sumdiff - prevsumdiff = -6.13135e-008
Probability of good choice for population of 828=48.614
prob - prevprob = 1.67594e-005
sumdiff - prevsumdiff = -6.09432e-008
Probability of good choice for population of 830=48.6157
prob - prevprob = 1.66988e-005
sumdiff - prevsumdiff = -6.05761e-008
Probability of good choice for population of 832=48.6173
prob - prevprob = 1.66386e-005
sumdiff - prevsumdiff = -6.02121e-008
Probability of good choice for population of 834=48.619
prob - prevprob = 1.65788e-005
sumdiff - prevsumdiff = -5.98511e-008
Probability of good choice for population of 836=48.6206
prob - prevprob = 1.65193e-005
sumdiff - prevsumdiff = -5.94931e-008
Probability of good choice for population of 838=48.6223
prob - prevprob = 1.64601e-005
sumdiff - prevsumdiff = -5.91382e-008
Probability of good choice for population of 840=48.6239
prob - prevprob = 1.64013e-005
sumdiff - prevsumdiff = -5.87861e-008
Probability of good choice for population of 842=48.6256
```

```
prob - prevprob = 1.63429e-005
sumdiff - prevsumdiff = -5.84371e-008
Probability of good choice for population of 844=48.6272
prob - prevprob = 1.62848e-005
sumdiff - prevsumdiff = -5.80909e-008
Probability of good choice for population of 846=48.6288
prob - prevprob = 1.62271e-005
sumdiff - prevsumdiff = -5.77475e-008
Probability of good choice for population of 848=48.6304
prob - prevprob = 1.61697e-005
sumdiff - prevsumdiff = -5.74071e-008
Probability of good choice for population of 850=48.632
prob - prevprob = 1.61126e-005
sumdiff - prevsumdiff = -5.70694e-008
Probability of good choice for population of 852=48.6336
prob - prevprob = 1.60558e-005
sumdiff - prevsumdiff = -5.67344e-008
Probability of good choice for population of 854=48.6352
prob - prevprob = 1.59994e-005
sumdiff - prevsumdiff = -5.64023e-008
Probability of good choice for population of 856=48.6368
prob - prevprob = 1.59434e-005
sumdiff - prevsumdiff = -5.60728e-008
Probability of good choice for population of 858=48.6384
prob - prevprob = 1.58876e-005
sumdiff - prevsumdiff = -5.57461e-008
Probability of good choice for population of 860=48.64
prob - prevprob = 1.58322e-005
sumdiff - prevsumdiff = -5.5422e-008
Probability of good choice for population of 862=48.6416
prob - prevprob = 1.57771e-005
sumdiff - prevsumdiff = -5.51005e-008
Probability of good choice for population of 864=48.6432
prob - prevprob = 1.57223e-005
sumdiff - prevsumdiff = -5.47816e-008
Probability of good choice for population of 866=48.6447
prob - prevprob = 1.56679e-005
sumdiff - prevsumdiff = -5.44653e-008
Probability of good choice for population of 868=48.6463
prob - prevprob = 1.56137e-005
sumdiff - prevsumdiff = -5.41516e-008
Probability of good choice for population of 870=48.6478
prob - prevprob = 1.55599e-005
sumdiff - prevsumdiff = -5.38404e-008
Probability of good choice for population of 872=48.6494
prob - prevprob = 1.55063e-005
sumdiff - prevsumdiff = -5.35317e-008
Probability of good choice for population of 874=48.6509
```

```
prob - prevprob = 1.54531e-005
sumdiff - prevsumdiff = -5.32254e-008
Probability of good choice for population of 876=48.6525
prob - prevprob = 1.54002e-005
sumdiff - prevsumdiff = -5.29216e-008
Probability of good choice for population of 878=48.654
prob - prevprob = 1.53476e-005
sumdiff - prevsumdiff = -5.26202e-008
Probability of good choice for population of 880=48.6555
prob - prevprob = 1.52952e-005
sumdiff - prevsumdiff = -5.23213e-008
Probability of good choice for population of 882=48.6571
prob - prevprob = 1.52432e-005
sumdiff - prevsumdiff = -5.20246e-008
Probability of good choice for population of 884=48.6586
prob - prevprob = 1.51915e-005
sumdiff - prevsumdiff = -5.17304e-008
Probability of good choice for population of 886=48.6601
prob - prevprob = 1.51401e-005
sumdiff - prevsumdiff = -5.14385e-008
Probability of good choice for population of 888=48.6616
prob - prevprob = 1.50889e-005
sumdiff - prevsumdiff = -5.11488e-008
Probability of good choice for population of 890=48.6631
prob - prevprob = 1.5038e-005
sumdiff - prevsumdiff = -5.08615e-008
Probability of good choice for population of 892=48.6646
prob - prevprob = 1.49875e-005
sumdiff - prevsumdiff = -5.05764e-008
Probability of good choice for population of 894=48.6661
prob - prevprob = 1.49372e-005
sumdiff - prevsumdiff = -5.02935e-008
Probability of good choice for population of 896=48.6676
prob - prevprob = 1.48872e-005
sumdiff - prevsumdiff = -5.00129e-008
Probability of good choice for population of 898=48.6691
prob - prevprob = 1.48374e-005
sumdiff - prevsumdiff = -4.97344e-008
Probability of good choice for population of 900=48.6706
prob - prevprob = 1.4788e-005
sumdiff - prevsumdiff = -4.94581e-008
Probability of good choice for population of 902=48.672
prob - prevprob = 1.47388e-005
sumdiff - prevsumdiff = -4.91839e-008
Probability of good choice for population of 904=48.6735
prob - prevprob = 1.46899e-005
sumdiff - prevsumdiff = -4.89119e-008
Probability of good choice for population of 906=48.675
```

```
prob - prevprob = 1.46412e-005
sumdiff - prevsumdiff = -4.8642e-008
Probability of good choice for population of 908=48.6764
prob - prevprob = 1.45929e-005
sumdiff - prevsumdiff = -4.83741e-008
Probability of good choice for population of 910=48.6779
prob - prevprob = 1.45447e-005
sumdiff - prevsumdiff = -4.81083e-008
Probability of good choice for population of 912=48.6793
prob - prevprob = 1.44969e-005
sumdiff - prevsumdiff = -4.78446e-008
Probability of good choice for population of 914=48.6808
prob - prevprob = 1.44493e-005
sumdiff - prevsumdiff = -4.75828e-008
Probability of good choice for population of 916=48.6822
prob - prevprob = 1.4402e-005
sumdiff - prevsumdiff = -4.73231e-008
Probability of good choice for population of 918=48.6837
prob - prevprob = 1.43549e-005
sumdiff - prevsumdiff = -4.70653e-008
Probability of good choice for population of 920=48.6851
prob - prevprob = 1.43081e-005
sumdiff - prevsumdiff = -4.68096e-008
Probability of good choice for population of 922=48.6865
prob - prevprob = 1.42616e-005
sumdiff - prevsumdiff = -4.65557e-008
Probability of good choice for population of 924=48.6879
prob - prevprob = 1.42153e-005
sumdiff - prevsumdiff = -4.63038e-008
Probability of good choice for population of 926=48.6893
prob - prevprob = 1.41692e-005
sumdiff - prevsumdiff = -4.60538e-008
Probability of good choice for population of 928=48.6908
prob - prevprob = 1.41234e-005
sumdiff - prevsumdiff = -4.58056e-008
Probability of good choice for population of 930=48.6922
prob - prevprob = 1.40778e-005
sumdiff - prevsumdiff = -4.55594e-008
Probability of good choice for population of 932=48.6936
prob - prevprob = 1.40325e-005
sumdiff - prevsumdiff = -4.53149e-008
Probability of good choice for population of 934=48.695
prob - prevprob = 1.39875e-005
sumdiff - prevsumdiff = -4.50724e-008
Probability of good choice for population of 936=48.6964
prob - prevprob = 1.39426e-005
sumdiff - prevsumdiff = -4.48316e-008
Probability of good choice for population of 938=48.6978
```

```
prob - prevprob = 1.3898e-005
sumdiff - prevsumdiff = -4.45926e-008
Probability of good choice for population of 940=48.6991
prob - prevprob = 1.38537e-005
sumdiff - prevsumdiff = -4.43554e-008
Probability of good choice for population of 942=48.7005
prob - prevprob = 1.38096e-005
sumdiff - prevsumdiff = -4.412e-008
Probability of good choice for population of 944=48.7019
prob - prevprob = 1.37657e-005
sumdiff - prevsumdiff = -4.38863e-008
Probability of good choice for population of 946=48.7033
prob - prevprob = 1.3722e-005
sumdiff - prevsumdiff = -4.36543e-008
Probability of good choice for population of 948=48.7046
prob - prevprob = 1.36786e-005
sumdiff - prevsumdiff = -4.34241e-008
Probability of good choice for population of 950=48.706
prob - prevprob = 1.36354e-005
sumdiff - prevsumdiff = -4.31956e-008
Probability of good choice for population of 952=48.7074
prob - prevprob = 1.35924e-005
sumdiff - prevsumdiff = -4.29687e-008
Probability of good choice for population of 954=48.7087
prob - prevprob = 1.35497e-005
sumdiff - prevsumdiff = -4.27435e-008
Probability of good choice for population of 956=48.7101
prob - prevprob = 1.35072e-005
sumdiff - prevsumdiff = -4.25199e-008
Probability of good choice for population of 958=48.7114
prob - prevprob = 1.34649e-005
sumdiff - prevsumdiff = -4.2298e-008
Probability of good choice for population of 960=48.7128
prob - prevprob = 1.34228e-005
sumdiff - prevsumdiff = -4.20777e-008
Probability of good choice for population of 962=48.7141
prob - prevprob = 1.33809e-005
sumdiff - prevsumdiff = -4.1859e-008
Probability of good choice for population of 964=48.7154
prob - prevprob = 1.33393e-005
sumdiff - prevsumdiff = -4.16419e-008
Probability of good choice for population of 966=48.7168
prob - prevprob = 1.32979e-005
sumdiff - prevsumdiff = -4.14264e-008
Probability of good choice for population of 968=48.7181
prob - prevprob = 1.32566e-005
sumdiff - prevsumdiff = -4.12124e-008
Probability of good choice for population of 970=48.7194
```

```
prob - prevprob = 1.32156e-005
sumdiff - prevsumdiff = -4.09999e-008
Probability of good choice for population of 972=48.7207
prob - prevprob = 1.31749e-005
sumdiff - prevsumdiff = -4.0789e-008
Probability of good choice for population of 974=48.722
prob - prevprob = 1.31343e-005
sumdiff - prevsumdiff = -4.05797e-008
Probability of good choice for population of 976=48.7233
prob - prevprob = 1.30939e-005
sumdiff - prevsumdiff = -4.03718e-008
Probability of good choice for population of 978=48.7246
prob - prevprob = 1.30537e-005
sumdiff - prevsumdiff = -4.01654e-008
Probability of good choice for population of 980=48.726
prob - prevprob = 1.30138e-005
sumdiff - prevsumdiff = -3.99604e-008
Probability of good choice for population of 982=48.7272
prob - prevprob = 1.2974e-005
sumdiff - prevsumdiff = -3.9757e-008
Probability of good choice for population of 984=48.7285
prob - prevprob = 1.29345e-005
sumdiff - prevsumdiff = -3.9555e-008
Probability of good choice for population of 986=48.7298
prob - prevprob = 1.28951e-005
sumdiff - prevsumdiff = -3.93544e-008
Probability of good choice for population of 988=48.7311
prob - prevprob = 1.2856e-005
sumdiff - prevsumdiff = -3.91552e-008
Probability of good choice for population of 990=48.7324
prob - prevprob = 1.2817e-005
sumdiff - prevsumdiff = -3.89575e-008
Probability of good choice for population of 992=48.7337
prob - prevprob = 1.27782e-005
sumdiff - prevsumdiff = -3.87611e-008
Probability of good choice for population of 994=48.735
prob - prevprob = 1.27397e-005
sumdiff - prevsumdiff = -3.85661e-008
Probability of good choice for population of 996=48.7362
prob - prevprob = 1.27013e-005
sumdiff - prevsumdiff = -3.83725e-008
Probability of good choice for population of 998=48.7375
prob - prevprob = 1.26631e-005
sumdiff - prevsumdiff = -3.81803e-008
Probability of good choice for population of 1000=48.7387
prob - prevprob = 1.26251e-005
sumdiff - prevsumdiff = -3.79894e-008
Probability of good choice for population of 1002=48.74
```

```
prob - prevprob = 1.25873e-005
sumdiff - prevsumdiff = -3.77998e-008
Probability of good choice for population of 1004=48.7413
prob - prevprob = 1.25497e-005
sumdiff - prevsumdiff = -3.76116e-008
Probability of good choice for population of 1006=48.7425
prob - prevprob = 1.25123e-005
sumdiff - prevsumdiff = -3.74246e-008
Probability of good choice for population of 1008=48.7438
prob - prevprob = 1.24751e-005
sumdiff - prevsumdiff = -3.7239e-008
Probability of good choice for population of 1010=48.745
prob - prevprob = 1.2438e-005
sumdiff - prevsumdiff = -3.70546e-008
Probability of good choice for population of 1012=48.7462
prob - prevprob = 1.24011e-005
sumdiff - prevsumdiff = -3.68716e-008
Probability of good choice for population of 1014=48.7475
prob - prevprob = 1.23644e-005
sumdiff - prevsumdiff = -3.66897e-008
Probability of good choice for population of 1016=48.7487
prob - prevprob = 1.23279e-005
sumdiff - prevsumdiff = -3.65092e-008
Probability of good choice for population of 1018=48.7499
prob - prevprob = 1.22916e-005
sumdiff - prevsumdiff = -3.63299e-008
Probability of good choice for population of 1020=48.7512
prob - prevprob = 1.22555e-005
sumdiff - prevsumdiff = -3.61518e-008
Probability of good choice for population of 1022=48.7524
prob - prevprob = 1.22195e-005
sumdiff - prevsumdiff = -3.59749e-008
Probability of good choice for population of 1024=48.7536
prob - prevprob = 1.21837e-005
sumdiff - prevsumdiff = -3.57993e-008
Probability of good choice for population of 1026=48.7548
prob - prevprob = 1.21481e-005
sumdiff - prevsumdiff = -3.56248e-008
Probability of good choice for population of 1028=48.756
prob - prevprob = 1.21126e-005
sumdiff - prevsumdiff = -3.54515e-008
Probability of good choice for population of 1030=48.7572
prob - prevprob = 1.20773e-005
sumdiff - prevsumdiff = -3.52794e-008
Probability of good choice for population of 1032=48.7584
prob - prevprob = 1.20422e-005
sumdiff - prevsumdiff = -3.51085e-008
Probability of good choice for population of 1034=48.7596
```

```
prob - prevprob = 1.20073e-005
sumdiff - prevsumdiff = -3.49387e-008
Probability of good choice for population of 1036=48.7608
prob - prevprob = 1.19725e-005
sumdiff - prevsumdiff = -3.47701e-008
Probability of good choice for population of 1038=48.762
prob - prevprob = 1.19379e-005
sumdiff - prevsumdiff = -3.46026e-008
Probability of good choice for population of 1040=48.7632
prob - prevprob = 1.19035e-005
sumdiff - prevsumdiff = -3.44363e-008
Probability of good choice for population of 1042=48.7644
prob - prevprob = 1.18692e-005
sumdiff - prevsumdiff = -3.4271e-008
Probability of good choice for population of 1044=48.7656
prob - prevprob = 1.18351e-005
sumdiff - prevsumdiff = -3.41069e-008
Probability of good choice for population of 1046=48.7668
prob - prevprob = 1.18011e-005
sumdiff - prevsumdiff = -3.39438e-008
Probability of good choice for population of 1048=48.768
prob - prevprob = 1.17674e-005
sumdiff - prevsumdiff = -3.37819e-008
Probability of good choice for population of 1050=48.7691
prob - prevprob = 1.17337e-005
sumdiff - prevsumdiff = -3.3621e-008
Probability of good choice for population of 1052=48.7703
prob - prevprob = 1.17003e-005
sumdiff - prevsumdiff = -3.34612e-008
Probability of good choice for population of 1054=48.7715
prob - prevprob = 1.1667e-005
sumdiff - prevsumdiff = -3.33025e-008
Probability of good choice for population of 1056=48.7726
prob - prevprob = 1.16338e-005
sumdiff - prevsumdiff = -3.31448e-008
Probability of good choice for population of 1058=48.7738
prob - prevprob = 1.16008e-005
sumdiff - prevsumdiff = -3.29882e-008
Probability of good choice for population of 1060=48.7749
prob - prevprob = 1.1568e-005
sumdiff - prevsumdiff = -3.28326e-008
Probability of good choice for population of 1062=48.7761
prob - prevprob = 1.15353e-005
sumdiff - prevsumdiff = -3.2678e-008
Probability of good choice for population of 1064=48.7773
prob - prevprob = 1.15028e-005
sumdiff - prevsumdiff = -3.25244e-008
Probability of good choice for population of 1066=48.7784
```

```
prob - prevprob = 1.14704e-005
sumdiff - prevsumdiff = -3.23719e-008
Probability of good choice for population of 1068=48.7795
prob - prevprob = 1.14382e-005
sumdiff - prevsumdiff = -3.22203e-008
Probability of good choice for population of 1070=48.7807
prob - prevprob = 1.14061e-005
sumdiff - prevsumdiff = -3.20698e-008
Probability of good choice for population of 1072=48.7818
prob - prevprob = 1.13742e-005
sumdiff - prevsumdiff = -3.19202e-008
Probability of good choice for population of 1074=48.783
prob - prevprob = 1.13425e-005
sumdiff - prevsumdiff = -3.17716e-008
Probability of good choice for population of 1076=48.7841
prob - prevprob = 1.13108e-005
sumdiff - prevsumdiff = -3.16239e-008
Probability of good choice for population of 1078=48.7852
prob - prevprob = 1.12794e-005
sumdiff - prevsumdiff = -3.14773e-008
Probability of good choice for population of 1080=48.7863
prob - prevprob = 1.1248e-005
sumdiff - prevsumdiff = -3.13315e-008
Probability of good choice for population of 1082=48.7875
prob - prevprob = 1.12168e-005
sumdiff - prevsumdiff = -3.11868e-008
Probability of good choice for population of 1084=48.7886
prob - prevprob = 1.11858e-005
sumdiff - prevsumdiff = -3.10429e-008
Probability of good choice for population of 1086=48.7897
prob - prevprob = 1.11549e-005
sumdiff - prevsumdiff = -3.09e-008
Probability of good choice for population of 1088=48.7908
prob - prevprob = 1.11241e-005
sumdiff - prevsumdiff = -3.0758e-008
Probability of good choice for population of 1090=48.7919
prob - prevprob = 1.10935e-005
sumdiff - prevsumdiff = -3.06169e-008
Probability of good choice for population of 1092=48.793
prob - prevprob = 1.1063e-005
sumdiff - prevsumdiff = -3.04767e-008
Probability of good choice for population of 1094=48.7941
prob - prevprob = 1.10327e-005
sumdiff - prevsumdiff = -3.03374e-008
Probability of good choice for population of 1096=48.7952
prob - prevprob = 1.10025e-005
sumdiff - prevsumdiff = -3.0199e-008
Probability of good choice for population of 1098=48.7963
```

```
prob - prevprob = 1.09724e-005
sumdiff - prevsumdiff = -3.00615e-008
Probability of good choice for population of 1100=48.7974
prob - prevprob = 1.09425e-005
sumdiff - prevsumdiff = -2.99248e-008
Probability of good choice for population of 1102=48.7985
prob - prevprob = 1.09127e-005
sumdiff - prevsumdiff = -2.97891e-008
Probability of good choice for population of 1104=48.7996
prob - prevprob = 1.08831e-005
sumdiff - prevsumdiff = -2.96542e-008
Probability of good choice for population of 1106=48.8007
prob - prevprob = 1.08536e-005
sumdiff - prevsumdiff = -2.95201e-008
Probability of good choice for population of 1108=48.8018
prob - prevprob = 1.08242e-005
sumdiff - prevsumdiff = -2.93869e-008
Probability of good choice for population of 1110=48.8028
prob - prevprob = 1.07949e-005
sumdiff - prevsumdiff = -2.92545e-008
Probability of good choice for population of 1112=48.8039
prob - prevprob = 1.07658e-005
sumdiff - prevsumdiff = -2.9123e-008
Probability of good choice for population of 1114=48.805
prob - prevprob = 1.07368e-005
sumdiff - prevsumdiff = -2.89923e-008
Probability of good choice for population of 1116=48.8061
prob - prevprob = 1.07079e-005
sumdiff - prevsumdiff = -2.88624e-008
Probability of good choice for population of 1118=48.8071
prob - prevprob = 1.06792e-005
sumdiff - prevsumdiff = -2.87333e-008
Probability of good choice for population of 1120=48.8082
prob - prevprob = 1.06506e-005
sumdiff - prevsumdiff = -2.8605e-008
Probability of good choice for population of 1122=48.8093
prob - prevprob = 1.06221e-005
sumdiff - prevsumdiff = -2.84775e-008
Probability of good choice for population of 1124=48.8103
prob - prevprob = 1.05938e-005
sumdiff - prevsumdiff = -2.83509e-008
Probability of good choice for population of 1126=48.8114
prob - prevprob = 1.05655e-005
sumdiff - prevsumdiff = -2.8225e-008
Probability of good choice for population of 1128=48.8124
prob - prevprob = 1.05374e-005
sumdiff - prevsumdiff = -2.80999e-008
Probability of good choice for population of 1130=48.8135
```

```
prob - prevprob = 1.05095e-005
sumdiff - prevsumdiff = -2.79755e-008
Probability of good choice for population of 1132=48.8145
prob - prevprob = 1.04816e-005
sumdiff - prevsumdiff = -2.7852e-008
Probability of good choice for population of 1134=48.8156
prob - prevprob = 1.04539e-005
sumdiff - prevsumdiff = -2.77291e-008
Probability of good choice for population of 1136=48.8166
prob - prevprob = 1.04263e-005
sumdiff - prevsumdiff = -2.76071e-008
Probability of good choice for population of 1138=48.8177
prob - prevprob = 1.03988e-005
sumdiff - prevsumdiff = -2.74858e-008
Probability of good choice for population of 1140=48.8187
prob - prevprob = 1.03714e-005
sumdiff - prevsumdiff = -2.73653e-008
Probability of good choice for population of 1142=48.8197
prob - prevprob = 1.03442e-005
sumdiff - prevsumdiff = -2.72454e-008
Probability of good choice for population of 1144=48.8208
prob - prevprob = 1.03171e-005
sumdiff - prevsumdiff = -2.71264e-008
Probability of good choice for population of 1146=48.8218
prob - prevprob = 1.02901e-005
sumdiff - prevsumdiff = -2.7008e-008
Probability of good choice for population of 1148=48.8228
prob - prevprob = 1.02632e-005
sumdiff - prevsumdiff = -2.68904e-008
Probability of good choice for population of 1150=48.8238
prob - prevprob = 1.02364e-005
sumdiff - prevsumdiff = -2.67735e-008
Probability of good choice for population of 1152=48.8249
prob - prevprob = 1.02097e-005
sumdiff - prevsumdiff = -2.66573e-008
Probability of good choice for population of 1154=48.8259
prob - prevprob = 1.01832e-005
sumdiff - prevsumdiff = -2.65418e-008
Probability of good choice for population of 1156=48.8269
prob - prevprob = 1.01568e-005
sumdiff - prevsumdiff = -2.6427e-008
Probability of good choice for population of 1158=48.8279
prob - prevprob = 1.01304e-005
sumdiff - prevsumdiff = -2.63129e-008
Probability of good choice for population of 1160=48.8289
prob - prevprob = 1.01042e-005
sumdiff - prevsumdiff = -2.61994e-008
Probability of good choice for population of 1162=48.8299
```

```
prob - prevprob = 1.00782e-005
sumdiff - prevsumdiff = -2.60867e-008
Probability of good choice for population of 1164=48.8309
prob - prevprob = 1.00522e-005
sumdiff - prevsumdiff = -2.59746e-008
Probability of good choice for population of 1166=48.8319
prob - prevprob = 1.00263e-005
sumdiff - prevsumdiff = -2.58633e-008
Probability of good choice for population of 1168=48.8329
prob - prevprob = 1.00006e-005
sumdiff - prevsumdiff = -2.57525e-008
Probability of good choice for population of 1170=48.8339
prob - prevprob = 9.97493e-006
sumdiff - prevsumdiff = -2.56425e-008
Probability of good choice for population of 1172=48.8349
prob - prevprob = 9.9494e-006
sumdiff - prevsumdiff = -2.55331e-008
Probability of good choice for population of 1174=48.8359
prob - prevprob = 9.92397e-006
sumdiff - prevsumdiff = -2.54244e-008
Probability of good choice for population of 1176=48.8369
prob - prevprob = 9.89866e-006
sumdiff - prevsumdiff = -2.53163e-008
Probability of good choice for population of 1178=48.8379
prob - prevprob = 9.87345e-006
sumdiff - prevsumdiff = -2.52088e-008
Probability of good choice for population of 1180=48.8389
prob - prevprob = 9.84834e-006
sumdiff - prevsumdiff = -2.5102e-008
Probability of good choice for population of 1182=48.8399
prob - prevprob = 9.82335e-006
sumdiff - prevsumdiff = -2.49958e-008
Probability of good choice for population of 1184=48.8408
prob - prevprob = 9.79846e-006
sumdiff - prevsumdiff = -2.48902e-008
Probability of good choice for population of 1186=48.8418
prob - prevprob = 9.77367e-006
sumdiff - prevsumdiff = -2.47853e-008
Probability of good choice for population of 1188=48.8428
prob - prevprob = 9.74899e-006
sumdiff - prevsumdiff = -2.4681e-008
Probability of good choice for population of 1190=48.8438
prob - prevprob = 9.72442e-006
sumdiff - prevsumdiff = -2.45773e-008
Probability of good choice for population of 1192=48.8447
prob - prevprob = 9.69994e-006
sumdiff - prevsumdiff = -2.44742e-008
Probability of good choice for population of 1194=48.8457
```

```
prob - prevprob = 9.67557e-006
sumdiff - prevsumdiff = -2.43717e-008
Probability of good choice for population of 1196=48.8467
prob - prevprob = 9.6513e-006
sumdiff - prevsumdiff = -2.42698e-008
Probability of good choice for population of 1198=48.8476
prob - prevprob = 9.62713e-006
sumdiff - prevsumdiff = -2.41685e-008
Probability of good choice for population of 1200=48.8486
prob - prevprob = 9.60306e-006
sumdiff - prevsumdiff = -2.40678e-008
Probability of good choice for population of 1202=48.8496
prob - prevprob = 9.5791e-006
sumdiff - prevsumdiff = -2.39677e-008
Probability of good choice for population of 1204=48.8505
prob - prevprob = 9.55523e-006
sumdiff - prevsumdiff = -2.38682e-008
Probability of good choice for population of 1206=48.8515
prob - prevprob = 9.53146e-006
sumdiff - prevsumdiff = -2.37692e-008
Probability of good choice for population of 1208=48.8524
prob - prevprob = 9.50779e-006
sumdiff - prevsumdiff = -2.36708e-008
Probability of good choice for population of 1210=48.8534
prob - prevprob = 9.48421e-006
sumdiff - prevsumdiff = -2.3573e-008
Probability of good choice for population of 1212=48.8543
prob - prevprob = 9.46074e-006
sumdiff - prevsumdiff = -2.34758e-008
Probability of good choice for population of 1214=48.8552
prob - prevprob = 9.43736e-006
sumdiff - prevsumdiff = -2.33791e-008
Probability of good choice for population of 1216=48.8562
prob - prevprob = 9.41408e-006
sumdiff - prevsumdiff = -2.3283e-008
Probability of good choice for population of 1218=48.8571
prob - prevprob = 9.39089e-006
sumdiff - prevsumdiff = -2.31874e-008
Probability of good choice for population of 1220=48.8581
prob - prevprob = 9.3678e-006
sumdiff - prevsumdiff = -2.30923e-008
Probability of good choice for population of 1222=48.859
prob - prevprob = 9.3448e-006
sumdiff - prevsumdiff = -2.29979e-008
Probability of good choice for population of 1224=48.8599
prob - prevprob = 9.32189e-006
sumdiff - prevsumdiff = -2.29039e-008
Probability of good choice for population of 1226=48.8609
```

```
prob - prevprob = 9.29908e-006
sumdiff - prevsumdiff = -2.28105e-008
Probability of good choice for population of 1228=48.8618
prob - prevprob = 9.27637e-006
sumdiff - prevsumdiff = -2.27176e-008
Probability of good choice for population of 1230=48.8627
prob - prevprob = 9.25374e-006
sumdiff - prevsumdiff = -2.26253e-008
Probability of good choice for population of 1232=48.8636
prob - prevprob = 9.23121e-006
sumdiff - prevsumdiff = -2.25335e-008
Probability of good choice for population of 1234=48.8646
prob - prevprob = 9.20877e-006
sumdiff - prevsumdiff = -2.24422e-008
Probability of good choice for population of 1236=48.8655
prob - prevprob = 9.18641e-006
sumdiff - prevsumdiff = -2.23514e-008
Probability of good choice for population of 1238=48.8664
prob - prevprob = 9.16415e-006
sumdiff - prevsumdiff = -2.22611e-008
Probability of good choice for population of 1240=48.8673
prob - prevprob = 9.14198e-006
sumdiff - prevsumdiff = -2.21713e-008
Probability of good choice for population of 1242=48.8682
prob - prevprob = 9.1199e-006
sumdiff - prevsumdiff = -2.20821e-008
Probability of good choice for population of 1244=48.8691
prob - prevprob = 9.09791e-006
sumdiff - prevsumdiff = -2.19933e-008
Probability of good choice for population of 1246=48.87
prob - prevprob = 9.076e-006
sumdiff - prevsumdiff = -2.19051e-008
Probability of good choice for population of 1248=48.8709
prob - prevprob = 9.05418e-006
sumdiff - prevsumdiff = -2.18173e-008
Probability of good choice for population of 1250=48.8718
prob - prevprob = 9.03245e-006
sumdiff - prevsumdiff = -2.173e-008
Probability of good choice for population of 1252=48.8727
prob - prevprob = 9.01081e-006
sumdiff - prevsumdiff = -2.16433e-008
Probability of good choice for population of 1254=48.8736
prob - prevprob = 8.98925e-006
sumdiff - prevsumdiff = -2.1557e-008
Probability of good choice for population of 1256=48.8745
prob - prevprob = 8.96778e-006
sumdiff - prevsumdiff = -2.14711e-008
Probability of good choice for population of 1258=48.8754
```

```
prob - prevprob = 8.9464e-006
sumdiff - prevsumdiff = -2.13858e-008
Probability of good choice for population of 1260=48.8763
prob - prevprob = 8.9251e-006
sumdiff - prevsumdiff = -2.13009e-008
Probability of good choice for population of 1262=48.8772
prob - prevprob = 8.90388e-006
sumdiff - prevsumdiff = -2.12166e-008
Probability of good choice for population of 1264=48.8781
prob - prevprob = 8.88275e-006
sumdiff - prevsumdiff = -2.11326e-008
Probability of good choice for population of 1266=48.879
prob - prevprob = 8.8617e-006
sumdiff - prevsumdiff = -2.10492e-008
Probability of good choice for population of 1268=48.8799
prob - prevprob = 8.84073e-006
sumdiff - prevsumdiff = -2.09662e-008
Probability of good choice for population of 1270=48.8808
prob - prevprob = 8.81985e-006
sumdiff - prevsumdiff = -2.08836e-008
Probability of good choice for population of 1272=48.8816
prob - prevprob = 8.79905e-006
sumdiff - prevsumdiff = -2.08015e-008
Probability of good choice for population of 1274=48.8825
prob - prevprob = 8.77833e-006
sumdiff - prevsumdiff = -2.07199e-008
Probability of good choice for population of 1276=48.8834
prob - prevprob = 8.75769e-006
sumdiff - prevsumdiff = -2.06387e-008
Probability of good choice for population of 1278=48.8843
prob - prevprob = 8.73713e-006
sumdiff - prevsumdiff = -2.0558e-008
Probability of good choice for population of 1280=48.8851
prob - prevprob = 8.71665e-006
sumdiff - prevsumdiff = -2.04776e-008
Probability of good choice for population of 1282=48.886
prob - prevprob = 8.69625e-006
sumdiff - prevsumdiff = -2.03978e-008
Probability of good choice for population of 1284=48.8869
prob - prevprob = 8.67594e-006
sumdiff - prevsumdiff = -2.03184e-008
Probability of good choice for population of 1286=48.8877
prob - prevprob = 8.6557e-006
sumdiff - prevsumdiff = -2.02394e-008
Probability of good choice for population of 1288=48.8886
prob - prevprob = 8.63554e-006
sumdiff - prevsumdiff = -2.01608e-008
Probability of good choice for population of 1290=48.8895
```

```
prob - prevprob = 8.61545e-006
sumdiff - prevsumdiff = -2.00826e-008
Probability of good choice for population of 1292=48.8903
prob - prevprob = 8.59545e-006
sumdiff - prevsumdiff = -2.00049e-008
Probability of good choice for population of 1294=48.8912
prob - prevprob = 8.57552e-006
sumdiff - prevsumdiff = -1.99276e-008
Probability of good choice for population of 1296=48.892
prob - prevprob = 8.55567e-006
sumdiff - prevsumdiff = -1.98507e-008
Probability of good choice for population of 1298=48.8929
prob - prevprob = 8.5359e-006
sumdiff - prevsumdiff = -1.97743e-008
Probability of good choice for population of 1300=48.8937
prob - prevprob = 8.5162e-006
sumdiff - prevsumdiff = -1.96982e-008
Probability of good choice for population of 1302=48.8946
prob - prevprob = 8.49657e-006
sumdiff - prevsumdiff = -1.96226e-008
Probability of good choice for population of 1304=48.8954
prob - prevprob = 8.47703e-006
sumdiff - prevsumdiff = -1.95473e-008
Probability of good choice for population of 1306=48.8963
prob - prevprob = 8.45756e-006
sumdiff - prevsumdiff = -1.94725e-008
Probability of good choice for population of 1308=48.8971
prob - prevprob = 8.43816e-006
sumdiff - prevsumdiff = -1.93981e-008
Probability of good choice for population of 1310=48.898
prob - prevprob = 8.41883e-006
sumdiff - prevsumdiff = -1.9324e-008
Probability of good choice for population of 1312=48.8988
prob - prevprob = 8.39958e-006
sumdiff - prevsumdiff = -1.92504e-008
Probability of good choice for population of 1314=48.8997
prob - prevprob = 8.38041e-006
sumdiff - prevsumdiff = -1.91771e-008
Probability of good choice for population of 1316=48.9005
prob - prevprob = 8.3613e-006
sumdiff - prevsumdiff = -1.91043e-008
Probability of good choice for population of 1318=48.9013
prob - prevprob = 8.34227e-006
sumdiff - prevsumdiff = -1.90318e-008
Probability of good choice for population of 1320=48.9022
prob - prevprob = 8.32331e-006
sumdiff - prevsumdiff = -1.89597e-008
Probability of good choice for population of 1322=48.903
```

```
prob - prevprob = 8.30442e-006
sumdiff - prevsumdiff = -1.8888e-008
Probability of good choice for population of 1324=48.9038
prob - prevprob = 8.2856e-006
sumdiff - prevsumdiff = -1.88167e-008
Probability of good choice for population of 1326=48.9046
prob - prevprob = 8.26686e-006
sumdiff - prevsumdiff = -1.87457e-008
Probability of good choice for population of 1328=48.9055
prob - prevprob = 8.24818e-006
sumdiff - prevsumdiff = -1.86751e-008
Probability of good choice for population of 1330=48.9063
prob - prevprob = 8.22958e-006
sumdiff - prevsumdiff = -1.86049e-008
Probability of good choice for population of 1332=48.9071
prob - prevprob = 8.21104e-006
sumdiff - prevsumdiff = -1.85351e-008
Probability of good choice for population of 1334=48.9079
prob - prevprob = 8.19258e-006
sumdiff - prevsumdiff = -1.84656e-008
Probability of good choice for population of 1336=48.9087
prob - prevprob = 8.17418e-006
sumdiff - prevsumdiff = -1.83965e-008
Probability of good choice for population of 1338=48.9096
prob - prevprob = 8.15585e-006
sumdiff - prevsumdiff = -1.83278e-008
Probability of good choice for population of 1340=48.9104
prob - prevprob = 8.13759e-006
sumdiff - prevsumdiff = -1.82594e-008
Probability of good choice for population of 1342=48.9112
prob - prevprob = 8.1194e-006
sumdiff - prevsumdiff = -1.81913e-008
Probability of good choice for population of 1344=48.912
prob - prevprob = 8.10128e-006
sumdiff - prevsumdiff = -1.81237e-008
Probability of good choice for population of 1346=48.9128
prob - prevprob = 8.08322e-006
sumdiff - prevsumdiff = -1.80563e-008
Probability of good choice for population of 1348=48.9136
prob - prevprob = 8.06523e-006
sumdiff - prevsumdiff = -1.79894e-008
Probability of good choice for population of 1350=48.9144
prob - prevprob = 8.04731e-006
sumdiff - prevsumdiff = -1.79227e-008
Probability of good choice for population of 1352=48.9152
prob - prevprob = 8.02945e-006
sumdiff - prevsumdiff = -1.78565e-008
Probability of good choice for population of 1354=48.916
```

```
prob - prevprob = 8.01166e-006
sumdiff - prevsumdiff = -1.77905e-008
Probability of good choice for population of 1356=48.9168
prob - prevprob = 7.99394e-006
sumdiff - prevsumdiff = -1.77249e-008
Probability of good choice for population of 1358=48.9176
prob - prevprob = 7.97628e-006
sumdiff - prevsumdiff = -1.76597e-008
Probability of good choice for population of 1360=48.9184
prob - prevprob = 7.95869e-006
sumdiff - prevsumdiff = -1.75947e-008
Probability of good choice for population of 1362=48.9192
prob - prevprob = 7.94115e-006
sumdiff - prevsumdiff = -1.75301e-008
Probability of good choice for population of 1364=48.92
prob - prevprob = 7.92369e-006
sumdiff - prevsumdiff = -1.74659e-008
Probability of good choice for population of 1366=48.9208
prob - prevprob = 7.90629e-006
sumdiff - prevsumdiff = -1.7402e-008
Probability of good choice for population of 1368=48.9216
prob - prevprob = 7.88895e-006
sumdiff - prevsumdiff = -1.73383e-008
Probability of good choice for population of 1370=48.9224
prob - prevprob = 7.87167e-006
sumdiff - prevsumdiff = -1.72751e-008
Probability of good choice for population of 1372=48.9232
prob - prevprob = 7.85446e-006
sumdiff - prevsumdiff = -1.72121e-008
Probability of good choice for population of 1374=48.9239
prob - prevprob = 7.83731e-006
sumdiff - prevsumdiff = -1.71495e-008
Probability of good choice for population of 1376=48.9247
prob - prevprob = 7.82022e-006
sumdiff - prevsumdiff = -1.70872e-008
Probability of good choice for population of 1378=48.9255
prob - prevprob = 7.8032e-006
sumdiff - prevsumdiff = -1.70252e-008
Probability of good choice for population of 1380=48.9263
prob - prevprob = 7.78624e-006
sumdiff - prevsumdiff = -1.69635e-008
Probability of good choice for population of 1382=48.9271
prob - prevprob = 7.76933e-006
sumdiff - prevsumdiff = -1.69021e-008
Probability of good choice for population of 1384=48.9278
prob - prevprob = 7.75249e-006
sumdiff - prevsumdiff = -1.6841e-008
Probability of good choice for population of 1386=48.9286
```

```
prob - prevprob = 7.73571e-006
sumdiff - prevsumdiff = -1.67803e-008
Probability of good choice for population of 1388=48.9294
prob - prevprob = 7.71899e-006
sumdiff - prevsumdiff = -1.67198e-008
Probability of good choice for population of 1390=48.9301
prob - prevprob = 7.70233e-006
sumdiff - prevsumdiff = -1.66597e-008
Probability of good choice for population of 1392=48.9309
prob - prevprob = 7.68573e-006
sumdiff - prevsumdiff = -1.65999e-008
Probability of good choice for population of 1394=48.9317
prob - prevprob = 7.66919e-006
sumdiff - prevsumdiff = -1.65403e-008
Probability of good choice for population of 1396=48.9324
prob - prevprob = 7.65271e-006
sumdiff - prevsumdiff = -1.64811e-008
Probability of good choice for population of 1398=48.9332
prob - prevprob = 7.63629e-006
sumdiff - prevsumdiff = -1.64221e-008
Probability of good choice for population of 1400=48.934
prob - prevprob = 7.61993e-006
sumdiff - prevsumdiff = -1.63635e-008
Probability of good choice for population of 1402=48.9347
prob - prevprob = 7.60362e-006
sumdiff - prevsumdiff = -1.63051e-008
Probability of good choice for population of 1404=48.9355
prob - prevprob = 7.58737e-006
sumdiff - prevsumdiff = -1.62471e-008
Probability of good choice for population of 1406=48.9362
prob - prevprob = 7.57119e-006
sumdiff - prevsumdiff = -1.61893e-008
Probability of good choice for population of 1408=48.937
prob - prevprob = 7.55505e-006
sumdiff - prevsumdiff = -1.61318e-008
Probability of good choice for population of 1410=48.9378
prob - prevprob = 7.53898e-006
sumdiff - prevsumdiff = -1.60746e-008
Probability of good choice for population of 1412=48.9385
prob - prevprob = 7.52296e-006
sumdiff - prevsumdiff = -1.60177e-008
Probability of good choice for population of 1414=48.9393
prob - prevprob = 7.507e-006
sumdiff - prevsumdiff = -1.5961e-008
Probability of good choice for population of 1416=48.94
prob - prevprob = 7.4911e-006
sumdiff - prevsumdiff = -1.59047e-008
Probability of good choice for population of 1418=48.9408
```

```
prob - prevprob = 7.47525e-006
sumdiff - prevsumdiff = -1.58486e-008
Probability of good choice for population of 1420=48.9415
prob - prevprob = 7.45945e-006
sumdiff - prevsumdiff = -1.57928e-008
Probability of good choice for population of 1422=48.9422
prob - prevprob = 7.44372e-006
sumdiff - prevsumdiff = -1.57372e-008
Probability of good choice for population of 1424=48.943
prob - prevprob = 7.42803e-006
sumdiff - prevsumdiff = -1.5682e-008
Probability of good choice for population of 1426=48.9437
prob - prevprob = 7.41241e-006
sumdiff - prevsumdiff = -1.5627e-008
Probability of good choice for population of 1428=48.9445
prob - prevprob = 7.39684e-006
sumdiff - prevsumdiff = -1.55723e-008
Probability of good choice for population of 1430=48.9452
prob - prevprob = 7.38132e-006
sumdiff - prevsumdiff = -1.55178e-008
Probability of good choice for population of 1432=48.9459
prob - prevprob = 7.36585e-006
sumdiff - prevsumdiff = -1.54637e-008
Probability of good choice for population of 1434=48.9467
prob - prevprob = 7.35044e-006
sumdiff - prevsumdiff = -1.54097e-008
Probability of good choice for population of 1436=48.9474
prob - prevprob = 7.33509e-006
sumdiff - prevsumdiff = -1.53561e-008
Probability of good choice for population of 1438=48.9481
prob - prevprob = 7.31979e-006
sumdiff - prevsumdiff = -1.53027e-008
Probability of good choice for population of 1440=48.9489
prob - prevprob = 7.30454e-006
sumdiff - prevsumdiff = -1.52496e-008
Probability of good choice for population of 1442=48.9496
prob - prevprob = 7.28934e-006
sumdiff - prevsumdiff = -1.51967e-008
Probability of good choice for population of 1444=48.9503
prob - prevprob = 7.2742e-006
sumdiff - prevsumdiff = -1.51441e-008
Probability of good choice for population of 1446=48.9511
prob - prevprob = 7.2591e-006
sumdiff - prevsumdiff = -1.50917e-008
Probability of good choice for population of 1448=48.9518
prob - prevprob = 7.24406e-006
sumdiff - prevsumdiff = -1.50396e-008
Probability of good choice for population of 1450=48.9525
```

```
prob - prevprob = 7.22908e-006
sumdiff - prevsumdiff = -1.49877e-008
Probability of good choice for population of 1452=48.9532
prob - prevprob = 7.21414e-006
sumdiff - prevsumdiff = -1.49361e-008
Probability of good choice for population of 1454=48.9539
prob - prevprob = 7.19926e-006
sumdiff - prevsumdiff = -1.48847e-008
Probability of good choice for population of 1456=48.9547
prob - prevprob = 7.18442e-006
sumdiff - prevsumdiff = -1.48336e-008
Probability of good choice for population of 1458=48.9554
prob - prevprob = 7.16964e-006
sumdiff - prevsumdiff = -1.47828e-008
Probability of good choice for population of 1460=48.9561
prob - prevprob = 7.15491e-006
sumdiff - prevsumdiff = -1.47321e-008
Probability of good choice for population of 1462=48.9568
prob - prevprob = 7.14023e-006
sumdiff - prevsumdiff = -1.46818e-008
Probability of good choice for population of 1464=48.9575
prob - prevprob = 7.12559e-006
sumdiff - prevsumdiff = -1.46316e-008
Probability of good choice for population of 1466=48.9582
prob - prevprob = 7.11101e-006
sumdiff - prevsumdiff = -1.45817e-008
Probability of good choice for population of 1468=48.9589
prob - prevprob = 7.09648e-006
sumdiff - prevsumdiff = -1.4532e-008
Probability of good choice for population of 1470=48.9597
prob - prevprob = 7.082e-006
sumdiff - prevsumdiff = -1.44826e-008
Probability of good choice for population of 1472=48.9604
prob - prevprob = 7.06756e-006
sumdiff - prevsumdiff = -1.44334e-008
Probability of good choice for population of 1474=48.9611
prob - prevprob = 7.05318e-006
sumdiff - prevsumdiff = -1.43845e-008
Probability of good choice for population of 1476=48.9618
prob - prevprob = 7.03884e-006
sumdiff - prevsumdiff = -1.43357e-008
Probability of good choice for population of 1478=48.9625
prob - prevprob = 7.02456e-006
sumdiff - prevsumdiff = -1.42872e-008
Probability of good choice for population of 1480=48.9632
prob - prevprob = 7.01032e-006
sumdiff - prevsumdiff = -1.4239e-008
Probability of good choice for population of 1482=48.9639
```

```
prob - prevprob = 6.99613e-006
sumdiff - prevsumdiff = -1.41909e-008
Probability of good choice for population of 1484=48.9646
prob - prevprob = 6.98198e-006
sumdiff - prevsumdiff = -1.41431e-008
Probability of good choice for population of 1486=48.9653
prob - prevprob = 6.96789e-006
sumdiff - prevsumdiff = -1.40955e-008
Probability of good choice for population of 1488=48.966
prob - prevprob = 6.95384e-006
sumdiff - prevsumdiff = -1.40482e-008
Probability of good choice for population of 1490=48.9667
prob - prevprob = 6.93984e-006
sumdiff - prevsumdiff = -1.4001e-008
Probability of good choice for population of 1492=48.9674
prob - prevprob = 6.92588e-006
sumdiff - prevsumdiff = -1.39541e-008
Probability of good choice for population of 1494=48.968
prob - prevprob = 6.91198e-006
sumdiff - prevsumdiff = -1.39074e-008
Probability of good choice for population of 1496=48.9687
prob - prevprob = 6.89812e-006
sumdiff - prevsumdiff = -1.38609e-008
Probability of good choice for population of 1498=48.9694
prob - prevprob = 6.8843e-006
sumdiff - prevsumdiff = -1.38147e-008
Probability of good choice for population of 1500=48.9701
prob - prevprob = 6.87053e-006
sumdiff - prevsumdiff = -1.37686e-008
Probability of good choice for population of 1502=48.9708
prob - prevprob = 6.85681e-006
sumdiff - prevsumdiff = -1.37228e-008
Probability of good choice for population of 1504=48.9715
prob - prevprob = 6.84313e-006
sumdiff - prevsumdiff = -1.36771e-008
Probability of good choice for population of 1506=48.9722
prob - prevprob = 6.8295e-006
sumdiff - prevsumdiff = -1.36317e-008
Probability of good choice for population of 1508=48.9728
prob - prevprob = 6.81591e-006
sumdiff - prevsumdiff = -1.35865e-008
Probability of good choice for population of 1510=48.9735
prob - prevprob = 6.80237e-006
sumdiff - prevsumdiff = -1.35416e-008
Probability of good choice for population of 1512=48.9742
prob - prevprob = 6.78888e-006
sumdiff - prevsumdiff = -1.34968e-008
Probability of good choice for population of 1514=48.9749
```

```
prob - prevprob = 6.77542e-006
sumdiff - prevsumdiff = -1.34522e-008
Probability of good choice for population of 1516=48.9756
prob - prevprob = 6.76202e-006
sumdiff - prevsumdiff = -1.34078e-008
Probability of good choice for population of 1518=48.9762
prob - prevprob = 6.74865e-006
sumdiff - prevsumdiff = -1.33637e-008
Probability of good choice for population of 1520=48.9769
prob - prevprob = 6.73533e-006
sumdiff - prevsumdiff = -1.33197e-008
Probability of good choice for population of 1522=48.9776
prob - prevprob = 6.72206e-006
sumdiff - prevsumdiff = -1.3276e-008
Probability of good choice for population of 1524=48.9782
prob - prevprob = 6.70882e-006
sumdiff - prevsumdiff = -1.32324e-008
Probability of good choice for population of 1526=48.9789
prob - prevprob = 6.69564e-006
sumdiff - prevsumdiff = -1.3189e-008
Probability of good choice for population of 1528=48.9796
prob - prevprob = 6.68249e-006
sumdiff - prevsumdiff = -1.31459e-008
Probability of good choice for population of 1530=48.9803
prob - prevprob = 6.66939e-006
sumdiff - prevsumdiff = -1.31029e-008
Probability of good choice for population of 1532=48.9809
prob - prevprob = 6.65633e-006
sumdiff - prevsumdiff = -1.30602e-008
Probability of good choice for population of 1534=48.9816
prob - prevprob = 6.64331e-006
sumdiff - prevsumdiff = -1.30176e-008
Probability of good choice for population of 1536=48.9822
prob - prevprob = 6.63033e-006
sumdiff - prevsumdiff = -1.29752e-008
Probability of good choice for population of 1538=48.9829
prob - prevprob = 6.6174e-006
sumdiff - prevsumdiff = -1.2933e-008
Probability of good choice for population of 1540=48.9836
prob - prevprob = 6.60451e-006
sumdiff - prevsumdiff = -1.2891e-008
Probability of good choice for population of 1542=48.9842
prob - prevprob = 6.59166e-006
sumdiff - prevsumdiff = -1.28492e-008
Probability of good choice for population of 1544=48.9849
prob - prevprob = 6.57885e-006
sumdiff - prevsumdiff = -1.28076e-008
Probability of good choice for population of 1546=48.9855
```

```
prob - prevprob = 6.56609e-006
sumdiff - prevsumdiff = -1.27662e-008
Probability of good choice for population of 1548=48.9862
prob - prevprob = 6.55336e-006
sumdiff - prevsumdiff = -1.2725e-008
Probability of good choice for population of 1550=48.9868
prob - prevprob = 6.54068e-006
sumdiff - prevsumdiff = -1.26839e-008
Probability of good choice for population of 1552=48.9875
prob - prevprob = 6.52803e-006
sumdiff - prevsumdiff = -1.26431e-008
Probability of good choice for population of 1554=48.9882
prob - prevprob = 6.51543e-006
sumdiff - prevsumdiff = -1.26024e-008
Probability of good choice for population of 1556=48.9888
prob - prevprob = 6.50287e-006
sumdiff - prevsumdiff = -1.25619e-008
Probability of good choice for population of 1558=48.9895
prob - prevprob = 6.49035e-006
sumdiff - prevsumdiff = -1.25216e-008
Probability of good choice for population of 1560=48.9901
prob - prevprob = 6.47787e-006
sumdiff - prevsumdiff = -1.24814e-008
Probability of good choice for population of 1562=48.9907
prob - prevprob = 6.46543e-006
sumdiff - prevsumdiff = -1.24415e-008
Probability of good choice for population of 1564=48.9914
prob - prevprob = 6.45302e-006
sumdiff - prevsumdiff = -1.24017e-008
Probability of good choice for population of 1566=48.992
prob - prevprob = 6.44066e-006
sumdiff - prevsumdiff = -1.23621e-008
Probability of good choice for population of 1568=48.9927
prob - prevprob = 6.42834e-006
sumdiff - prevsumdiff = -1.23227e-008
Probability of good choice for population of 1570=48.9933
prob - prevprob = 6.41606e-006
sumdiff - prevsumdiff = -1.22835e-008
Probability of good choice for population of 1572=48.994
prob - prevprob = 6.40381e-006
sumdiff - prevsumdiff = -1.22444e-008
Probability of good choice for population of 1574=48.9946
prob - prevprob = 6.39161e-006
sumdiff - prevsumdiff = -1.22055e-008
Probability of good choice for population of 1576=48.9952
prob - prevprob = 6.37944e-006
sumdiff - prevsumdiff = -1.21668e-008
Probability of good choice for population of 1578=48.9959
```

```
prob - prevprob = 6.36731e-006
sumdiff - prevsumdiff = -1.21282e-008
Probability of good choice for population of 1580=48.9965
prob - prevprob = 6.35522e-006
sumdiff - prevsumdiff = -1.20898e-008
Probability of good choice for population of 1582=48.9971
prob - prevprob = 6.34317e-006
sumdiff - prevsumdiff = -1.20516e-008
Probability of good choice for population of 1584=48.9978
prob - prevprob = 6.33116e-006
sumdiff - prevsumdiff = -1.20136e-008
Probability of good choice for population of 1586=48.9984
prob - prevprob = 6.31918e-006
sumdiff - prevsumdiff = -1.19757e-008
Probability of good choice for population of 1588=48.999
prob - prevprob = 6.30724e-006
sumdiff - prevsumdiff = -1.1938e-008
Probability of good choice for population of 1590=48.9997
prob - prevprob = 6.29534e-006
sumdiff - prevsumdiff = -1.19005e-008
Probability of good choice for population of 1592=49.0003
prob - prevprob = 6.28348e-006
sumdiff - prevsumdiff = -1.18631e-008
Probability of good choice for population of 1594=49.0009
prob - prevprob = 6.27165e-006
sumdiff - prevsumdiff = -1.18259e-008
Probability of good choice for population of 1596=49.0016
prob - prevprob = 6.25986e-006
sumdiff - prevsumdiff = -1.17888e-008
Probability of good choice for population of 1598=49.0022
prob - prevprob = 6.24811e-006
sumdiff - prevsumdiff = -1.17519e-008
Probability of good choice for population of 1600=49.0028
prob - prevprob = 6.2364e-006
sumdiff - prevsumdiff = -1.17152e-008
Probability of good choice for population of 1602=49.0034
prob - prevprob = 6.22472e-006
sumdiff - prevsumdiff = -1.16786e-008
Probability of good choice for population of 1604=49.004
prob - prevprob = 6.21308e-006
sumdiff - prevsumdiff = -1.16422e-008
Probability of good choice for population of 1606=49.0047
prob - prevprob = 6.20147e-006
sumdiff - prevsumdiff = -1.1606e-008
Probability of good choice for population of 1608=49.0053
prob - prevprob = 6.1899e-006
sumdiff - prevsumdiff = -1.15699e-008
Probability of good choice for population of 1610=49.0059
```

```
prob - prevprob = 6.17837e-006
sumdiff - prevsumdiff = -1.1534e-008
Probability of good choice for population of 1612=49.0065
prob - prevprob = 6.16687e-006
sumdiff - prevsumdiff = -1.14982e-008
Probability of good choice for population of 1614=49.0071
prob - prevprob = 6.15541e-006
sumdiff - prevsumdiff = -1.14626e-008
Probability of good choice for population of 1616=49.0077
prob - prevprob = 6.14398e-006
sumdiff - prevsumdiff = -1.14271e-008
Probability of good choice for population of 1618=49.0084
prob - prevprob = 6.13259e-006
sumdiff - prevsumdiff = -1.13918e-008
Probability of good choice for population of 1620=49.009
prob - prevprob = 6.12123e-006
sumdiff - prevsumdiff = -1.13566e-008
Probability of good choice for population of 1622=49.0096
prob - prevprob = 6.10991e-006
sumdiff - prevsumdiff = -1.13216e-008
Probability of good choice for population of 1624=49.0102
prob - prevprob = 6.09862e-006
sumdiff - prevsumdiff = -1.12868e-008
Probability of good choice for population of 1626=49.0108
prob - prevprob = 6.08737e-006
sumdiff - prevsumdiff = -1.12521e-008
Probability of good choice for population of 1628=49.0114
prob - prevprob = 6.07615e-006
sumdiff - prevsumdiff = -1.12175e-008
Probability of good choice for population of 1630=49.012
prob - prevprob = 6.06497e-006
sumdiff - prevsumdiff = -1.11831e-008
Probability of good choice for population of 1632=49.0126
prob - prevprob = 6.05382e-006
sumdiff - prevsumdiff = -1.11488e-008
Probability of good choice for population of 1634=49.0132
prob - prevprob = 6.04271e-006
sumdiff - prevsumdiff = -1.11147e-008
Probability of good choice for population of 1636=49.0138
prob - prevprob = 6.03162e-006
sumdiff - prevsumdiff = -1.10808e-008
Probability of good choice for population of 1638=49.0144
prob - prevprob = 6.02058e-006
sumdiff - prevsumdiff = -1.10469e-008
Probability of good choice for population of 1640=49.015
prob - prevprob = 6.00956e-006
sumdiff - prevsumdiff = -1.10133e-008
Probability of good choice for population of 1642=49.0156
```

```
prob - prevprob = 5.99858e-006
sumdiff - prevsumdiff = -1.09797e-008
Probability of good choice for population of 1644=49.0162
prob - prevprob = 5.98764e-006
sumdiff - prevsumdiff = -1.09463e-008
Probability of good choice for population of 1646=49.0168
prob - prevprob = 5.97673e-006
sumdiff - prevsumdiff = -1.09131e-008
Probability of good choice for population of 1648=49.0174
prob - prevprob = 5.96585e-006
sumdiff - prevsumdiff = -1.088e-008
Probability of good choice for population of 1650=49.018
prob - prevprob = 5.955e-006
sumdiff - prevsumdiff = -1.0847e-008
Probability of good choice for population of 1652=49.0186
prob - prevprob = 5.94418e-006
sumdiff - prevsumdiff = -1.08142e-008
Probability of good choice for population of 1654=49.0192
prob - prevprob = 5.9334e-006
sumdiff - prevsumdiff = -1.07815e-008
Probability of good choice for population of 1656=49.0198
prob - prevprob = 5.92265e-006
sumdiff - prevsumdiff = -1.07489e-008
Probability of good choice for population of 1658=49.0204
prob - prevprob = 5.91194e-006
sumdiff - prevsumdiff = -1.07165e-008
Probability of good choice for population of 1660=49.021
prob - prevprob = 5.90125e-006
sumdiff - prevsumdiff = -1.06842e-008
Probability of good choice for population of 1662=49.0216
prob - prevprob = 5.8906e-006
sumdiff - prevsumdiff = -1.06521e-008
Probability of good choice for population of 1664=49.0222
prob - prevprob = 5.87998e-006
sumdiff - prevsumdiff = -1.06201e-008
Probability of good choice for population of 1666=49.0227
prob - prevprob = 5.86939e-006
sumdiff - prevsumdiff = -1.05882e-008
Probability of good choice for population of 1668=49.0233
prob - prevprob = 5.85884e-006
sumdiff - prevsumdiff = -1.05565e-008
Probability of good choice for population of 1670=49.0239
prob - prevprob = 5.84831e-006
sumdiff - prevsumdiff = -1.05249e-008
Probability of good choice for population of 1672=49.0245
prob - prevprob = 5.83782e-006
sumdiff - prevsumdiff = -1.04934e-008
Probability of good choice for population of 1674=49.0251
```

```
prob - prevprob = 5.82736e-006
sumdiff - prevsumdiff = -1.0462e-008
Probability of good choice for population of 1676=49.0257
prob - prevprob = 5.81693e-006
sumdiff - prevsumdiff = -1.04308e-008
Probability of good choice for population of 1678=49.0262
prob - prevprob = 5.80653e-006
sumdiff - prevsumdiff = -1.03997e-008
Probability of good choice for population of 1680=49.0268
prob - prevprob = 5.79616e-006
sumdiff - prevsumdiff = -1.03688e-008
Probability of good choice for population of 1682=49.0274
prob - prevprob = 5.78582e-006
sumdiff - prevsumdiff = -1.0338e-008
Probability of good choice for population of 1684=49.028
prob - prevprob = 5.77551e-006
sumdiff - prevsumdiff = -1.03073e-008
Probability of good choice for population of 1686=49.0286
prob - prevprob = 5.76523e-006
sumdiff - prevsumdiff = -1.02767e-008
Probability of good choice for population of 1688=49.0291
prob - prevprob = 5.75499e-006
sumdiff - prevsumdiff = -1.02463e-008
Probability of good choice for population of 1690=49.0297
prob - prevprob = 5.74477e-006
sumdiff - prevsumdiff = -1.0216e-008
Probability of good choice for population of 1692=49.0303
prob - prevprob = 5.73459e-006
sumdiff - prevsumdiff = -1.01858e-008
Probability of good choice for population of 1694=49.0309
prob - prevprob = 5.72443e-006
sumdiff - prevsumdiff = -1.01557e-008
Probability of good choice for population of 1696=49.0314
prob - prevprob = 5.71431e-006
sumdiff - prevsumdiff = -1.01258e-008
Probability of good choice for population of 1698=49.032
prob - prevprob = 5.70421e-006
sumdiff - prevsumdiff = -1.00959e-008
Probability of good choice for population of 1700=49.0326
prob - prevprob = 5.69414e-006
sumdiff - prevsumdiff = -1.00663e-008
Probability of good choice for population of 1702=49.0331
prob - prevprob = 5.68411e-006
sumdiff - prevsumdiff = -1.00367e-008
Probability of good choice for population of 1704=49.0337
prob - prevprob = 5.6741e-006
sumdiff - prevsumdiff = -1.00072e-008
Probability of good choice for population of 1706=49.0343
```

```
prob - prevprob = 5.66412e-006
sumdiff - prevsumdiff = -9.9779e-009
Probability of good choice for population of 1708=49.0348
prob - prevprob = 5.65417e-006
sumdiff - prevsumdiff = -9.94869e-009
Probability of good choice for population of 1710=49.0354
prob - prevprob = 5.64425e-006
sumdiff - prevsumdiff = -9.9196e-009
Probability of good choice for population of 1712=49.036
prob - prevprob = 5.63436e-006
sumdiff - prevsumdiff = -9.89063e-009
Probability of good choice for population of 1714=49.0365
prob - prevprob = 5.6245e-006
sumdiff - prevsumdiff = -9.86178e-009
Probability of good choice for population of 1716=49.0371
prob - prevprob = 5.61467e-006
sumdiff - prevsumdiff = -9.83304e-009
Probability of good choice for population of 1718=49.0376
prob - prevprob = 5.60486e-006
sumdiff - prevsumdiff = -9.80442e-009
Probability of good choice for population of 1720=49.0382
prob - prevprob = 5.59509e-006
sumdiff - prevsumdiff = -9.77592e-009
Probability of good choice for population of 1722=49.0388
prob - prevprob = 5.58534e-006
sumdiff - prevsumdiff = -9.74754e-009
Probability of good choice for population of 1724=49.0393
prob - prevprob = 5.57562e-006
sumdiff - prevsumdiff = -9.71927e-009
Probability of good choice for population of 1726=49.0399
prob - prevprob = 5.56593e-006
sumdiff - prevsumdiff = -9.69111e-009
Probability of good choice for population of 1728=49.0404
prob - prevprob = 5.55627e-006
sumdiff - prevsumdiff = -9.66307e-009
Probability of good choice for population of 1730=49.041
prob - prevprob = 5.54663e-006
sumdiff - prevsumdiff = -9.63514e-009
Probability of good choice for population of 1732=49.0415
prob - prevprob = 5.53702e-006
sumdiff - prevsumdiff = -9.60733e-009
Probability of good choice for population of 1734=49.0421
prob - prevprob = 5.52744e-006
sumdiff - prevsumdiff = -9.57963e-009
Probability of good choice for population of 1736=49.0426
prob - prevprob = 5.51789e-006
sumdiff - prevsumdiff = -9.55203e-009
Probability of good choice for population of 1738=49.0432
```

```
prob - prevprob = 5.50837e-006
sumdiff - prevsumdiff = -9.52455e-009
Probability of good choice for population of 1740=49.0437
prob - prevprob = 5.49887e-006
sumdiff - prevsumdiff = -9.49718e-009
Probability of good choice for population of 1742=49.0443
prob - prevprob = 5.4894e-006
sumdiff - prevsumdiff = -9.46993e-009
Probability of good choice for population of 1744=49.0448
prob - prevprob = 5.47996e-006
sumdiff - prevsumdiff = -9.44278e-009
Probability of good choice for population of 1746=49.0454
prob - prevprob = 5.47054e-006
sumdiff - prevsumdiff = -9.41573e-009
Probability of good choice for population of 1748=49.0459
prob - prevprob = 5.46115e-006
sumdiff - prevsumdiff = -9.3888e-009
Probability of good choice for population of 1750=49.0465
prob - prevprob = 5.45179e-006
sumdiff - prevsumdiff = -9.36198e-009
Probability of good choice for population of 1752=49.047
prob - prevprob = 5.44246e-006
sumdiff - prevsumdiff = -9.33526e-009
Probability of good choice for population of 1754=49.0476
prob - prevprob = 5.43315e-006
sumdiff - prevsumdiff = -9.30865e-009
Probability of good choice for population of 1756=inf
prob - prevprob = inf
sumdiff - prevsumdiff = inf
Probability of good choice for population of 1758=inf
prob - prevprob = nan
sumdiff - prevsumdiff = nan
Probability of good choice for population of 1760=inf
prob - prevprob = nan
sumdiff - prevsumdiff = nan
Probability of good choice for population of 1762=inf
prob - prevprob = nan
sumdiff - prevsumdiff = nan
Probability of good choice for population of 1764=inf
prob - prevprob = nan
sumdiff - prevsumdiff = nan
Probability of good choice for population of 1766=inf
prob - prevprob = nan
sumdiff - prevsumdiff = nan
Probability of good choice for population of 1768=inf
prob - prevprob = nan
sumdiff - prevsumdiff = nan
Probability of good choice for population of 1770=inf
```

```
prob - prevprob = nan
sumdiff - prevsumdiff = nan
Probability of good choice for population of 1772=inf
prob - prevprob = nan
sumdiff - prevsumdiff = nan
Probability of good choice for population of 1774=inf
prob - prevprob = nan
sumdiff - prevsumdiff = nan
Probability of good choice for population of 1776=inf
prob - prevprob = nan
sumdiff - prevsumdiff = nan
Probability of good choice for population of 1778=inf
prob - prevprob = nan
sumdiff - prevsumdiff = nan
Probability of good choice for population of 1780=inf
prob - prevprob = nan
sumdiff - prevsumdiff = nan
Probability of good choice for population of 1782=inf
prob - prevprob = nan
sumdiff - prevsumdiff = nan
Probability of good choice for population of 1784=inf
prob - prevprob = nan
sumdiff - prevsumdiff = nan
Probability of good choice for population of 1786=inf
prob - prevprob = nan
sumdiff - prevsumdiff = nan
Probability of good choice for population of 1788=inf
prob - prevprob = nan
sumdiff - prevsumdiff = nan
Probability of good choice for population of 1790=inf
prob - prevprob = nan
sumdiff - prevsumdiff = nan
Probability of good choice for population of 1792=inf
prob - prevprob = nan
sumdiff - prevsumdiff = nan
Probability of good choice for population of 1794=inf
prob - prevprob = nan
sumdiff - prevsumdiff = nan
Probability of good choice for population of 1796=inf
prob - prevprob = nan
sumdiff - prevsumdiff = nan
Probability of good choice for population of 1798=inf
prob - prevprob = nan
sumdiff - prevsumdiff = nan
Probability of good choice for population of 1800=inf
prob - prevprob = nan
sumdiff - prevsumdiff = nan
Probability of good choice for population of 1802=inf
```

```
prob - prevprob = nan
sumdiff - prevsumdiff = nan
Probability of good choice for population of 1804=inf
prob - prevprob = nan
sumdiff - prevsumdiff = nan
Probability of good choice for population of 1806=inf
prob - prevprob = nan
sumdiff - prevsumdiff = nan
Probability of good choice for population of 1808=inf
prob - prevprob = nan
sumdiff - prevsumdiff = nan
Probability of good choice for population of 1810=inf
prob - prevprob = nan
sumdiff - prevsumdiff = nan
Probability of good choice for population of 1812=inf
prob - prevprob = nan
sumdiff - prevsumdiff = nan
Probability of good choice for population of 1814=inf
prob - prevprob = nan
sumdiff - prevsumdiff = nan
Probability of good choice for population of 1816=inf
prob - prevprob = nan
sumdiff - prevsumdiff = nan
Probability of good choice for population of 1818=inf
prob - prevprob = nan
sumdiff - prevsumdiff = nan
Probability of good choice for population of 1820=inf
prob - prevprob = nan
sumdiff - prevsumdiff = nan
Probability of good choice for population of 1822=inf
prob - prevprob = nan
sumdiff - prevsumdiff = nan
Probability of good choice for population of 1824=inf
prob - prevprob = nan
sumdiff - prevsumdiff = nan
Probability of good choice for population of 1826=inf
prob - prevprob = nan
sumdiff - prevsumdiff = nan
Probability of good choice for population of 1828=inf
prob - prevprob = nan
sumdiff - prevsumdiff = nan
Probability of good choice for population of 1830=inf
prob - prevprob = nan
sumdiff - prevsumdiff = nan
Probability of good choice for population of 1832=inf
prob - prevprob = nan
sumdiff - prevsumdiff = nan
Probability of good choice for population of 1834=inf
```

```
prob - prevprob = nan
sumdiff - prevsumdiff = nan
Probability of good choice for population of 1836=inf
prob - prevprob = nan
sumdiff - prevsumdiff = nan
Probability of good choice for population of 1838=inf
prob - prevprob = nan
sumdiff - prevsumdiff = nan
Probability of good choice for population of 1840=inf
prob - prevprob = nan
sumdiff - prevsumdiff = nan
Probability of good choice for population of 1842=inf
prob - prevprob = nan
sumdiff - prevsumdiff = nan
Probability of good choice for population of 1844=inf
prob - prevprob = nan
sumdiff - prevsumdiff = nan
Probability of good choice for population of 1846=inf
prob - prevprob = nan
sumdiff - prevsumdiff = nan
Probability of good choice for population of 1848=inf
prob - prevprob = nan
sumdiff - prevsumdiff = nan
Probability of good choice for population of 1850=inf
prob - prevprob = nan
sumdiff - prevsumdiff = nan
Probability of good choice for population of 1852=inf
prob - prevprob = nan
sumdiff - prevsumdiff = nan
Probability of good choice for population of 1854=inf
prob - prevprob = nan
sumdiff - prevsumdiff = nan
Probability of good choice for population of 1856=inf
prob - prevprob = nan
sumdiff - prevsumdiff = nan
Probability of good choice for population of 1858=inf
prob - prevprob = nan
sumdiff - prevsumdiff = nan
Probability of good choice for population of 1860=inf
prob - prevprob = nan
sumdiff - prevsumdiff = nan
Probability of good choice for population of 1862=inf
prob - prevprob = nan
sumdiff - prevsumdiff = nan
Probability of good choice for population of 1864=inf
prob - prevprob = nan
sumdiff - prevsumdiff = nan
Probability of good choice for population of 1866=inf
```

```
prob - prevprob = nan
sumdiff - prevsumdiff = nan
Probability of good choice for population of 1868=inf
prob - prevprob = nan
sumdiff - prevsumdiff = nan
Probability of good choice for population of 1870=inf
prob - prevprob = nan
sumdiff - prevsumdiff = nan
Probability of good choice for population of 1872=inf
prob - prevprob = nan
sumdiff - prevsumdiff = nan
Probability of good choice for population of 1874=inf
prob - prevprob = nan
sumdiff - prevsumdiff = nan
Probability of good choice for population of 1876=inf
prob - prevprob = nan
sumdiff - prevsumdiff = nan
Probability of good choice for population of 1878=inf
prob - prevprob = nan
sumdiff - prevsumdiff = nan
Probability of good choice for population of 1880=inf
prob - prevprob = nan
sumdiff - prevsumdiff = nan
Probability of good choice for population of 1882=inf
prob - prevprob = nan
sumdiff - prevsumdiff = nan
Probability of good choice for population of 1884=inf
prob - prevprob = nan
sumdiff - prevsumdiff = nan
Probability of good choice for population of 1886=inf
prob - prevprob = nan
sumdiff - prevsumdiff = nan
Probability of good choice for population of 1888=inf
prob - prevprob = nan
sumdiff - prevsumdiff = nan
Probability of good choice for population of 1890=inf
prob - prevprob = nan
sumdiff - prevsumdiff = nan
Probability of good choice for population of 1892=inf
prob - prevprob = nan
sumdiff - prevsumdiff = nan
Probability of good choice for population of 1894=inf
prob - prevprob = nan
sumdiff - prevsumdiff = nan
Probability of good choice for population of 1896=inf
prob - prevprob = nan
sumdiff - prevsumdiff = nan
Probability of good choice for population of 1898=inf
```

```
prob - prevprob = nan
sumdiff - prevsumdiff = nan
Probability of good choice for population of 1900=inf
prob - prevprob = nan
sumdiff - prevsumdiff = nan
Probability of good choice for population of 1902=inf
prob - prevprob = nan
sumdiff - prevsumdiff = nan
Probability of good choice for population of 1904=inf
prob - prevprob = nan
sumdiff - prevsumdiff = nan
Probability of good choice for population of 1906=inf
prob - prevprob = nan
sumdiff - prevsumdiff = nan
Probability of good choice for population of 1908=inf
prob - prevprob = nan
sumdiff - prevsumdiff = nan
Probability of good choice for population of 1910=inf
prob - prevprob = nan
sumdiff - prevsumdiff = nan
Probability of good choice for population of 1912=inf
prob - prevprob = nan
sumdiff - prevsumdiff = nan
Probability of good choice for population of 1914=inf
prob - prevprob = nan
sumdiff - prevsumdiff = nan
Probability of good choice for population of 1916=inf
prob - prevprob = nan
sumdiff - prevsumdiff = nan
Probability of good choice for population of 1918=inf
prob - prevprob = nan
sumdiff - prevsumdiff = nan
Probability of good choice for population of 1920=inf
prob - prevprob = nan
sumdiff - prevsumdiff = nan
Probability of good choice for population of 1922=inf
prob - prevprob = nan
sumdiff - prevsumdiff = nan
Probability of good choice for population of 1924=inf
prob - prevprob = nan
sumdiff - prevsumdiff = nan
Probability of good choice for population of 1926=inf
prob - prevprob = nan
sumdiff - prevsumdiff = nan
Probability of good choice for population of 1928=inf
prob - prevprob = nan
sumdiff - prevsumdiff = nan
Probability of good choice for population of 1930=inf
```

```
prob - prevprob = nan
sumdiff - prevsumdiff = nan
Probability of good choice for population of 1932=inf
prob - prevprob = nan
sumdiff - prevsumdiff = nan
Probability of good choice for population of 1934=nan
prob - prevprob = nan
sumdiff - prevsumdiff = nan
Probability of good choice for population of 1936=nan
prob - prevprob = nan
(Output truncated)
```

6 Acknowledgement

I dedicate this article to God.

7 Bibliography

References

- [1] Few Algorithms for ascertaining merit of a document - <http://arxiv.org/pdf/1006.4458.pdf>
- [2] TAC 2010 Update summarization by Interview Algorithm (http://www.nist.gov/tac/publications/2010/appendices/Summarization/guided/CMI_IT.pdf)
- [3] Interview algorithm is in IP=PSPACE
- [4] Complexity theory, Sanjeev arora and Boaz Barak
- [5] Majority function in non-uniform NC1, Mix Barrington
- [6] Sorting networks for majority function, Ajtai ,Komlos, Szemeredi
- [7] Valiant's non-constructive majority function circuit

Arrow's Theorem, Circuit For Democracy and Psuedorandom Choice and P Versus NP - Draft

SrinivasanKannan(alias)Ka.Shrinivaasan(alias)ShrinivasKannan
Ph : 9789346927, 9003082186, 9791165980

KrishnaiResearchOpenSourceProducts : http://sourceforge.net/users/ka_shrinivaasan,
ZODIACDATASOFT : https://github.com/shrinivaasanka/ZodiacDatasoft
https://www.ohloh.net/accounts/ka_shrinivaasan
ResearchWebsite : https://sites.google.com/site/kuja27/
(ka.shrinivaasan@gmail.com, shrinivas.kannan@gmail.com, kashrinivaasan@live.com)

September 17, 2014

Abstract

In this draft article a circuit for majority voting that simulates the real-world electors is constructed and its relation to Arrow's theorem and implications for P Vs NP is described. This is a typeset version that unifies already existing handwritten notes and drafts.

1 Pseudorandom Choice and Circuit for Democracy

1. Conventional Majority circuit that outputs 1 if atleast half the inputs are one is chosen
2. Inputs to this majority circuit are SAT circuits for each voter in a democracy.
3. Wishlist of each of the voter is reduced to a CNF formula and a SAT circuit is constructed for it.
4. The variables and clauses in the voter SAT are expectations of each of the voter that is assigned 1 or 0 by the voter while deciding to vote for or against.
5. Thus above Majority circuit augmented with SAT oracles or circuits is in NP. This reduction from 3-SAT is rather trivial.
6. Above circuit makes democracy an NP-complete problem.
7. Using the arguments for 100% convergence case of P(good) series (mentioned in bibliography links below) there exists a Pseudorandom generator based Choice in P or NC that achieves the same objective thereby creating an intriguing anomaly.
8. In the P(good) series if both LHS(Pseudorandom choice) and RHS(Majority voting) are 1 (zero error), then there is a polytime algorithm (pseudorandom choice) to the NP problem of Majority voting which is a counterexample. One way functions exist when the series does not converge and do not exist when series converges to 1. Thus $P \neq NP$ if and only if there is no perfection in voting (and in general any entity or process in universe).

- Moreover, finding even a single perfect voter who never makes an error itself looks to be intractable. Infact this circuit requires all voter SATs to be perfect for the series to converge to 1 thereby placing a tighter restriction than mere intractability.

2 Boolean function sensitivity, Condorcet Elections, Arrow's Theorem and P(Good) series

- https://sites.google.com/site/kuja27/ImplicationRandomGraphConvexHullsAndPerfectVoterProblem01 – 11.pdf* on decidability of existence of perfect voter and the probability series for a good choice of *https://sites.google.com/site/kuja27/CircuitForComputingErrorProbabilityOfMajorityVoter* are related to already well studied problems in social choice theory but problem definition is completely different.
- Arrow's theorem of social choice for an irrational outcome in condorcet election of more than 2 candidates and its complexity theory fourier analysis proof [GilKalai] are described in *www.cs.cmu.edu/odonnell/papers/analysis-survey.pdf*.
- Irrational outcome is a paradox where the society is "confused" or "ranks circularly" in choice of a candidate in multipartisan condorcet voting. Rational outcome converges to 91.2% with a possibility of 8.8% irrational outcome.
- What is perplexing is the fact that this seems to contravene guarantee of unique restricted partitions described based on money changing problem and lattices in *https://sites.google.com/site/kuja27/S04-17.pdf* and *https://sites.google.com/site/kuja27/IntegerPartitionAndHashFunctions2014.pdf* which are also for elections in multipartisan setting (if condorcet election is done).
- Probably a "rational outcome" is different from "good choice" where rationality implies without any paradoxes in ranking alone without giving too much weightage to the "goodness" of a choice by the elector. Actual real-life elections are not condorcet elections where NAE tuples are generated.
- It is not a conflict between Arrow's theorem as finding atleast 1 denumerant in multipartisan voting partition is NP-complete (as it looks) - which can be proved by ILP as in point 20 of *https://sites.google.com/site/kuja27/IntegerPartitionAndHashFunctions2014.pdf* - the assumption is that candidates are not ranked; they are voted independently in a secret ballot by electors and they can be more than 3. The elector just chooses a candidate and votes without giving any ordered ranking for the candidates which makes it non-condorcet.
- Moreover Arrow's theorem for 3 candidate condorcet election implies a non-zero probability of error in voting which by itself prohibits a perfect voting system.
- If generalized to any election and any number of candidates it could be an evidence in favour of $P \neq NP$ by proving that perfect voter does not exist and using democracy Maj+SAT circuits and P(Good) probability series convergence (As described in handwritten notes and drafts:
 - http://sourceforge.net/projects/acadpdrafts/files/ImplicationGraphsPGoodEquationAndPNotE.pdf*
 - https://sites.google.com/site/kuja27/ImplicationRandomGraphConvexHullsAndPerfectVoterProblem01 – 11.pdf*,

- (c) <https://sites.google.com/site/kuja27/LowerBoundsForMajorityVotingPseudorandomChoice.pdf>
 - (d) <https://sites.google.com/site/kuja27/CircuitForComputingErrorProbabilityOfMajorityVoting20.pdf>
 - (e) <https://sites.google.com/site/kuja27/PhilosophicalAnalysisOfDemocracyCircuitAndPRGChoice03-26.pdf>.
9. But it is not known that if Arrow's theorem can be generalized for infinite number of candidates as above and whether such an electoral system is decidable.
10. The possibility of a circular ranking in 3-condorcet election implies that there are some scenarios where voter can err though not exactly an error in making a "good choice" (or Perfect Voter Problem is decidable in case of 3 candidates condorcet election).
11. Error by a Voter SAT circuit implies that voter votes 0 instead of 1 and 1 instead of 0. This is nothing but the sensitivity of the voter boolean function i.e number of erroneous variable assignments by the voter that change the per-voter decision input to the Majority circuit.
12. Thus more the sensitivity or number of bits to be flipped to change the voter decision, less the probability of error by the voter. If sensitivity is denoted by s , $1/q^s$ is probability that a single bit is flipped and probability of error by the voter is p then $p = k/q^s$ for some constant k which is derived by the conditional probability that $\Pr[m \text{ bits are flipped}] = \Pr[m\text{-th bit is flipped}/(m-1) \text{ bits already flipped}]*\Pr[(m-1) \text{ bits are flipped}] = 1/q * 1/q^{m-1} = 1/q^m$ (and $m=s$).
13. This expression for p can be substituted in the Probability series defined in <https://sites.google.com/site/kuja27/CircuitForComputingErrorProbabilityOfMajorityVoting20.pdf>. Probability of single bit is $1/q$ if the number of variables across all clauses is q and each variable is flipped independent of the other.

3 Majority Circuit with SAT input voters - Illustration with a P(good) series example

Document Viewer

PhilosophicalAnalysisOfDemocracyCircuitAndPRGChoice_2014-03-26.pdf NONE

16 January Thursday 16-01-2014

24/03/2014 P = probability of good choice
 $(1-p) = \text{complements above}$

17 January Friday 17

Population size $2n$

Probability of at least $n+1$ people or voters making good decision.

$$P(\text{good} \geq n+1) = P(n+1) + P(n+2) + \dots + P(2n)$$

$$= \sum_{n+1}^{2n} \binom{2n}{h} p^h (1-p)^{2n-h} + \sum_{n+2}^{2n} \binom{2n}{h+2} p^{h+2} (1-p)^{2n-h-2} + \dots$$

$$+ \dots + \sum_{2n}^{2n} \binom{2n}{h} p^h$$

when $p=1$ above is $\sum_{2n}^{2n} \binom{2n}{h} = \frac{2n!}{(n+1)!} = 1$

NP Complete Democracy Circuit

Corresponding MAJORITY with SAT circuit

each voter has a SAT circuit which needs to be assigned by a candidate. Candidate is chosen if at least $(n+1)$ SATs are satisfied

If all voters are zero-error above is an error-free democracy circuit which is NP-complete.

Previous is for RHS of P(good) when all 17-01-2014 voters are perfect.

LHS of P(good) is pseudorandom choice which is in P and probability of good choice = no of good voters / no of total voters since chosen one is one of the voters. If all voters are error-free pseudorandom choice is in P.

Example: 3 voters $n+1 = 2$

8 possible voting patterns for 3 voters

$$= P(2) + P(3)$$

$$= \frac{3}{8} + \frac{1}{8} = \frac{4}{8} = \frac{1}{2}$$

$$(3C_2 + 3C_3) \times \frac{1}{2^3}$$

$$= \frac{3!}{2!1!} + \frac{3!}{1!2!} = 3+1 = \frac{4}{2^3} = \frac{1}{2}$$

If good and bad decisions are (anergically) distributed some bit patterns in above should never occur.

February 2014

Elision Campaign is the process of "satisfying" the Voter circuits.

4 Acknowledgement

I dedicate this article to God.

5 Bibliography

References

- [1] The series convergence is illustrated in <http://sourceforge.net/p/asfer/code/HEAD/tree/cpp-src/miscellaneous/MajorityVotingErrorProbabilityConvergence.JPG>
- [2] Algorithms for Intrinsic Merit of a Document - Recursive Gloss Overlap Algorithm for wordnet subgraph - <http://arxiv.org/abs/1006.4458>

- [3] Slides illustrating WordNet subgraph or Definition Graph Construction using Recursive Gloss Overlap Algorithm - <https://sites.google.com/site/kuja27/PresentationTAC2010.pdf?attredirects=0>
- [4] Document Summarization from WordNet subgraph obtained by Recursive Gloss Overlap - <https://sites.google.com/site/kuja27/DocumentSummarizationusingSpectralGraphTheoryGOGraph2010.pdf?attredirects=0>
- [5] Primitive implementation of the above at: <http://sourceforge.net/p/asfer/code/HEAD/tree/python-src/InterviewAlgorithm/>
- [6] TAC 2010 - Update Summarization using Interview Algorithm - <http://www.nist.gov/tac/publications/2010/participant.papers/CMIIIT.proceedings.pdf>
- [7] WordNet - <http://wordnet.princeton.edu/>
- [8] Equating Majority Voting and Pseudorandom Choice - <https://sites.google.com/site/kuja27/LowerBoundsForMajorityVotingPseudorandomChoice.pdf?attredirects=0>
- [9] Probability of good majority choice Series Derivation - <https://sites.google.com/site/kuja27/CircuitForComputingErrorProbabilityOfMajorityVoting2014.pdf>
- [10] Axiom of Choice and Majority Voting <https://sites.google.com/site/kuja27/IndepthAnalysisOfVariantOfAxiomOfChoiceAndMajorityVoting>
- [11] A PRG for Pseudorandom Choice using Chaos Theory Attractors <https://sites.google.com/site/kuja27/ChaoticPRG.pdf?attredirects=0>
- [12] Interview Algorithm and Interactive Proof <https://sites.google.com/site/kuja27/InterviewAlgorithmInPSPACE>
- [13] Equivalence of Integer Partitions and Hash Functions (This also gives an alternative proof of NP-completeness of multipartisan democracy using reduction from restricted partition problem Money Changing Problem and Schur theorem for partitions)- <https://sites.google.com/site/kuja27/IntegerPartitionAndHashFunctions2014.pdf>
- [14] Handwritten notes: Philosophical analysis of Democracy circuit and Pseudorandom choice <https://sites.google.com/site/kuja27/PhilosophicalAnalysisOfDemocracyCircuitAndPRGChoice20140417.pdf?attredirects=0>
- [15] Handwritten notes: on Schur Theorem, Restricted Partitions and Hash Functions, NP-Completeness of MultiPartisan Majority Voting <https://sites.google.com/site/kuja27/SchurTheoremMCPAndDistinctPartitions20140417.pdf?attredirects=0>
- [16] Handwritten notes: An experimental theory of Convex Hull of the logical implication graph and Perfect Voter problem - <https://sites.google.com/site/kuja27/ImplicationRandomGraphConvexHullsAndPerfectVoterProblem20140117.pdf?attredirects=0>
- [17] Handwritten notes: Experimental Notes on Logical Implication Graphs: An experimental logical implication graph model for P and NP - <http://sourceforge.net/projects/acadpdrafts/files/ImplicationGraphsPGoodEquationAndPNotEqualToNP.pdf>

- [18] A gadget for randomized space filling to simulate some natural phenomena and LP for it -
<https://sites.google.com/site/kuja27/Analysis%20of%20a%20Randomized%20Space%20Filling%20Algorithm>
- [19] Social Choice Theory, Arrow's Theorem and Boolean functions - <http://www.ma.huji.ac.il/kalai/CHAOS.pdf>

1 Problem statement

Is there a way to find a complement of a function dened on integers(Both domain and range are integers)?

1.1 Example:

Suppose you have a set of positive integers = 1,2,3,4,5.... Let us say we create a subset of even integers out of it = 2,4,6,8,... This set of even numbers can be generated by using the function $f(x) = 2x$, $x=1,2,3...$ The complement of this set is set of odd numbers and they can be generated using $g(x) = 2x-1$, $x=1,2,3,4,...$ So I dene $g(x)=2x-1$ to be the complement function of $f(x)=2x$ over the set of integers.

Similarly for the function $f(x,y) = xy$, the complement function (ignoring non-trivial factors) is over the set of all primes. Is there a generic way to arrive at the complement function given an arbitrary integer function

2 Computational difficulty of the complement

If we reduce halting problem to the question of finding complement then it is undecidable (since it is equivalent to asking if turing machine computing the complement terminates)

3 Circuit version of above - Construction of a circuit for the complement (for integer valued functions)

Circuit for complement is constructed as below:

1. find the values of f for all $n=1,2,3,4,5...$ and make each of them a clause
2. construct DNF of above clauses
3. complement the DNF to get CNF (k-CNF where $k = \log(n)$)

This gives an infinite k-CNF formula which has a circuit of unbounded fanin but exponential size. If the circuit construction algorithm for above terminates is again undecidable.

Thus we have a complement class of TC0 i.e kind of co-TC0 which synonymizes the complement of $f(x,y) = xy$ and this co-TC0 is the complement language of integer multiplication.

Decidability of Existence and Construction of a Complement of a given function

Ka.Shrinivaasan, Chennai Mathematical Institute (CMI)
(shrinivas@cmi.ac.in)

April 28, 2011

Abstract

This article defines a complement of a function and conditions for existence of such a complement function and presents few algorithms to construct a complement.

1 Goal

Goal is to analyze decidability of finding a complement of a function defined over both domain and range as integers. There are two cases to consider:

1. The domain and range of the function are infinite
2. The domain and range of the function are finite

2 Example of a complement function (when the domain and range are infinite)

Consider an infinite set of positive integers = $\{1, 2, 3, 4, 5, \dots\}$. Let us say we create a subset of even integers out of it = $\{2, 4, 6, 8, \dots\}$. This set of even numbers can be generated by using the function $f(x) = 2x$, $x=0,1,2,3, \dots$. A complement of this set is infinite set of odd numbers and they can be generated using $g(x) = 2x+1$, $x=0,1,2,3,4, \dots$. So we define $g(x) = 2x + 1$ to be a *complement function* of $f(x) = 2x$ over the set of integers.

Similarly for the function $f(x, y) = xy$, a complement function (ignoring trivial factors 1 and n) is over the set of all primes. Thus finding a complement g of $f(x,y)$ amounts to finding distribution of primes i.e g shows a pattern in primes. This was a question posted in google group sci.math (http://groups.google.com/group/sci.math/browse_thread/thread/46ab0335ce9205d9/f3c67ee19926e02e)

3 Existence of a complement when domain and range are finite

3.1 Definitions

1. Let U be a universe of n -bit integers of size 2^n integers
2. Function f_n is defined over the domain of a -bit integers and range U of n bit integers - $f: \{0, 1\}^a \rightarrow \{0, 1\}^n$ for $a > 0$ and $n > 0$
3. Function g_n is defined over the domain of b -bit integers and range U of n bit integers - $g: \{0, 1\}^b \rightarrow \{0, 1\}^n$ for $b > 0$ and $n > 0$
4. $L(f_n) = \text{language generated by the function } f_n = \{f(x_0), f(x_1), \dots, f(x_{2^a-1})\}$
5. $L(g_n) = \text{language generated by a complement function } g_n = \{g(x_0), g(x_1), \dots, g(x_{2^b-1})\}$
6. Function g_n is defined as a *complement* of f_n if
 - (a) g_n is total
 - (b) $g_n(x) \neq f_n(x) \quad \forall x$
 - (c) $L(f_n) \cup L(g_n) = \{0, 1\}^n$ and
 - (d) $L(f_n) \cap L(g_n) = \emptyset$
7. The subscript n for the functions f and g denotes the finiteness of the size of input and output.

3.2 Conditions for existence of a complement

For a complement to exist the universe U - the range of functions f_n and g_n - must be partitioned by the functions f_n and g_n . Due to this, if

1. $f_n: \{0, 1\}^a \rightarrow \{0, 1\}^n$ and
2. $g_n: \{0, 1\}^b \rightarrow \{0, 1\}^n$

then the inequality $2^a + 2^b \geq 2^n$ must hold. This is because, function f_n maps 2^a elements to X number of elements ($X \leq 2^a$) in U and function g_n maps 2^b elements to Y elements ($Y \leq 2^b$) in U . This makes $X + Y \leq 2^a + 2^b$. Since $X + Y = 2^n$ by definition of complement, $2^a + 2^b \geq 2^n$.

3.3 Algorithm for complement construction

Input to a complement construction algorithm is the function f_n and the output is a complement function g_n of f_n . (Univariate function (polynomial) has been assumed. This can be generalized to multivariate polynomials.). Algorithm comprises of 3 high-level steps.

3.3.1 STEP 1-Getting the range set for complement function

From the function f_n we can get the set $T = \{f_n(i) / 0 \leq i \leq 2^a - 1\}$. From this the set $S = U \setminus T$ can be constructed.

3.3.2 STEP 2-Construction of mapping

Before we construct a function representation, a mapping has to be established between a set $P = \{0, 1, 2, 3, \dots, 2^b - 1\}$ and the set S . This mapping is a lookup table which imposes an ordering on the set S . We have that $|S| = 2^n - |T|$. For a valid function to exist, we must have $2^b \geq 2^n - |T|$ (since for a valid function, domain must be greater than or equal in size to range) and hence we have to choose b such that $b \geq \log_2(2^n - |T|)$. Number of possible mappings (and hence functions) between P and S is $(|S|)^{|P|}$. But this includes mappings which are not onto also. To cover set S , we have to choose one onto mapping to extract a complement function out of various possible complement functions. One such mapping is done as follows

1. $i=0$
2. $\text{while}(i <= 2^b)$
 - (a) If set S is not empty, choose uniformly at random, an element e from set S and map it to i and add to the lookup table as $\langle Key : Value \rangle$ pair $\langle i : e \rangle$. Remove element e from the set S . This implies $g_n(i) = e$ where g_n is a complement to be constructed.
 - (b) else if the set S is empty, map the first element in the ordering $g_n(0)$ to i by adding $\langle i : g_n(0) \rangle$ to lookup table i.e $g_n(i) = g_n(0)$.
 - (c) $i := i+1$

The above mapping procedure selects one mapping out of $|S|!$ onto mappings (since after adding each element to lookup table, number of possible candidates reduces by 1 element from previous iteration). Many other ways of mapping are possible different from the procedure above. Each one of these mapping procedures result in different complement function.

3.3.3 STEP 3-Construction of complement function representation from mapping

Once we have the mapping constructed as in previous step, we can either apply 1) polynomial interpolation (or) 2) Arithmetization using simple replacement of boolean expressions with arithmetic expressions (or) by fourier polynomials 3) or by applying lambda calculus to get the function representation for a complement. These 3 algorithms for construction of complement function from mapping are explained in detail next.

Algorithm 1 - By polynomial interpolation

1. Since we have been given with a mapping from previous step between $x_i \in \{0, 1\}^b$, and $y_i \in S$, we can apply polynomial interpolation theorem which states that given $x_1, y_1, x_2, y_2, \dots, x_n, y_n$ with $x_i \neq x_k$ for $i \neq k$, \exists polynomial $p^{(n)} \in F[x]$ of degree less than or equal to $n-1$, such that $p^{(n)}(x_i) = y_i$. The recurrence for building the polynomial is as follows:

$$p^{(i+1)}(x) = p^{(i)}(x) + \{[y_{j+1} - p^{(i)}(x_{j+1})]/q^{(i+1)}(x_{j+1})\} * [q^{(i+1)}(x)]$$

$$\text{where } q^{(i+1)}(x) = \prod_{j=1}^i (x - x_j)$$

Algorithm 2 - By representing boolean function by polynomial:

1. From the mapping obtained in previous section, we can construct function representation for a complement. Let binary of $x = s_b s_{b-1} \dots s_3 s_2 s_1$ and binary of $Mapping(g_n)(x) = z_n z_{n-1} \dots z_3 z_2 z_1$ obtained by looking up x from mapping lookup table constructed earlier. The i^{th} bit of $Mapping(g_n)(x)$ is denoted as $g_n^i(x)$.
2. Construct a DNF boolean function for z_i with 2^b clauses where each clause corresponds to a binary string in $\{0, 1\}^b$ with binary digits replaced by variables as: $z_i = (\neg s_b \cap \neg s_{b-1} \cap \dots \cap \neg s_3 \cap \neg s_2 \cap \neg s_1 \cap g_n^i(x_0)) \cup (\neg s_b \cap \neg s_{b-1} \cap \dots \cap \neg s_3 \cap s_2 \cap \neg s_1 \cap g_n^i(x_1)) \cup \dots (s_b \cap s_{b-1} \cap \dots \cap s_3 \cap s_2 \cap s_1 \cap g_n^i(x_{2^b-1}))$. For example clause for a binary string 001010 with $r=6$ will be $(\neg s_6 \cap \neg s_5 \cap s_4 \cap \neg s_3 \cap s_2 \cap \neg s_1)$. The bits of $Mapping(g_n)(x)$ will be hard-coded in the formula above from mapping already constructed.
3. The boolean function above outputs the i^{th} bit of $g_n(x)$ where $binary(x) = s_b s_{b-1} \dots s_3 s_2 s_1$. Intuitively, this boolean function is a multiplexer which selects the i -th bit for $g_n(x)$ among 2^b candidate bits. We have to construct n boolean functions as above for each of the n bits of $g_n(x)$
4. These boolean functions can be minimized to obtain minimum equivalent expression. After minimization we can arithmetize the boolean functions in one of the two ways below to get arithmetic expressions (polynomials):
 - (a) Simple Arithmetization
 - i. In the above boolean expression, replace a positive literal s_i by a variable s_i and a negative literal $(\neg s_i)$ by an arithmetic expression $(1 - s_i)$.
 - ii. In the above boolean expression, replace a subexpression of the form $y_m \cap y_n$ by the arithmetic expression $t_m * t_n$
 - iii. In the above boolean expression, replace a subexpression of the form $y_m \cup y_n$ by the arithmetic expression $1 - ((1 - t_m) * (1 - t_n))$
 - iv. Repeat above steps recursively till the full boolean function for z_i gets arithmetized.
 - v. At the end of above algorithm the whole boolean function for z_i is arithmetized as a polynomial in variables $s_b, s_{b-1}, \dots, s_3, s_2, s_1$ which are the bit positions of input x. Let us denote this by $Arith(z_i)$. Since we have arithmetic expressions for individual bit positions of $g_n(x)$ in terms of bit positions of x, we can construct $g_n(x)$ by weighted summation given as $g_n(x) = 2^n * Arith(z_n) + 2^{n-1} * Arith(z_{n-1}) + \dots + 1 * Arith(z_1)$. This is the required complement function $g_n(s_b, s_{b-1}, \dots, s_3, s_2, s_1)$
 - (b) Arithmetization by fourier expansion
 - i. Since we have a boolean function for each bit position z_i in step 2, the RHS of each z_i can be expanded as a fourier polynomial in variables $s_b, s_{b-1}, \dots, s_3, s_2, s_1$ which are the bit positions of input x. Thus we have a polynomial for i -th bit of $g_n(x)$ in terms of bit positions of x. Let us denote this expansion as $fourier(z_i)$
 - ii. Thus $g_n(x)$ can be represented as the weighted summation of the bit positions of $g_n(x)$ as: $g_n(x) = 2^n * ((fourier(z_n) + 1)/2) +$

$2^{n-1}*((fourier(z_{n-1})+1)/2)+...+1*((fourier(z_1)+1)/2)$ since fourier polynomials evaluate to $\{-1, 1\}$ and this transformation is needed to get the bits $\{0, 1\}$

- iii. Complement function $g_n(s_b, s_{b-1}, \dots, s_3, s_2, s_1)$ thus has been constructed by applying fourier representation ($g_n(x)$ is a polynomial in the bit positions of x ($s_b, s_{b-1}, \dots, s_3, s_2, s_1$))

Algorithm 3 - applying Lambda Calculus:

1. Obtaining $g_n(x)$ given x , suffices for showing the existence complement. First we develop an iterative procedure for returning $g_n(x)$ if x is given as follows:
2. Procedure GetGofX(x):
 - (a) $u:=0$
 - (b) $z:=-1$
 - (c) choose b such that $b \geq \log_2(2^n - |T|)$ where $T = \{f_n(i) / 0 \leq i \leq 2^a - 1\}$.
 - (d) if ($x > 2^b - 1$) printerror "error: out of range"
 - (e) while($u \leq 2^n - 1$ and $z \leq 2^b - 1$)
 - i. if u is in T then $u:=u+1$
 - ii. if u is not in T then
 - A. $z:=z+1$
 - B. if($u \leq 2^n - 1$)
 - C. {
 - D. if(z equals x) add the entry $< x : u >$ to mapping table to signify $g_n(x) = u$ and return u as $g_n(x)$
 - E. else add the entry $< z : u >$ to mapping table to signify $g_n(z) = u$ and set $u:=u+1$
 - F. }
 - G. else break
 - (f) if($u \geq 2^n$) then add the entry $< x : g_n(0) >$ to mapping table to signify $g_n(x) = g_n(0)$ and return $g_n(0)$ as $g_n(x)$ by looking up mapping table for key 0
3. Correctness of the above procedure can be proved since we are guaranteed that $2^b \geq (2^n - |T|)$ by choice of b . The while loop breaks when all the 2^n elements fall either into $L(f_n)$ or to $L(g_n)$ and similar to previous section on construction of a mapping, we hardcode $g_n(x) = g_n(0)$
4. The inequality $2^a + 2^b \geq 2^n$ must hold as mentioned earlier in section on conditions for existence of complement (where a and b are input sizes to f and g respectively).
5. Now that we have an iterative procedure for $g_n(x)$, we can convert this into lambda expression by replacing loops with a fixed point combinator from lambda calculus. The equals() relation above is bitwise equivalence represented by literals for each bit. This lambda expression can again be converted to a standard logical formula(for example a boolean formula) in terms of bit positions.

4 Existence of a complement when domain and range are infinite

4.1 Definitions

1. Function f is defined over the domain of integers and range of integers -
 $f: \mathbb{Z} - \mathbb{Z}$
2. Function g is defined over the domain of integers and range of integers -
 $g: \mathbb{Z} - \mathbb{Z}$
3. $L(f) = \text{language generated by the function } f = \{f(x_0), f(x_1), \dots\}$ - an infinite set
4. $L(g) = \text{language generated by a complement function } g = \{g(x_0), g(x_1), \dots\}$
- an infinite set
5. Function g is defined as a *complement* of f if
 - (a) g is total
 - (b) $g(x) \neq f(x) \quad \forall x$
 - (c) $L(f) \cup L(g) = \{1, 2, 3, 4, 5, \dots\}$ and
 - (d) $L(f) \cap L(g) = \emptyset$
6. The subscript n for the functions f and g has been removed compared to previous section.

4.2 Decidability of complementation

To find an algorithm which would give a complement function g , given input function f . There are two cases to consider:

1. $L(f)$ is not recursive but recursively enumerable.
2. $L(f)$ is recursive

4.3 Case 1 - $L(f)$ is not recursive but recursively enumerable

Q - $L(f)$ recognizer

1. for all x
 - (a) if $y == f(x)$ then return "y is in $L(f)$ "
2. return "y is not in $L(f)$ "

$L(f)$ is recursively enumerable since TM Q , outputs yes if a string y is in $L(f)$ ($y = f(x)$ for some x) but loops when y is not in $L(f)$. But $L(f)$ is not recursive since TM Q does not halt on all inputs. From definitions above we have complement of $L(f) = L(g)$. [Question of if Q outputs yes/no for any y is equivalent to asking if Q halts on any input. Reduction is defined as :

$x \in L(Q) \iff R(x) \in \text{HaltingProblem}.$] $L(f)$ is recursive if and only if both $L(f)$ and its complement $L(g)$ are recursively enumerable as proved below.

Claim: $L(f)$ is recursive if and only if both $L(f)$ and its complement $L(g)$ are recursively enumerable.

1. (\Rightarrow) If $L(f)$ is recursive then $L(f)$ is recursively enumerable. If $L(f)$ is recursive then complement of $L(f) = L(g)$ is recursive and hence $L(g)$ is recursively enumerable.
2. (\Leftarrow) Run two TMs $M(f)$ and $M(g)$ for $L(f)$ and $L(g)$. Input y . If y is in $L(f)$ then $M(f)$ returns yes and $M(g)$ loops. If y is not in $L(f)$ then $M(f)$ loops and $M(g)$ returns yes. Run $M(h)$ accepting y which simulates $M(f)$ and $M(g)$. If $M(f)$ return yes then $M(h)$ returns yes and if $M(g)$ returns yes then $M(h)$ returns no. Thus $M(h)$ returns yes or no for input y without looping.

Since $L(f)$ is not recursive, $L(g)$ is not recursively enumerable. So in this case, $L(g)$ has no turing machine that recognizes it.

4.4 Case 2 - $L(f)$ is recursive

$L(f)$ being recursive implies that there exists a turing machine which given input y always halts with accept($y \in L(f)$) or reject (*else*) states.

If $L(f)$ is recursive, $L(g)$ is also recursive by closure under complementation. This implies $L(g)$ has a turing machine deciding membership.

4.5 Decidability of construction of a complement function for infinite recursive set $L(g)$ - for Case 2 above

In previous section, $L(f_n)$ and $L(g_n)$ were finite languages. This restriction was helpful for defining finite DNF formulas for complement g_n (or polynomial interpolation which is also for finite set of points). But for infinite recursive set $L(g)$, algorithms described in previous sections like polynomial interpolation or arithmetization do not work. This is because, we (either) might have to construct infinite boolean formulas and evaluate such an infinite formula (or) might have to interpolate over infinite set of points. This procedure might never terminate. This can be proved undecidable by reduction from halting problem. Thus infinite cardinality for $L(g)$ makes construction of complement for infinite recursive sets undecidable.

4.6 Complementation as a nontrivial property - Application of Rice's theorem

1. Let Turing machine P_f which accepts an integer x as input and writes $f(x)$ in output and goes to accept state. P_f rejects if x is not an integer. $L(P_f) = L(f_n)$. (A function can be thought of as a lambda expression which is equivalent to turing machine)
2. Let Turing machine Q_g which accepts an integer x as input and writes $g(x)$ in output and goes to accept state. Q_g rejects if x is not an integer.

$$L(Q_g) = L(g).$$

3. Let Turing machine X_i which accepts a pair of encodings of Turing machines (P_f, Q_g) . $L(X_i) = \{(P_f^k, Q_g^k)\}$, $k = 1, 2, 3, \dots$. X_i accepts only if input is an ordered pair of turing machine encoding of 2 functions over integers, and rejects otherwise.
4. Let Turing machine Y which accepts encodings of Turing machines X_i and simulates X_i .
 $L(Y) = \{X_i\}$, $i = 1, 2, 3, \dots$

We want to test a property P such that language decided by X_i - the set of ordered pairs of TM encodings - are complements of each other. This property P is nontrivial since not all ordered pairs are in P and if $L(X_m) = L(X_n)$ then both X_m and X_n are in P. Since by Rice's theorem it is undecidable if a language decided by a Turing machine has a nontrivial property, L(Y) is undecidable.

5 References

1. Lecture Notes on Algebra and Computation, MadhuSudhan, MIT (for Polynomial Interpolation)
2. Various literature on Lambda Calculus on internet
3. A question posted to Google Group sci.math <http://groups.google.com/group/>-
 - (a) [/sci.math/browse_thread/thread/46ab0335ce9205d9/f3c67ee19926e02e](http://sci.math/browse_thread/thread/46ab0335ce9205d9/f3c67ee19926e02e)
4. Complexity-I and Complexity-II Lecture Notes(2009 through 2010) - IMSc
5. Various literature on Rice's Theorem on internet
6. Automata Theory and Formal Languages - Aho, Hopcroft and Ullman
7. Introduction to Theory of Computation - Michael Sipser

K.Srinivasan - ஸ்ரீநிவாஸன் கண்ணன் - ஶரீனிவாஸன் கண்ண

(also spelt as: Srinivasan Kannan, Ka.Shrinivaasan,
Shrinivas Kannan)

Permanent Address:

Srinivasan Kannan,
S/O. (Late) P.R.E.S.Kannan,
172, Gandhi Adigal Salai,



Kumbakonam-612001, TamilNadu, India.

e-mail : ka.shrinivaasan@gmail.com

shrinivas.kannan@gmail.com

kashrinivaasan@live.com

Mobile :9789346927

Name spellings in

consultancy/employer/academic

records : K.Srinivasan (academics, BaaN,Sun

Microsystems), Kannan Srinivasan (Sun

Microsystems),

Srinivasan Kannan (Verizon, Clockwork

Interviews, CloudEnablers),

Shrinivas Kannan(webMethods/SoftwareAG and

CMI), Ka.Shrinivaasan (Global Analytics)

Personal website(research) :-

<https://acadpdrafts.readthedocs.io>

NeuronRain Documentation:- <https://neuronrain->

[documentation.readthedocs.io/en/latest/](https://neuronrain-documentation.readthedocs.io/en/latest/)

Krishna iResearch Open Source Repositories:

[https://sourceforge.net/users/ka_shrinivaasan,](https://sourceforge.net/users/ka_shrinivaasan)

[https://github.com/shrinivaasanka,](https://github.com/shrinivaasanka)

<https://gitlab.com/shrinivaasanka>

OpenHub profile:

https://www.openhub.net/accounts/ka_shrinivaasan

Krishna iResearch GitHub Organization:

<https://github.com/Krishna-iResearch>

BRIHASPATHI - Private Virtual Classrooms:

GitHub -

Private repositories of virtual classrooms for various commercial online courses (for graduate students and professionals - requires GitHub student logins) - BigData

and Machine Learning, Topics in Mathematics and Computer Science, Linux Kernel and Cloud, Vedic Astrology, English, Hindi - <https://github.com/Brihaspathi> - Consultancy offered on BigData-Machine Learning, Linux Kernel-Cloud and other IT arena, Vedic Astrology (Brihaspathi Jyotish Vigyan Kendra - <https://github.com/Brihaspathi/BrihaspathiJyotishVigyanKendra> - ப்ரிஹஸ்பதி ஜ்யோதிஷ் விக்யாந் கேந்த்ரா - Personal reading, Matrimonial matchmaking,...)

JAIMINI Closed Source Derivative of NeuronRain:

GitHub - <https://github.com/Brihaspathi/jaimini>

SourceForge - <https://sourceforge.net/projects/jaimini/>

GitLab - <https://gitlab.com/shrinivaasanka/jaimini>

LatexPDF CV:

[https://github.com/shrinivaasanka/Krishna_iResearch_DxygenDocs/blob/
master/kuja27_website_mirrored/site/kuja27/CV.pdf](https://github.com/shrinivaasanka/Krishna_iResearch_DxygenDocs/blob/master/kuja27_website_mirrored/site/kuja27/CV.pdf)

Date of Birth as per records: 19 May 1977

Passport: M9583737

PAN: AOVPS2844F

Aadhaar: 772717942542

Marital status: Unmarried (Never Married)

COMPLETE ACADEMIC AND WORK HISTORY

ACADEMICS

- 1. Elementary Schooling, RC Morning Star, Kumbakonam (1982-1987)**
- 2. Higher Secondary, Town Higher Secondary**

School,Kumbakonam (1987-1994) (SSLC -470/500 and Plus Two 1115/1200)

3. B.A - Hindi - Praveen Uttarardh - Dakshin Bharat

Hindi Prachar Sabha,Chennai (privately tutored from Kumbakonam) - (1988-1992)

4. Bachelor of Engineering (Computer Science & Engg) (1995-99),

PSG College of Technology, Coimbatore-641004 INDIA (CGPA/Percentage: 8.8/87.75)

5. M.Sc (Computer Science), Chennai Mathematical Institute, Chennai (2008 - 10) CGPA 8.08 (coursework - excluding MSc thesis), 8.06(including MSc thesis)

6. Ph.D (Research Scholar in Computer Science), Chennai Mathematical Institute, Chennai (coursework, TAC 2010 - expansion of MSc thesis above - and Complement

Function miniproject - August 2010 - August 2011) - resigned due to personal reasons and went back to Industry. Presently pursuing independent research.

COURSES DONE IN M.Sc and Ph.D and Self-study

M.Sc courses [CMI and IIT-Chennai] and self-study (2008-2010):

Haskell, Operating systems, Distributed systems, Theory of computation, Databases, Logic-1, Complexity-1, Principles of Programming Languages (Java and lambda calculus), Algorithms, Graph theory, Cryptography, Datamining-1, Information Retrieval, Automata, Concurrency, Timed Systems

Ph.D courses[CMI and IMSc] and self study (August 2010

- October 2011):

Complexity-2, Topics in data mining(Recommender Systems, Streaming Algorithms), Randomized algorithms(including PTAS), Logspace computation, Program Verification, Program Analysis, Program Slicing, Computational number theory and algebra, Computational geometry, Expander graphs, Combinatorics(Generating functions), Probabilistic method, Communications Complexity, Linear Programming And Combinatorial Optimization and Computational Biology (BioInformatics) algorithms for sequence alignment.

PUBLICATIONS

(Google Scholar URL - [http://scholar.google.co.in/citations?
user=eLZY7CIAAAJ&hl=en](http://scholar.google.co.in/citations?user=eLZY7CIAAAJ&hl=en))

**1.(During PhD) Decidability of Existence and Construction
of a Complement of a function with Prof.Meena Mahajan,
IMSc, 2011**

[\(http://arxiv.org/abs/1106.4102\)](http://arxiv.org/abs/1106.4102)

- Some unreviewed draft additions to the above for
Complement Function Circuit and its connection to
Riemann Zeta Function, Ramanujan graphs and Ihara
Zeta Functions are at:

<http://sourceforge.net/p/asfer/code/HEAD/tree/AstroInferDesign.txt> with some illustrations:

https://github.com/shrinivaasanka/Krishna_iResearch_DoxygenDocs/blob/master/kuja27_website_mirrored/site/kuja27/RamanujanGraphsRiemannZetaFunctionAndIharaZetaFunction

on.pdf?attredirects=0&d=1 and

<https://github.com/shrinivaasanka/>

Krishna_iResearch_DoxygenDocs/blob/master/

kuja27_website_mirrored/site/kuja27/

RZFAndIZF_25October2014.pdf?attredirects=0&d=1

2.(Master's Thesis) Few Algorithms for Ascertaining Merit of a document (Citation graph Maxflow, Recursive Gloss Overlap Algorithm and Interview algorithm applying either of the previous two with applications of them) with Profs. Ravindran(IIT Chennai) and Madhavan Mukund (CMI) in 2009-2010 - <http://arxiv.org/abs/1006.4458>.

3. (During PhD) Evaluated NIST TAC 2010 dataset

(Summarization track) for

Update Summarization by applying Interview Algorithm -

appeared in proceedings of TAC 2010 at:

<http://www.nist.gov/tac/publications/2010/>

[participant.papers/CMI_IIT.proceedings.pdf](http://www.nist.gov/tac/publications/2010/participant.papers/CMI_IIT.proceedings.pdf)

RESEARCH STATEMENTS:

4. Research Statement 1 -

[https://github.com/shrinivaasanka/Krishna_iResearch_DoxygenD](https://github.com/shrinivaasanka/Krishna_iResearch_DoxygenDocs/blob/master/kuja27_website_mirrored/site/kuja27/)

[ocs/blob/master/kuja27_website_mirrored/site/kuja27/](https://github.com/shrinivaasanka/Krishna_iResearch_DoxygenDocs/blob/master/kuja27_website_mirrored/site/kuja27/)

[ResearchStatement2.pdf?attredirects=0](https://github.com/shrinivaasanka/Krishna_iResearch_DoxygenDocs/blob/master/kuja27_website_mirrored/site/kuja27/)

5. Research Statement 2 - with some proof sketches

**(includes a new timeout manager algorithm implemented
in Global Analytics Decision Platform 3.0) -**

[https://github.com/shrinivaasanka/Krishna_iResearch_DoxygenD](https://github.com/shrinivaasanka/Krishna_iResearch_DoxygenDocs/blob/master/kuja27_website_mirrored/site/kuja27/)

ocs/blob/master/kuja27_website_mirrored/site/kuja27/

PhDThesisProposal.pdf?attredirects=0

6. Research Statement 3 - with some proof sketches -

https://github.com/shrinivaasanka/Krishna_iResearch_Doxyg

enDocs/blob/master/kuja27_website_mirrored/site/kuja27/

Research_Writeup.pdf?attredirects=0&d=1

NeuronRain Theory Drafts (2003-present) - private

unaffiliated research - theory aligned to features of

NeuronRain opensource products

Complexity Theoretic Analysis of Non-majority and Majority Social Choice, Majority Voting Boolean Composition Circuit and KRW Conjecture, P versus NP, ABC Conjecture, Four color Theorem, Zorn Lemma, Axiom of Choice (AOC), XOR Lemma and Hardness Amplification, Circuit Lowerbounds, Pseudorandomness(generators and extractors), Goodness of Voting and Margulis-Russo Threshold/Condorcet Jury Theorem (and its recent versions by [Black], [Ladha]), Theoretical Electronic Voting Machines and Pre-poll -Post-poll Analytics, Vowelless Syllable Boundary Text Compression and Compressed Sensing, Computational Chaos, Polynomial Reconstruction Problem, Complement Functions - Complementary Sets and their Function

Representation(e.g Beatty Functions), Combinatorics
(Ramsey coloring of sequences), PAC Learning, Prime-
Composite complementation and pattern in primes,
Goldbach conjecture, Arithmetic Progressions, Diophantine
Analysis and Representation, Riemann Zeta Function,
Hypergeometric Functions, Clouds - Logical time and
causality(EventNet), Formal Languages (Turing degrees,
Embedding in vector space, Lambda Calculus, Category
Theory, Logic) and learning lambda expressions from
Natural Language Text, Cognitive Psychology - Grounded
Cognition and ThoughtNet Evocation, Partial order
intrinsic merit rankings and Galois connections, Graph
theoretic/Computational Neurolinguistic/Question-
Answering Interview Intrinsic Merit/Fitness/Fame and
Experiential Learning in the context of WWW (people,
text, audio - speech and music, visuals-video and images,

economies) and Social/Economic networks, Social Network Models - Cellular Automaton and Random Graph Diffusion of Concepts-Memes-Fads-Cybercrimes, Game Theory, BKS Conjecture and Question-Answering, Machine Translation, Algorithmic Graph Theoretic Learning Models, Computational Learning Theory, Software Analytics/Program Analysis/Debug Analytics, Operating System Kernel and Scheduler Analytics, Astronomical Analytics of Celestial Bodies and correlations to Seismic-Atmospheric-Oceanic events, Urban planning analytics, Computational Astrophysics - N-body problem, Media Analytics and Advertisement Analytics, Preferential Attachment, Brand Loyalty and Business Intelligence, People Analytics/HR Analytics, Sports Analytics, Handwriting and Face Recognition for unique identification, Fame/Merit Equilibrium (Welfare Functions,

Flow Market Equilibrium and Convex-Concave Programming in Algorithmic Economics applied to Fame-Merit) and Economic Merit(Intrinsic pricing), Cryptocurrencies and Money Trail (EventNet Graph), Optimal Denomination and Money Changing - Coin Problem, Mechanism Design, Time series analysis (economic and weather forecasts), Neural Networks and Deep Learning, Quantum mechanics and Intrinsic Fitness/Merit(Bose-Einstein condensation in networks), Locality Sensitive Hashing and Separate Chaining Hash tables, Multiple Agent Resource Allocation, Integer Partitions(additive and multiplicative), Set Partitions, Space filling/Lagrangian Four Square Theorem Tiling/Circle Packing, Exact Cover, Random Closed Packing, Number Theory, Quadratic and Linear

Programming, Cellular Automata, Satisfiability (Least Square SAT Solvers and QBFSAT), Random restrictions and Hastad Switching Lemma, Classical NC-PRAM-BSP (k-mergesort, segment tree, wavelet tree, ray shooting queries, planar point location, sorting networks, local search of rasterized hyperbolic segment arithmetic progressions), Randomized NC and Quantum NC Computational Geometric Integer Factoring, Rasterization of Algebraic Curves, Algebraic Geometry, Knot Theory, Topology and Connections amongst them -

(most recent draft updates to all publications previously

and earlier drafts below - in text

format - nonlinear theoretical writeups interspersed

between NeuronRain code commits in SourceForge,

GitLab and GitHub - links to relevant feature

implementations and theory drafts in design notes of

NeuronRain repositories -

AstroInfer, USBmd, VIRGO, KingCobra, GRAFIT, Krishna_iRese

arch_Doxygen_Docs, Acadpdrafts):

Krishna_iResearch_DoxygenDocs (GitHub)

Krishna_iResearch_DoxygenDocs (GitLab)

Krishna_iResearch_DoxygenDocs (SourceForge)

Earlier Drafts:

7.(draft1) IntegerPartitions and Hash Functions

[\(https://github.com/shrinivaasanka/Krishna_iResearch_DoxygenDocs/blob/master/kuja27_website_mirrored/site/kuja27/IntegerPartitionAndHashFunctions.pdf?attredirects=0\)](https://github.com/shrinivaasanka/Krishna_iResearch_DoxygenDocs/blob/master/kuja27_website_mirrored/site/kuja27/IntegerPartitionAndHashFunctions.pdf?attredirects=0)

8.(draft2) IntegerPartitions and Hash Functions

[\(https://github.com/shrinivaasanka/Krishna_iResearch_DoxygenDocs/blob/master/kuja27_website_mirrored/site/kuja27/IntegerPartitionAndHashFunctions_2014.pdf?attredirects=0&d=1\)](https://github.com/shrinivaasanka/Krishna_iResearch_DoxygenDocs/blob/master/kuja27_website_mirrored/site/kuja27/IntegerPartitionAndHashFunctions_2014.pdf?attredirects=0&d=1)

9.(draft1) Interview Algorithm is in $IP=PSPACE$

[\(https://github.com/shrinivaasanka/Krishna_iResearch_DoxygenDocs/blob/master/kuja27_website_mirrored/site/kuja27/InterviewAlgorithmInPSPACE.pdf?attredirects=0\)](https://github.com/shrinivaasanka/Krishna_iResearch_DoxygenDocs/blob/master/kuja27_website_mirrored/site/kuja27/InterviewAlgorithmInPSPACE.pdf?attredirects=0)

10.(draft1) Few Non-trivial questions and Shell Turing

Machines

(https://github.com/shrinivaasanka/Krishna_iResearch_DoxygenDocs/blob/master/kuja27_website_mirrored/site/kuja27/UndecidabilityOfFewNonTrivialQuestions.pdf?attredirects=0)

11. (draft1) Lower Bounds for Majority Voting and Pseudorandom choice -

https://github.com/shrinivaasanka/Krishna_iResearch_DoxygenDocs/blob/master/kuja27_website_mirrored/site/kuja27/LowerBoundsForMajorityVotingPseudorandomChoice.pdf

12. (draft1) Circuits for computation of error probability in majority voting -

https://github.com/shrinivaasanka/Krishna_iResearch_DoxygenDocs/blob/master/kuja27_website_mirrored/site/kuja27/CircuitForComputingErrorProbabilityOfMajorityVoting.pdf

13. (draft2) Circuits for computation of error probability in majority voting -

https://github.com/shrinivaasanka/Krishna_iResearch_DoxygenDocs/blob/master/kuja27_website_mirrored/site/kuja27/

CircuitForComputingErrorProbabilityOfMajorityVoting_2014.pdf

14. (draft1) Indepth analysis of a variant of Majority Voting with relation to ZFC -

https://github.com/shrinivaasanka/Krishna_iResearch_DoxygenDocs/blob/master/kuja27_website_mirrored/site/kuja27/

IndepthAnalysisOfVariantOfMajorityVotingwithZFAOC.pdf?

attredirects=0

15. (draft2) Indepth analysis of a variant of Majority

Voting with relation to ZFC -

https://github.com/shrinivaasanka/Krishna_iResearch_DoxygenDocs/blob/master/kuja27_website_mirrored/site/kuja27/IndepthAnalysisOfVariantOfMajorityVotingwithZFAOC_2014.pdf?attredirects=0

16.(draft1) A Chaos theoretic Parallel Pseudorandom generator in RNC For Majority Voting and Pseudorandom

Choice

[\(https://github.com/shrinivaasanka/Krishna_iResearch_DoxygenDocs/blob/master/kuja27_website_mirrored/site/kuja27/ChaoticPRG.pdf?attredirects=0\)](https://github.com/shrinivaasanka/Krishna_iResearch_DoxygenDocs/blob/master/kuja27_website_mirrored/site/kuja27/ChaoticPRG.pdf?attredirects=0)

17. (draft1) Analysis of a Randomized Space Filling

Algorithm and its Linear Program Formulation

[\(https://github.com/shrinivaasanka/Krishna_iResearch_DoxygenDocs/blob/master/kuja27_website_mirrored/site/kuja27/ChaoticPRG.pdf?attredirects=0\)](https://github.com/shrinivaasanka/Krishna_iResearch_DoxygenDocs/blob/master/kuja27_website_mirrored/site/kuja27/ChaoticPRG.pdf?attredirects=0)

[genDocs/blob/master/kuja27_website_mirrored/site/kuja27/Analysis%20of%20a%20Randomized%20Space%20Filling%20Algorithm%20and%20its%20Linear%20Program%20Formulation.pdf?attredirects=0](http://sourceforge.net/p/asfer/code/HEAD/tree/AstroInferDesign.txt)

18. (draft2) Analysis of a Randomized Space Filling Algorithm and its Linear Program Formulation (Parallel PRG and Cellular Automaton algorithm) -

<http://sourceforge.net/p/asfer/code/HEAD/tree/AstroInferDesign.txt>

19. (draft1) Discrete Hyperbolic Polylogarithmic Sieve For Integer Factorization (version 1) -

https://github.com/shrinivaasanka/Krishna_iResearch_DoxygenDocs/blob/master/kuja27_website_mirrored/site/kuja27/DiscreteHyperbolicPolylogarithmicSieveForIntegerFactorization.pdf?attredirects=0

attredirects=0&d=1

20. (draft2) Discrete Hyperbolic Polylogarithmic Sieve For Integer Factorization - with Interpolation Search (version 2) -

https://github.com/shrinivaasanka/Krishna_iResearch_DoxygenDocs/blob/master/kuja27_website_mirrored/site/kuja27/DiscreteHyperbolicPolylogarithmicSieveForIntegerFactorization_updated_interpolation_search.pdf?attredirects=0&d=1

21. (draft3) Discrete Hyperbolic Polylogarithmic Sieve For Integer Factorization - with Interpolation Search (version 3) -

https://github.com/shrinivaasanka/Krishna_iResearch_DoxygenDocs/blob/master/kuja27_website_mirrored/site/kuja27/DiscreteHyperbolicPolylogarithmicSieveForIntegerFactoriza

[tion_updated_interpolation_search_30June2013.pdf?](#)

[attredirects=0&d=1](#)

22. (draft4) Discrete Hyperbolic Polylogarithmic Sieve For Integer Factorization - with Interpolation Search (version 4 and version 5 with handwritten illustrations and calculations) -

[http://sourceforge.net/projects/acadpdrafts/files/DiscreteHyperbolicPolylogarithmicSieveForIntegerFactorization_updated_interpolation_search.pdf/download](#)

23. (draft5) Discrete Hyperbolic Polylogarithmic Sieve For Integer Factorization - using Rectangular Binary (or) Interpolation Search

[http://sourceforge.net/projects/acadpdrafts/files/DiscreteHyperbolicPolylogarithmicSieveForIntegerFactoriza](#)

[tion_updated_rectangular_interpolation_search.pdf/](#)

[download](#)

24. (draft6) Informal Notes on Derivation of Upperbound

for Discrete Hyperbolic Factorization with Stirling

Formula - using Rectangular Binary or Interpolation

Search (

<http://sourceforge.net/projects/acadpdrafts/files/>

[DiscreteHyperbolicFactorization_UpperboundDerivedWithSt](#)

[irlingFormula_2013-09-10.pdf/download\)](#)

25. (draft7) Discrete Hyperbolic Polylogarithmic Sieve For

Integer Factorization - using Rectangular Binary (or)

Interpolation Search and Upperbound derived with

Stirling Formula

[http://sourceforge.net/projects/acadpdrafts/files/DiscreteHyperbolicPolylogarith](http://sourceforge.net/projects/acadpdrafts/files/)

micSieveForIntegerFactorization_updated_rectangular_interpolation_search_and
_StirlingFormula_Upperbound.pdf/download

(a C++ implementation of this algorithm is at:

<http://sourceforge.net/p/asfer/code/HEAD/tree/cpp-src/miscellaneous/DiscreteHyperbolicFactorizationUpperbound.cpp>

and factorization result logs at:

<http://sourceforge.net/p/asfer/code/HEAD/tree/cpp-src/miscellaneous/>

**26. (draft8) An NC algorithm and some Sequential Search
Algorithms for Discrete Hyperbolic Polylogarithmic Sieve
For Factorization using Binary or Interpolation Search
with Stirling Formula and Logarithmic Sorted Tile Merge
in PRAM model -**

http://sourceforge.net/projects/acadpdrafts/files/DiscreteHyperbolicPolylogarithmicSieveForIntegerFactorization_PRAM_TileMergeAndSearch_And_Stirling_Upperbound.pdf/

download

27. (draft 9) Parallel RAM algorithm for Discrete

Hyperbolic Factorization - AsFer PRAM implementation

design notes with tile id(s)

(<http://sourceforge.net/p/asfer/code/HEAD/tree/asfer-docs/ImplementationDesignNotesForDiscreteHyperbolicFactorizationInPRAM.jpg>)

28. (draft10) Computational Geometric Factorization - An

NC algorithm and some Sequential Search Algorithms for

Discrete Hyperbolic Polylogarithmic Sieve For

Factorization using Binary or Interpolation Search with

Stirling Formula and Logarithmic Sorted Tile Merge in

PRAM model - updated draft with PRAM to NC reduction

and input size details and references -

[http://sourceforge.net/projects/acadpdrafts/files/DiscreteHyperbolicPolylogarithmicSieveForIntegerFactorization_PRAM_TileMergeAndSearch_And_Stirling_Upperbound_updatedraft.pdf/download](http://sourceforge.net/projects/acadpdrafts/files/DiscreteHyperbolicPolylogarithmicSieveForIntegerFactorization_PRAM_TileMergeAndSearch_And_Stirling_Upperbound_updatedraft.pdf)

29. Individual Invention Disclosure for Survival Index

Based Transaction Timeout

Manager (Sun Microsystems) - 2002-2003

https://github.com/shrinivaasanka/Krishna_iResearch_DoxygenDocs/blob/master/kuja27_website_mirrored/site/kuja27/SurvivalIndexBasedTxnTimeoutManager.pdf?attredirects=0

30. AstroInfer (AsFer) Open Source Product Design with

theoretical interludes which substantially expand on the

publications above -

<http://sourceforge.net/p/asfer/code/HEAD/tree/asfer-docs/AsferInferDesign.txt>

31. VIRGO Open Source Product Design with theoretical interludes -

<http://sourceforge.net/p/virgo-linux/code-0/HEAD/tree/trunk/virgo-docs/VirgoDesign.txt>

32. King Cobra - Byzantine distributed request servicing software design with queues and arbiters - theoretical interludes -

<https://sourceforge.net/p/kcobra/code-svn/KingCobraDesignNotes.txt>

33. (draft1) Informal notes - 1 : on Implication Graphs, Error probability of Majority Voting and P Versus NP

Question

<http://sourceforge.net/projects/acadpdrafts/files/>

ImplicationGraphsPGoodEquationAndPNotEqualToNPQuesti
on_excerpts.pdf/download

34. (draft1) Informal notes - 2 : on Minimum Convex
Hulls of Implication Graphs and Hidden Markov Model on
class nodes of Concept Hypergraph

35. (draft1) Informal notes - 3 : on Minimum Convex
Hulls of Implication Random Growth Networks and
Perfect Voter Decidability

36. (draft1) Informal handwritten notes on Philosophical
Analysis of Democracy Circuit and Pseudorandom Choice -

https://github.com/shrinivaasanka/Krishna_iResearch_Doxyg enDocs/blob/master/kuja27_website_mirrored/site/kuja27/PhilosophicalAnalysisOfDemocracyCircuitAndPRGChoice_2014-03-26.pdf

37. (draft1) Informal Handwritten Notes - Distinct Partitions, Hash collision chains and Schur Theorem -

https://github.com/shrinivaasanka/Krishna_iResearch_Doxyg enDocs/blob/master/kuja27_website_mirrored/site/kuja27/SchurTheoremMCPAndDistinctPartitions_2014-04-17.pdf

38. (draft1) Document Summarization from WordNet Subgraph obtained by Recursive Gloss Overlap -

https://github.com/shrinivaasanka/Krishna_iResearch_Doxyg enDocs/blob/master/kuja27_website_mirrored/site/kuja27/DocumentSummarization_using_SpectralGraphTheory_RGO

Graph_2014.pdf?attredirects=0&d=1

39. (draft1) Arrow's Theorem, Circuit For Democracy and Pseudorandom Choice and P Versus NP -

https://github.com/shrinivaasanka/Krishna_iResearch_DoxygenDocs/blob/master/kuja27_website_mirrored/site/kuja27/CircuitsForDemocracyAndPseudorandomChoice_and_PVsNP.pdf?attredirects=0&d=1

40. (draft1) Krishna iResearch Open Source Products

(AsFer, USBmd, VIRGO, KingCobra, Acadpdrafts) - High

Level Handdrawn Architecture Diagram -

<http://sourceforge.net/p/acadpdrafts/code/ci/master/tree/>

Krishna_iResearch_opensourceproducts_archdiagram.pdf

41. (draft1) Miscellaneous notes on Krishna iResearch

Open Source products design, Democracy Circuit,

Complement Function circuit and Parallel RAM to NC

reduction for ANSV algorithm in Discrete Hyperbolic

Factorization

TEAM PATENTS (Sun Microsystems-

Oracle)

1 8,521,875  Identity for data sources

75

2 8,145,759  Dynamically configurable resource pool

,759

3 7,743,083  Common transaction manager interface for local and global transactions

47,739 Read/write lock transaction manager freezing

T

,252

57,640 Transaction manager freezing

T

,545

67,610 Simultaneous global transaction and local

T

,305 transaction management in an application server

77,165 Transaction optimization of read-only data

T

,061 sources

87,134 Utility for configuring and verifying data sources

T

,008

97,082 Specifying transaction manager type at various

T

,432 application levels

Disclosure - Patent Pending (Copyright: Sun

Microsystems/Oracle - 2002 - Reference Number: P8490)

Survival Index Based Transaction Timeout Manager (Java)

PoC implemented on SunOne-iPlanet Application Server

6.5 J2EE-JTS Transaction Manager - now GlassFish

- <https://github.com/javaee/glassfish/tree/master/appserver>

DETAILS OF WORK (1999 – Present)

C/C++/Java/Python on Windows and various Unix flavours

PRODUCTS

Netscape Application Server 4.0, iPlanet/SunONE

Application Server 6.x, SunONE Web/Proxy Server 3.6/4.0,

Apache web server 1.4.x/2.0.x, webMethods Broker

Messaging Server 5.x/6.x/7.x, ASFER Classification and

Inference Software for Large Data Sets

(10.0,12.0,14.9.9,15.1.8,15.6.15), USBmd

(1.0,3.0,14.9.9,15.1.8,15.6.15), VIRGO Linux Kernel

Extensions for Cloud

(5.0,6.0,7.0,8.0,10.0,12.0,14.9.9,15.1.8,15.6.15), Global

Decision Platform(GDP) 2.3.0/ 2.3.1/ 2.5.0/ 2.5.1/ 2.7

/2.7.1/3.0, KingCobra (1.0,14.9.9,15.1.8,15.6.15),

Automated Modelling Platform(AMP), Python 2.4.3 and 2.7

source code, Linux kernel [3.2.0, 3.7.8, 3.9.x, 3.15.5,

4.0.5, 4.1.5, 4.10.3, 4,13,3], Maitreya's Dreams 7.0.3, STL,

BOOST, wxWidgets library, Swiss Ephemeris API (based

on NASA JPL DE406 Ephemeris).

LANGUAGES/PLATFORMS/TOOLS

C/C++, Java 1.5-1.8, J2EE, Python, R and rpy2, Haskell, Solaris 6-10, Windows 2000, HP-UX 11.23 (PA-RISC and IA64), AIX, Linux, Sun Studio, CORBA (Visibroker), VAX FORTRAN, VAX VMS, GNU collection, MS Visual Studio .NET C++, WinDbg, DDD, Visual Basic and ASP, OptimizIt, Eclipse, EclipseCDT, Code::Blocks, ActiveState Python, Cython, IDLE, Haskell FP, Insight, GreatCircle, Valgrind, Callgrind, Python Dumbo MapReduce on Hadoop, Python NLTK, Java MapReduce on Hadoop, Apache HBase, Apache Pig, Apache Hive, Apache Cassandra, Apache Mahout, Apache Spark, Pygraphs, GraphViz, DOT, PiCloud, ActiveMQ, RabbitMQ, Pyutilib workflow, PyF workflow, Linux kernel - [3.2.0, 3.7.8, 3.9.x, 3.15.5, 4.0.5, 4.1.5, 4.10.3, 4.13.3], Maitreya's Dreams 7.0.3, wxWidgets library, Swiss Ephemeris API (based on NASA JPL DE406 Ephemeris), Bioinformatics Sequence

Alignment Tools (BioPython, ClustalOmega), ROOT(CERN),
Geolocation Python API, Beautiful Soup, CRF tools, GATE,
SentiWordNet, Stanford CoreNLP, ENT, MongoDB, WEKA,
Jinja2, Flask, Cloud related - Corestack, Openstack, Mistral,
Jclouds, Libcloud, Docker etc., Tornado, ZeroMQ, Kafka

May 2003 - present - Krishna iResearch (parallel, self-started, not-for-profit, non-funded, open-source long-term research initiative) -

https://sourceforge.net/users/ka_shrinivaasan ,

<https://github.com/shrinivaasanka> ,

<https://gitlab.com/shrinivaasanka>

Working on Research, Design and Development of open

source projects - a machine-learning, cloud and queue augmented Linux kernel - NeuronRain - started by self, copyleft licensed under GPLv3 . Premium technical support is available for above opensource codebases. The GitHub repositories implement NeuronRain Green (for cloud and generic datasets), SourceForge repositories implement NeuronRain Research (for astronomical datasets) and GitLab repositories implement NeuronRain Antariksh (for Drone development). An introductory presentation on NeuronRain based Analytics is at:

https://github.com/shrinivaasanka/Grafit/blob/master/EnterpriseAnalytics_with_NeuronRain.pdf

Latest version 2023#2#2:

1. [NEURONRAIN - Krishna_iResearch_DoxygenDocs - FAQ](#)

and Documentation for AsFer, VIRGO, KingCobra,

Acadpdrafts and USBmd open source product codebases

which are subsystems of Krishna iResearch Intelligent

Cloud OS - NeuronRain

(https://github.com/shrinivaasanka/Krishna_iResearch_DoxygenDocs ,

<http://neuronrain-documentation.readthedocs.io/en/latest/>)

2. **NEURON RAIN** - **AsFer** - **AstroInfer** astronomy and

astrology machine learning inference open source

software which uses algorithms viz.,

2.1 Bayesian Classifier,

2.2 Support Vector Machines,

2.3 Decision Tree Classifier,

2.4 String mining - Pairwise String Sequence

Alignment (Needleman-Wunsch) ,

2.5 String mining - Multiple String Sequence

Alignment(BioPython,ClustalOmega),

2.6 String mining - Powerset construction (C++) and String matching implementation in python(Jellyfish distance measures like Jaro-winkler, Edit-distance, Phonetic Matching rate etc.,) and

2.7 Parsers and Encoding features for USGS and NOAA HURDAT2 data

2.8 Automated Generation of Training and Test set for Classifiers that segregate the datasets into Event Classes and Automated generation of encoded astronomical object files specific to each Event Class.

2.9 Computational Geometric Factorization - Implementation of Bitonic K-MergeSort of Tiles in Pixelated Hyperbolic Arc (Sequential and PRAM) (for drafts in

[https://github.com/shrinivaasanka/Krishna_iResearch_DoxygenDocs/blob
/master/kuja27_website_mirrored/site/kuja27/](https://github.com/shrinivaasanka/Krishna_iResearch_DoxygenDocs/blob/master/kuja27_website_mirrored/site/kuja27/) - A Spark-Python-
C++ implementation of Parallel NC Factorization
has been included which internally executes NC
Bitonic tiles Parallel Merge Sort on Spark Cloud.

2.10 A Concept Hypergraph and HMM(Hidden
Markov Model) based Experimental Inference Model
Design inspired by Cognitive Psychology - a
minimal sentiwordnet evocatives implementation of
this sans HMM named ThoughtNet is part of AsFer

2.11 Design for extracting patterns from
numerical data

2.12 Time Series Analysis - A Chaos Attractor
implementation that generates a sequence of
numbers using logistic equation and uses Python
and R to compute a correlation coefficient between

a non-Chaotic and Chaotic sequence of numbers.

2.13 Discrete Fourier Transform computation for a parsed numerical input sequence using Python and R

2.14 Time Series Analysis - Spline interpolation, LOESS regression, Approximate Linear Polynomial interpolation and graph plot for a parsed numerical input sequence using Python and R (at present implemented for Dow Jones Industrial Average historical dataset and Riemann Zeta Function Zeros dataset)

2.15 An Experimental Text Compression algorithm that uses Hidden Markov Model Maximum Likelihood Estimation based on a String Complexity measure that compresses vowels in text

2.16 An Experimental Minimum Description

Length implementation that uses the previous string complexity measure, computes MDL using Kraft inequality and Shannon entropy.

2.17 Probabilistic Approximate Correct - PAC -

Learning Implementation related to

<http://arxiv.org/abs/1106.4102>

2.18 Wagner-Fischer edit distance implementation

for pairwise similarity of encoded strings

2.19 An example implementation of Interview

Algorithm and Intrinsic Merit Ranking for :

<http://arxiv.org/abs/1006.4458> and

http://www.nist.gov/tac/publications/2010/participant.papers/CMI_IIT.proceedings.pdf

2.20 An experimental Expirable Objects implementation

2.21 Linear and Logistic Regression

- 2.22 K-Means clustering implementation
- 2.23 K-Nearest Neighbours supervised classifier implementation
- 2.24 Linear Perceptron with Gradient Descent Learning
- 2.25 Encoding of Maitreya Dreams to strings and some bugfixes for SWISS Ephemeris degree computation errors
- 2.26 An experimental logical clock for cloud - EventNet - using Python pygraphs and C++ boost::graph with GraphViz rendering (from dot files)
- 2.27 Longest Common Substring implementation for extracting common patterns in already classified and clustered encoded strings dataset
- 2.28 Python script for translating the Pairwise Longest Common Substrings obtained from a cluster of

Kmeans or KNN clustered astronomical dataset into rules that correlate location of astronomical entities to a class of events.

2.29 Utility Knuth-Morris-Pratt String Match algorithm implementation

2.30 Multipurpose bigdata analytics subsystem (storage abstraction on Cassandra/Hbase/Hive/File) for computing metrics from datasets before and after processing by above ML algorithms.

2.31 Streaming Algorithms - Flajolet-Martin LogLog and HyperLogLog Counter implementations for computing cardinality (distinct elements) in streamed multiset data

2.32 Streaming Algorithms - CountMinSketch implementation for computing frequencies (heavy hitters) of elements in streamed multiset data

2.33 Streaming Algorithms - Bloom Filter

implementation for membership query in streamed multiset data stored in Apache Cassandra/Hbase/Hive/File NoSQL tables or disk and stream-simulated through python Generator/Iterable abstraction

2.34 Pig script clients for Hbase/Hive and

Python clients for Hive/Cassandra/Hbase

2.35 Apache Spark Python RDD Transformation

MapReduce script for mining Linux kernel logs and exporting the mined data as config to VIRGO kernel_analytics module. This integrates the AsFer or other Machine Learning algorithms into VIRGO Linux Kernel (with USBmd, KingCobra modules).

2.36 Sequence Mining of ordered encoded strings

(Apriori GSP Algorithm) for inferring Class Association Rules for USGS Earthquake datasets

2.37 Named Entity Recognition using Conditional Random Fields and Hidden Markov Models (skew normal distribution - by finding Viterbi path through Viterbi Dynamic Programming Algorithm)

2.38 Graph Search - Mine for Graph Relations (and render them in graphics) in Text Data by constructing the WordNet subgraph using the recursive gloss overlap algorithm in <http://arxiv.org/abs/1006.4458> and http://www.nist.gov/tac/publications/2010/participant.papers/CMI_IIT.proceeding.s.pdf

2.39 New Unsupervised text classification algorithm implementation that computes nodes with prominent core numbers and PageRanks (nodes with high core-numbers and PageRanks are names of the classes) in the definition graphs obtained by Recursive Gloss Overlap algorithm above.

2.40 Lambda Expression Compositionality from the depth first search closure of Recursive Gloss Overlap graph obtained from a text projected to WordNet.

2.41 Social Network Analysis - Graph creation from social networking sites data (LinkedIn profiles, Twitter Followers etc.,) and their Psycho-Social-Sentiment Analysis (Bonacich Power Centrality, Eigen Vector Centrality, Recursive Gloss Overlap Graph Sentiment Belief Propagation and SentiWordNet based sentiment scoring etc.,)

2.42 Cloud Object Move (or) Cloud Perfect Forwarding implementation - This is a standalone client-server implementation that extends C++ std::move() and rvalue references to moving objects across cloud without copies. Functionality similar to this is required in KingCobra Neuro currency object transactional move

without duplication.

2.43 Deep Learning - Convolution Neural Networks implementation (multifeature kernel maps) for a primitive pattern recognition

2.44 Deep Learning - Backpropagation in Multilayered Neural Networks implementation - applied to a Software Analytics usecase of CPU and Memory usage

2.45 Sequence Mining of ordered encoded strings (Apriori GSP Algorithm) for inferring Class Association Rules for HURDAT2 NOAA Hurricane datasets

2.46 Dependency Injection based BigData Storage Backend Abstraction subsystem for SQL and NoSQL (MySQL, MongoDB etc.,)

2.47 C++-Python Embedding - Python CAPI Embedding which executes python machine learning scripts from C++

2.48 Config File Support for selectively enabling/disabling AsFer algorithms.

2.49 Web Spider implementation for crawl-scrape of URLs (e.g streamed news updates) with Scrapy Framework

2.50 Sentiment Analyzer implementation (with RGO algorithm etc.,) for Spidered Texts (e.g streamed news updates)

2.51 Streaming Frequency Estimation (Heavy Hitters) with CountMeanMinSketch implementation

2.52 An expanded implementation of Interview Algorithm and Intrinsic Merit Ranking with added complexity-machine learning theoretical analysis for :

<http://arxiv.org/abs/1006.4458> and

http://www.nist.gov/tac/publications/2010/participant.papers/CMI_IIT.proceedings.pdf

that ranks a crawled-spidered webpage, writes a DOT file of the graph, visualizes the RGO NetworkX graph in matplotlib, constructs a junction tree and computes standard qualitative graph complexity measures (viz., connectivity, mincut, treewidth of junction tree etc.,) where as intrinsic merit is quantitative (depends on number of edges, vertices, extent of overlap and depth of recursion)

2.53 A rule search script that sifts through a range of astronomical data (with Maitreya-Swiss Ephemeris) and matches sequence mined rules from past training data to predict future weather phenomena (experimental).

2.54 Python Tornado based WebService that exposes REST API via a browser GUI in URL

http://host:33333/neuronrain_auth At present a login and simple template that accepts AsFer scripts and arguments

as inputs and executes, has been implemented (and an experimental redundant AngularJS client interface) With this NeuronRain is a SaaS and PaaS product that can be deployed on Cloud OS and Containers like Docker.

2.55 Apache Spark MapReduce implementation of Interview Algorithm and Intrinsic Merit Ranking (Global MemCache and Local dict() cache enabled) that parallelizes Recursive Gloss Overlap graph construction algorithm in:

<http://arxiv.org/abs/1006.4458> and

http://www.nist.gov/tac/publications/2010/participant.papers/CMI_IIT.proceedings.pdf

and is thus deployable on large scale clouds. Presently benchmarked on single node Spark cluster with local threads SparkContext and various spark-defaults.conf settings.

2.56 Singular Value Decomposition and Principal Component Analysis - R+rpy2 wrapper implementation.

2.57 Approximation with a Probability Distribution - Kullback-Leibler Divergence

2.58 Basic statistics - L1,L2 norms, median etc.,

2.59 A Generic Theory and Implementation of Recursive Lambda Function Growth Learning Algorithm based on Cognitive and Psycholinguistics of which Recursive Gloss Overlap implementation is a special case.

2.60 Python-C-VIRGO Kernel - Cpython extensions - invoking VIRGO system calls (both VIRGO32 and VIRGO64 kernels) from Python userspace for dynamic setting of kernel analytics config key-value pairs

2.61 Python-C++-VIRGO Kernel - Boost::Python extensions - invoking VIRGO system calls(both VIRGO32 and VIRGO64 kernels) from Python userspace for dynamic

setting of kernel analytics config key-value pairs

2.62 Cython Python-C compiler Optimization of previous PySpark Interview Algorithm implementation that has been benchmarked on single node cluster to have substantial performance gains.

2.63 Software Analytics - SATURN Program Analysis Software integrated with VIRGO - SATURN error logs are analyzable by AsFer.

2.64 Perl WordNet::Similarity support and WordNet hypernym-hyponym path subroutine for 2 text words.

2.65 Bitonic Sort implementation - Serial and Parallel on Spark - This can be used as a bigdata sorting tool for numeric data which is in NC when run on Cloud and is $O(n\log n)$ in serial. In the absence of PRAM implementation, NC Bitonic Sort has been invoked as a suitable alternative in parallel tile merge sort step of

Parallel Discrete Hyperbolic Factorization Implementation.

2.66 Long Term Short Term Memory Recurrent Neural Networks implementation

2.67 Reinforcement Learning - Contextual Multi Armed Bandit Evocation based on ThoughtNet - Monte Carlo Policy Search implementation and ThoughtNet evocative actions based implementation

2.68 File and Neo4j Graph Database ThoughtNet hypergraph storage backend (which can be generalized to store any graph relations including WordNet subgraphs constructed by Recursive Gloss Overlap and can be queried in NoSQL)

2.69 Automatic ThoughtNet Hypergraph Creation from Recursive Gloss Overlap classified edges input from environment.

2.70 Software Analytics - Spark computed Cyclomatic

Complexity of codebases from VIRGO linux SATURN

driver generated .dot graph files

2.71 Audio Pattern Mining - Machine learning of patterns in music samples by computing Jensen-Shannon Divergence Distance between Fast Fourier Transform Frequency Probability Distributions of music data

2.72 Patterns within Strings - Longest Repeated Substring Implementation by Suffix Arrays - as a special case pattern in binary encoded time series data is retrieved.

2.73 Locality Sensitive Hashing Implementation - for clustering similar documents and nearest neighbour search.

2.74 ZeroMQ CLI client and server frontend for concurrent requests to NeuronRain

2.75 Kafka Message Queue backend for AsFer

datasource abstraction which includes KingCobra request-reply disk persisted queue.

2.76 Boyer-Moore Algorithm implementation for finding Majority element in Streaming Sequences.

2.77 Graph Mining - GSpan algorithm implementation for graph substructure mining - for extracting patterns in text-graphs and ThoughtNet/EventNet Graphs.

2.78 ThoughtNet Hypergraph Evocation based Recommender Systems Implementation

2.79 Software Analytics - by Deep Learning (Convolution, RecurrentLSTM, RecurrentGRU, BackPropagation) from system performance logs

2.80 Polynomial representation of text strings and distance measures based on it (L2 norms and Jensen-Shannon divergence) - alternative to edit distance

2.81 Polynomial Reconstruction - Error correcting

codes - BerlekampWelch decoder implementation

2.82 Approximate polynomial time 3SAT Solver -
solves 3SAT with high probability depending on error of
least squares algorithm (LSMR/LSQR/CVXOPT)

2.83 Java Spark Streaming and ETL for analyzing
generic arbitrary URL data - extracts text from any URL
with Jsoup and does Spark Dstream and RDD
transformations and actions on it.

2.84 Deep Learning - Convolution Networks +
Backpropagation Multilayered Neural Networks
implementation for image pattern recognition(PILlow
imaging library support).

2.85 Deep Learning - Recurrent Neural Networks -
Gated Recurrent Unit implementation

2.86 Unified Streaming Abstract Generator ETL
pipeline (Jsoup + Java Spark Streaming

Parquet/Kafka/Cassandra/Hive/Logfiles + Java/Python

Spark transformations and Loading)

2.87 Cloud implementations of recursive gloss overlap graph intrinsic merit, bitonic sort have been Benchmarked with Spark 2.1.0 and HiveServer2 Metastore is supported

2.88 Recursive Lambda Function Growth - Random Walk based intrinsic merit computation by growing recursive lambda function composition tree for each random walk in Recursive Gloss Overlap Graph of a text

2.89 Korner Graph Entropy - an alternative Quantitative Intrinsic Merit measure for Recursive Gloss Overlap graph of a text

2.90 Use cases for deploying NeuronRain

2.91 Graph Theoretic alternative intrinsic measures for Recursive Gloss Overlap Graph based on Kleinberg small world lattice, Cheeger Constant

2.92 Index implementation for crawled documents with LocalitySensitiveHashing-Redis key-value store and ThoughtNet Hypergraph as backends

2.93 Small World Indexless Hyperball web crawler - based on Stanley Milgram, Kleinberg lattice Small world experiments and Hyperball algorithm.

2.94 Recursive Lambda Function Growth - maps text to a neural network - Graph Tensor Neuron Network Intrinsic Merit which is amalgamation of Graph Neural Network and Tensor Neuron Network for evaluation of lambda function composition tree for each random walk in Recursive Gloss Overlap Graph of a text - supports both WordNet and ConceptNet

2.95 ThoughtNet Index and LSH Index Query from GUI and Rank text by Graph Tensor Neuron Intrinsic Merit and Korner Entropy

2.96 Unguided Text Summarization - based on
Unsupervised Recursive Gloss Overlap graph core number
classifier: traversal of k-core subgraph and sentences
related to prominent core number class vertices

2.97 ConceptNet 5.6 Python RESTful Client
implementation (a potential alternative to WordNet) -
contains a Common Ancestor algorithm implementation
for ConceptNet distance based on /related ConceptNet
(word2vec word embedding for similarity) REST endpoint

2.98 Unsupervised text classifier by ranking based on
Betweenness Centrality, Closeness Centrality, Degree
Centrality of the word vertices in Recursive Gloss Overlap
graph of texts.

2.99 Scheduler Analytics from Deep-learnt process
performance data for VIRGO Linux Kernel Scheduler
(kernel_analytics VIRGO module)

2.100 Expander graphs - Graph Density (regularity lemma) and Bose-Einstein Condensation Intrinsic Fitness merit measures for Recursive Lambda Function Growth definition graphs of a text

2.101 MAX3CNFSAT ranking of intrinsic merit which unifies outcomes of all intrinsic merit algorithms as variables in a satisfiability CNF and approximately solves it by least squares solver.

2.102 Support Vector Machines Python Implementation

2.103 Computational Geometric Factorization - Spark Implementation of Optimized Parallel Local Binary Search of Pixelated Hyperbolic Arc Tile Intervals for Factor points – found to be faster than Bitonic K-MergeSort based Factorization in dual core single node Spark cluster benchmarks

2.104 Compressed Sensing - Compressed Sketch of an image bitmap - by Moore-Penrose Pseudoinverse

2.105 Non-trivial Proof-of-Work (recreate hash till it has 2 leading ff in hex) and Boost UUID hash (unique id concatenated with denomination) for fictitious Electronic Currency implementation in KingCobra (named “Neuro”)

2.106 Neuro currency C++ move semantics - Option to choose between std::move and std::forward of rvalue reference + std::move in zero-copy Perfect Forwarding of Neuro currency message over cloud

2.107 Complement Function Map construction - Diophantine Representation by SymPy Sum of Four Squares Diophantine Solver - for

<https://arxiv.org/abs/1106.4102v1>

2.108 Transactional and Secure Neuro Currency Perfect Forward - Neuro cloud std::move has been

openSSL enabled and done in Python transaction manager
boundary begin()/commit()

2.109 Implementation of Locality Sensitive Hashing
based Algorithm for searching Unsorted Lists by
dimension lift of one dimensional numerical dataset
(similar to Mercer polynomial kernel)

2.110 Socket Streaming Generic Server Decorator for
Streaming Analytics - iterator function has to be
implemented for a datasource and must be decorated by
this Generic Socket Server - userspace counterpart of
streaming kernel analytics in VIRGO64

2.111 Sequential Optimization algorithm for
Computational Geometric Factorization

2.112 Integer Factorization by Brahmagupta/Pell
Diophantine Equation Solver

2.113 Computational Geometric Factorization -

Approximate and sometimes exact Integer Factoring by
Hardy-Ramanujan Ray Shooting Queries

2.114 Recursive Lambda Function Growth - Rich Club
Coefficient and Simple Cycles based graph tensor neuron
intrinsic merit

2.115 Computational Geometric Factorization -
Approximate Integer Factoring by Hardy-Ramanujan-Prime
Number Theorem, Cramer, Baker-Harman-Pintz, Zhang
prime factors estimate ray shooting queries

2.116 ThoughtNet and Recursive Lambda Function
Growth integration - ThoughtNet hypergraph edges are
weighed either by SentiWordNet or Graph Tensor Neuron
Network intrinsic merit

2.117 Epsilon bias non-uniform choice of literals for
unequal number of variables-clauses and New efficiency
measure for LSMR/LSQR SAT solver - real parity

2.118 Scheduler Analytics - Deep Learning

(GRU,ConvNet,BackPropagation,LSTM) Scheduler Class

analytics for process performance stream dataset created

from Psutils and Graphic performance stream dataset

(stream of images) created from Perf/BPF

2.119 Social Network Analysis - Facebook Graph

REST API Analytics on Python 3.4

2.120 Quantum NC algorithm for Computational

Geometric Factorization - counterpart of classical PRAM-

NC k-mergesort/segment-tree Computational Geometric

Factorization

2.121 Deep Learning - Reinforcement Learning

Recommender Systems and Recursive Lambda Function

Growth integration - Recommender systems hypergraph

edges are weighed either by SentiWordNet or Graph

Tensor Neuron Network intrinsic merit

2.122 Spark Structured Streaming (Java 1.8 + Spark 2.4.x) for advertisement analytics - periodic word, count, timestamp information in DataFrames - windowed and non-windowed (timestamped)

2.123 Computational Geometric Factorization
Algorithm based on Parallel construction of Wavelet Tree
on PRAMs/Multicore

2.124 Latent Semantic Analysis/Indexing of term-document matrices and Low Rank Approximation

2.125 Software Analytics - Userspace Runtime
FlameGraph (along with Perf) and
Valgrind/Callgrind/Kcachegrind Call Graph creation

2.126 Software Analytics - Graph Mining of linux
kernelspace SATURN Control Flow Graph and userspace
Valgrind/Callgrind/Kcachegrind Call Graph DOT files by
Gspan algorithm

2.127 Streaming Algorithms – Approximate Counting implementation

2.128 Streaming Algorithms – Distinct Elements implementation

2.129 Software Analytics – GraphX/GraphFrames
Parallel Graph Processing support for computing
Cyclomatic complexity measures of code CFG.

2.130 Merit of Large Scale Visuals by ImageNet
based ImageGraph and VideoGraph EventNet Tensor
Products Algorithm Implementation – objects in frames of
video are annotated by Keras with Theano backend

2.131 Music Pattern Mining – Histogram timeseries
and Notes Onset Detection – sequence of notes in music
are deemed to be strings in a formal language

2.132 Social Network Analysis – People Analytics and
HR Analytics of Social and Professional Profiles by

Recursive Lambda Function Growth Graph Tensor Neuron
Intrinsic Merit, Log Normal Least Energy Intrinsic Merit
and Profile vertex degree based Experiential Intrinsic
Merit

2.133 Machine Translation - by mapping vertices of
Text Graph in Recursive Lambda Function Growth
implementation from English to any other Natural
Language - translation REST API in
PyDictionary/GoSlate/Googletrans invoked - sentence
formation is by graph traversal

2.134 Software Analytics - Deep Learning
(GRU,ConvNet,BackPropagation,LSTM) analytics for
systemwide performance based on Psutil
CPU/Disk/Memory load statistics - implements a
NeuronRain usecase

2.135 Audio/Music Pattern Mining - Complexity

Analysis of Intrinsic Merit of Audio/Music as String of Notes over octave alphabets – Minimum Description Length, Entropy, Minimum DFA etc.,

2.136 Pattern mining large scale visuals – Urban Planning/GIS Analytics - Convex hull analytics of GIS Images

2.137 Intrinsic Merit of Text analyzed as sequence of integers and Ramsey coloring of text graphs

2.138 NeuronRainApps – NeuronRain Usecases - Analytics Driven Drone Autopilot Online Shopping Delivery (requires DroneSDK-Python and Python 3.6+)

2.139 Software Analytics - /proc/sched_debug support for Scheduler Analytics - stream of processor runqueue dictionaries which can be analyzed by streaming and machine learning algorithms

2.140 Software Analytics + Streaming Analytics

integration - Streaming Abstraction - Operating System

Statistics as data stream which is analyzable like any other BigData Source

2.141 Intrinsic Merit of Music - Music Synthesizer
from DFA (State Machine), Mathematical Functions and Random Samples

2.142 Intrinsic Merit of Music - Mel Frequency
Cepstral Coefficient analysis

2.143 Streaming Set Partition Analytics - Pattern
mining Stream of histograms by adjusted rand coefficient and adjusted mutual information (text word frequency histograms, separate chaining hashtables, voting based on LSH/separate chaining hashtables, business intelligence histograms etc.,)

2.144 Set Partition to Tile Cover - Finding Lagrange
Four Square Tile Cover of a rectangle from buckets of a

Set Partition

2.145 Merit of Large Scale Visuals - Extraction of EventNet causality graph from a Video

2.146 Medical Imageing Analytics - Pattern recognition in medical images (ECG,MRI,Scans) for diagnosis

2.147 Streaming Set Partition Analytics - Comparing Set Partitions by Partition Rank measure in SymPy API (implemented In GRAFIT course material as an example - Hashing Dynamic Sets and OS Scheduler Timer - complements previous streaming set partition analytics implementation In AsFer - alternative to rand index and mutual info)

2.148 Topological Handwriting Recognition – Deep Learning Convolution Backpropagation – Approximate DP Polynomial deformations from handwriting Image contours

and distance similarity measure between DP polynomials

2.149 Intrinsic Merit of Music - Learning Weighted Deterministic Finite State Automaton from strings of musical notes (Python 3.7)

2.150 People Analytics - Tenure Histogram model of a professional profile - Partition Rank of Tenure Histogram Set Partition

2.151 People Analytics - Tenure Histogram vector model of a professional profile - Kendall-Tau Rank Correlation Coefficients of Tenure Histogram Vectors

2.152 Urban Planning/GIS Analytics - Patch Analysis of Remote Sensing GIS Imagery of Urban Sprawls

2.153 Image Analytics - ImageNet Keras-Theano - Scikit Learn Random Forest Classification wrapper for images

2.154 Image Analytics - ImageNet Keras-Theano -

Unsupervised Recursive Gloss Overlap classifier based on ImageNet

2.155 Fraud Analytics - complements low level wireless LAN fraud analytics in US

Bmd - clustering and PCA based credit card financial transactions dataset analysis and Pandas Correlation Coefficient

2.156 Intrinsic Merit of Audio - Recursive Lambda

Function Growth Merits and Core Number based Dense Subgraph Classification of Speech text Recognized from audio

2.157 Intrinsic Merit of Music - Music Genre Classification based on pairwise earth mover distance between MFCCs of music waveforms

2.158 Urban Planning/GIS Image Analytics - scikit learn extract_patches_2d() integration

- 2.159 People Analytics - Kaggle LinkedIn DataSet Analysis - Experiential and Degree Intrinsic Merit
- 2.160 Time Series Analysis - ARMA and ARIMA implementations - for stock quotes example data
- 2.161 Time Series Analysis and Fraud Analytics - Credit Card Transactions Dataset
- 2.162 GIS Remote Sensing Image Analytics - Image Segmentation
- 2.163 Computational Geometric Factorization - Tile Search Optimization - Spark 2.4.3 and QuadCore benchmarks
- 2.164 Named Entity Recognition - by majority PoS of Ontology Semantic Paths between PoS annotated and unannotated words
- 2.165 Intrinsic Merit of Text - ConceptNet Text Graph Implementation and Graph Complexity measures -

alternative to WordNet Text Graph

2.166 Social Network Analysis - People Analytics -
PIPL.com python API integration - Syllable based name
clustering

2.167 Social Network Analysis - People Analytics -
Contextual Name Parsing and Syllable Vector compression

2.168 People Analytics - Set Partition based
Electronic Voting Machine(EVM) implementation and a
Drone EVM usecase

2.169 Topological Handwriting and Face Recognition
- Product Homotopies

2.170 Economic Intrinsic Merit - Gravity Model of
Volume of Trade and GDP as fitness measure

2.171 People Analytics - Set Partition Analytics
based Drone Electronic Voting Machine (partial
implementation - has dependency on Drones)

- 2.172 People Analytics - Handwriting Recognition -
Contour Homotopies - Matplotlib Rasterization
- 2.173 People Analytics - Handwriting Recognition -
Inner Product Space of Contour Interpolated Polynomials
- 2.174 People Analytics - Election Analytics - Set
Partition Electronic Voting Machine Streaming analytics
by Histogram distance measures
- 2.175 People Analytics - Drone Electronic Voting
Machine - JSON persistence of EVMs - Datasource for
Voting Analytics
- 2.176 People Analytics - Drone Electronic Voting
Machine - Pseudorandom Majority Voting Balls-Bins
Simulation and Voting Analytics
- 2.177 Set Partition Analytics - Set Partition to Tile
Cover Reduction by Computational Geometric
Factorization - Least Squares Approximate Solution

(LSMR) for the underdetermined linear equations

2.178 Apart from AstroInfer repositories, GRAFIT course materials have some pedagogical analytics: Async IO (suitable for Drones) and cloud implementations based on Python Spark MLLib - Advertisement Analytics by PageRank and Collaborative Filtering, PrefixSpan Astronomical Analytics of Celestial bodies, FP-Growth frequent itemset analytics for OS Scheduler and a design of Earliest Deadline First Worst Case Execution Time OS Scheduler - AstroInfer codebase might import these cloud implementations

2.179 People Analytics - Election Analytics - Drone Electronic Voting Machine - Streaming Boyer-Moore Majority voting

2.180 People Analytics - Drone Electronic Voting Machine - Streaming Voting Analytics - Bertrand Ballot

Theorem Approximation of Majority Voting

2.181 Astronomical Pattern Mining - Rule Search

Script upgrade for Maitreya Dreams Swiss Ephemeris text client maitreya 8.0.1 (maitreya8t)

2.182 Merit of Large Scale Visuals - Python 3.x

Tensor Rank intrinsic merit of EventNet of a video - measure of connectedness of a Video (Video mapped to EventNet Causality Graph) - Tensor Decomposition of Video 3-way EventNet Tensor Product to Rank-1 tensor components by Tensorly - Slicing for footages of unequal frames

2.183 Computational Geometric Factorization - Quadcore Python 3.6,3.7,5,3.9.0 Upgrade Benchmarks - 60 bits, 512 bits and 1000+, 2000+, 4000+, 18000+ and 20000+ bits integers (1024, 1067, 1213, 1243, 2014, 2037, 2050, 2060, 2061, 2063, 4960, 18178, 18181,

20437, 20523, 20610 bits) - randomly chosen, smooth and hard

2.184 All NeuronRain Python 2.7 Source Files can be upgraded to PEP8 and Python 3.x by autopep8 and 2to3-2.7 utilities. Python 3.x is faster than Python 2.x and has many upgraded builtins including range(), tuned python malloc, async IO. Spark Factorization and EventNet Video Tensor Product implementations in NeuronRain have been upgraded to Python 3.x for compatibility and performance.

2.185 Complement Diophantines - Python 3.7.5 upgrade and Least Squares Vandermonde Polynomial Fit Learning

2.186 People Analytics - Drone Electronic Voting Machine - Python 3.7.5 upgrade and Paper ballot shuffle simulation (and theory on paper currency unique id non-

digital one time password, Money Trail - Neuro
cryptocurrency transactional EventNet Ledger as
Expander Graph and Random Walks in it - Pseudorandom
Extractor and Generator for Unique Identification)

2.187 Chaos Pseudorandom Generators - Python 3.7.5
1-dimensional Binary Cellular Automaton, Mandelbrot set,
Verhulst Logistic and Lehmer-Palmore PRGs
implementation

2.188 Compressed Sensing - Python 3.7.5 Alphabet-
Syllable vectorspace embedding and similar name
clustering in Social Networks and People Analytics

2.189 Leaky Bucket Timeseries Analyzer
Implementation - Python 3.7.5

2.190 PAC Learning - Python 3.7.5 upgrade and
Patterns in Primes

2.191 Text Compression, Compressed Sensing,

Syllable boundaries decompression implementation -

Python 3.7.5 upgrade

2.192 Social Network Analytics - LinkedIn

Connections Analysis, Name filter and Python 3.7.5

upgrades

2.193 GIS and Urban Sprawl Analytics - Image Segmentation and Face components statistics

2.194 GIS and Urban Sprawl Analytics - Delaunay Triangulation and Voronoi diagram of Segmented Image, Multi Agent 4-coloring of Urban Sprawl FaceGraphs for Fair Division of resources

2.195 Interview Algorithm - Spark Recursive Gloss Overlap Implementation - Python 3.7.5 upgrade and Quadcore Benchmarks

2.196 People Analytics - Human Resource Analytics - Recursive Lambda Function Growth Intrinsic Merit of

**Name-filtered Profile - Chaotic Hidden Markov Model of
Tenures/Attrition and Martingales**

**2.197 Computational Geometric Factorization and Set
Partition to Lagrange Four Square Theorem Square Tile
Cover Reduction - Randomization**

**2.198 Intrinsic Merit of syllable hyphenated text and
its audio - Waveform Distance of Audios as Originality
merit measure, Acoustic Distance of Strings, Name
Similarity, Music Clustering - alternative to Levenshtein
edit distance of strings**

**2.199 People Analytics - Human Resource Analytics -
Chaotic Hidden Markov Model of Tenures/Attrition -
implementation - JSON persistence and Weighted
Automata model**

**2.200 Recursive Lambda Function Growth - local
lookup alternatives for machine translated multilingual**

textgraphs and proper noun filtering

2.201 People Analytics - LinkedIn DataSet - Tenure

Statistics

2.202 Word Embedding on a Vectorspace - Spark

SkipGram Log Likelihood - Text Similarity, Dissimilarity
and Merit Measure of Originality (in GRAFIT)

2.203 People Analytics - R - Ricker Logistic -
Population dynamics of urban sprawls (in GRAFIT)

2.204 People Analytics - Social Network Analysis -
CoronaVirus2019 analyzer - Statistics in R - Fitting a
Probability Distribution (in GRAFIT)

2.205 People Analytics - Social Network Analysis -
CoronaVirus2019 analyzer - Chaotic pandemic model,
Correlation coefficients in R (in GRAFIT)

2.206 People Analytics - Social Network Analysis -
CoronaVirus2019 analyzer - Linear Models in R - Linear

and Logistic Regressions (in GRAFIT)

2.207 People Analytics - Social Network Analysis -
CoronaVirus2019 analyzer - CAGraph Logit (in GRAFIT)

2.208 Intrinsic Merit of Academic Publications -
Bibliometrics - SkipGram Word2Vec of publication full
text - Concept Cloud Wordle - an alternative to H-index,
First Order Logic, Monoidal Categories, Girard Geometry
of Interactions (Gol), Sequent Calculus, ProofNets, TOP
Categories, Shell Turing Machines and Kernel
Lifting, Lambda functions and Beta Reduction, Meaning
representation (in GRAFIT)

2.209 People Analytics - Theoretical (Drone)
Electronic Voting Machines - Population Count - Number
of 1s (Decimal parity) in a bitstream - Various algorithms
- Parallel computation in Spark (in GRAFIT)

2.210 Bibliometrics - Patents as ProofNets and Search

- Novelty detection and Originality merit measure for Patent search, Latent Semantic Analysis, Low Rank Approximation (in GRAFIT)

2.211 Fraud and Cybercrime analytics - Faraday Cageing for Van Eck Phreaking, Analyzing Userspace Phishing/Spoof/Spams, Kernel Ftrace callgraph analyzer, Address book and dictionary attacks, Bayesian Filters, ERSIR, SIS and Cellular Automaton Graph models of Cybercrime transmission - Datasets for CoronaVirus 2019 Pandemic R analyzers could be replaced by cybercrime Susceptible-Infected-Recovered random graph botnet data.

2.212 Fraud and Cybercrime analytics - Digital Watermarking of Large Scale Visuals in OpenCV (in GRAFIT)

2.213 Numeric Compression - Sublogarithmic Space Prime power encoding of huge integers by Unique

Factorization Theorem and Computational Geometric

Factorization and its application in CPU instruction sets
(in GRAFIT)

2.214 NeuronRainApps - NeuronRain Usecases - PX4

Micro Air Vehicle SDK Python Asynchronous I/O Drone

Simulator - Drone_MAVSDK_Client and

Drone_MAVSDK_Server -requires PX4 Firmware +

JMAVSIM + MAVSDK-Python + Python 3.7.5 (in GRAFIT)

2.215 NeuronRainApps - NeuronRain Usecases -

Online Shopping Delivery - PX4 Micro Air Vehicle SDK

Python Asynchronous I/O Simulator - requires PX4

Firmware + JMAVSIM + MAVSDK-Python + Python 3.7.5

2.216 Graph Analytics and Drone Navigation - A* (A-
Star) Best First Search Pathfinding and Motion Planning
Algorithm Implementation for Drones and generic Graph
search (in GRAFIT)

2.217 GIS analytics - OpenCV Image Contours,
Contour graphs and Segmentation for Drone Obstacle
Avoidance and Motion Planning

2.218 Merit of Large Scale Visuals and Music -
OpenCV Image Contours and Segmentation for Medical
Imageing and Intrinsic merit of music - (Functional)
Magnetic Resonance Imageing (MRI and fMRI)

2.219 GIS Analytics and Drone Autonomous Delivery -
Drone Obstacle Avoidance by 3D Navigation on Airspace
defined by Road Geometry (Google Roads REST API and
Drone Swarms example)

2.220 People Analytics - Social Network Analysis -
CoronaVirus2019 Analyzer - Exponential Fit in R

2.221 Generating all possible permuted strings of an
alphabet

2.222 Fibonaccian Search - alternative to binary

search in Computational Geometric Parallel Planar Point Location Factorization and other BigData queries

2.223 People Analytics - People profiles as Tensors - Chaotic Hidden Markov Model of Tenure Transitions - TensorFlow integration for conversion of People profile data to Tensors

2.224 People Analytics and Compressed Sensing - Alphabet-Syllable Vectorspace Embedding Distance implementation for Strings - Strings as Tensors - alternative to Levenshtein Edit distance

2.225 Merit of Large Scale Visuals (LSVR) - TensorFlow + Keras Backend - alternative to Theano + Keras Backend

2.226 EventNet Python Parser - Events as Tensorflow Tensors - EventNet vertices and edges are parsed from text files and converted to Tensorflow Tensors

2.227 People Analytics - Alphabet Vectorspace

Embedding - Earth Mover Distance between String

Syllable Tensors

2.228 Graph Search and Analytics - NetworkX Graph

Edit Distance - Inexact Graph Matching - Graph mining
of dissimilar graphs

2.229 Software Analytics - Cyclomatic Complexity
Program Analyzer - Graph Edit Distance - Inexact Graph
Matching for various types of Program analysis graphs
(CFG, Slice dependency, Callgraph)

2.230 Text Analytics - Word Sense Disambiguation -
NeuronRain implementation of Lesk's algorithm -
alternative to NLTK and PyWSD

2.231 People Analytics - HR and Talent Analytics -
Domain Specific Dictionary - LinkedIn Dataset analytics in
Spark 3.0.1+ Hadoop 3.2

2.232 People Analytics - HR and Talent Analytics - Chaotic Hidden Markov Model - Career Transition Score of a Profile - Weighted Automaton of Career Transition

2.233 People Analytics - HR and Talent Analytics - Chaotic Hidden Markov Model - Career Polynomial of a Profile and Inner Product Space Distance Similarity between 2 people profiles

2.234 Software Analytics and People Analytics - Program Analyzers in NeuronRain could also serve as People Analytics tools specific to Professional profiles of People in Information Technology domain thus facilitating automated recruitment of talent - Cyclomatic complexity analyzers, Userspace and Kernelspace Call graph analyzers, SATURN kernel CFG analyzers could be useful in analyzing

complexity of any opensource code

repository

2.235 Bibliometrics - Merit of Academic Publications -
Transformers Attention Model implementation (in GRAFIT)

2.236 Recursive Lambda Function Growth -
Transformer Attention Model - Textgraph Degree
attention

2.237 Named Entity Recognition - Transformers
Attention Model - Textgraph degree attention

2.238 Recursive Lambda Function Growth Textgraph
as EventNet Causality Graphical Event Model -
implementation

2.239 Recursive Lambda Function Growth EventNet
GEM - CoronaVirus 2019 GEM usecase

2.240 Recursive Lambda Function Growth EventNet
GEM - 2 different research articles collated

2.241 Digital Watermarking for both Image and Video
by CV2 overlay

2.242 ThoughtNet Evocation of Thought Hyperedges
by Set Intersection of Hypervertices

2.289 Set Partition Analytics - Set Partition to Tile
Cover Reduction by Computational Geometric
Factorization - CVXOPT GLPK Integer Linear
Programming Exact Solution for the underdetermined
linear equations

2.290 Merit of Large Scale Visuals - GIS and Urban
Sprawl Analytics, CoronaVirus 2019 Analyzer, Histogram
Partition Distance Similarity of images

2.291 Drone Electronic Voting Machine - Set Partition
Analytics electronic_voting_machine() invocation - PX4
SITL JMAVSIM simulation - Geocoding integration for
address to longitude-latitude translation

2.292 Drone Electronic Voting Machine - Set Partition

Analytics electronic_voting_machine_asynchronous()

wrapper definition and invocation - PX4 SITL JMAVSIM

simulation - Fictitious address

2.293 Drone Online Shopping Delivery -

NeuronRainApps - Refactoring for ConvexHull and GIS

analytics variables read over

socket stream

2.294 Set Partition Analytics - Complementary Set

Partitions

2.295 GIS Urban Sprawl Analytics - NASA VIIRS

NightLights Urban Area and Population Estimation - from

training data of contour area to square kilometre area

and average population density (GeoTIFF lookup and map

projections have been ignored as they are data format

specific)

2.296 GIS Urban Sprawl Analytics - NASA VIIRS

**NightLights 2012 and 2016 Urban sprawl distance
similarity and Urban area rankings**

2.297 GIS Urban Sprawl Analytics - NASA VIIRS

**NightLights 2016 and 2021 Urban sprawl distance
similarity and Urban Area Rankings**

**2.298 Economic merit - Urban and Rural GDP
estimation from GIS Urban Sprawl Analytics**

**2.299 GIS Weather Analytics (Windy.com layers for
wind,rain,temperature as inputs to drone natural obstacle
avoidance algorithm) and Image grayscale inversion for
the non-urban (rural) complement of urban night time
lights**

**2.300 Large Scale Visuals Streaming NoSQL
Datastore (MongoDB-GridFS) support in
Streaming_AbstractGenerator Facade**

2.301 GIS Urban Sprawl Analytics -
Moments,Centroid,Minimum Enclosing Circle of an urban
sprawl

2.302 GIS Urban Sprawl Analytics - Population
Estimator from Urban Sprawl Contours - usecase
comparative study of few urban sprawl growth pattern
examples

2.303 GIS Weather Analytics and Climate Analytics
(Tensorflow Sequential model for climate datasets, historic
extreme weather events and correlation to celestial n-
body choreography patterns) from OpenWeatherMap
PyOWM and ECMWF CliMetLab

2.304 GIS Urban Sprawl Analytics - a Multiple Agent
Resource Allocation DNFSAT Algorithm

2.305 GIS Analytics - MongoDB GridFS backend
standalone python client implementation

2.306 GIS Urban Sprawl Analytics - Urban Sprawl Extent from NASA SEDAC 2015 GPW 4 Population Count and SEDAC 2020 GPW 4.11 Population Density by contour, convex hull segmentation and minimum enclosing circles

2.307 GIS Urban Sprawl Analytics - Voronoi Tessellation Clustering Diagram implementation engulfing urban sprawl contours and minimum enclosing circles - a rural + urban extent estimator - alternative to neural network and DBSCAN urban extent models

2.308 GIS Urban Sprawl Analytics - Centrality Measures of Urban Sprawl Facegraph

2.309 GIS Urban Sprawl Analytics - Delaunay Triangulation plot - approximation of transportation network between urban areas - an Eulerian circuit and Hamiltonian estimator for Efficient Drone navigation

2.310 GIS Urban Sprawl Analytics - Distance similarity of Voronoi Tessellations (Earth mover distance, Graph Edit Distance, Modularity - community detection) and Measuring Urban Expansion

2.311 GIS Urban Sprawl Analytics - Computation of Urbanization Correlation from Voronoi Shapely Polygon areas (Rural + Urban) and Urban Sprawl Contour areas

2.312 People Analytics - Merit of Large Scale Visuals - Unique identification - Topological Face Recognition - Voronoi Tessellation and Delaunay Triangulation of Faces, their Morphology and Shape Grammar

2.313 People Analytics - Merit of Large Scale Visuals - Unique identification - Topological Face Recognition - Distance similarity between Facial Image Voronoi Tessellations of similar Faces and Facial Landmark Detection - Wasserstein distance between Voronoi Facet

Shapely polygon areas of 2 facial images

2.314 People Analytics - Merit of Large Scale Visuals

- Unique identification - Topological Face Recognition -

Voronoi diagram and Delaunay Triangulation of Facial

images from 68-landmark centroids computed by Dlib face landmarks detection

2.315 People Analytics - Merit of Large Scale Visuals

- Unique identification - Topological Face Recognition -

Netrd Voronoi Facegraph Graph matching distance

similarity measures, Histogram partition distance

similarity (cv2.EMD()) and Voronoi FaceGraph

Isomorphism-Subgraph Isomorphism Similarity (NetworkX) of 2 facial images

2.316 People Analytics - Merit of Large Scale Visuals

- Topological Face Recognition - Voronoi diagram and

Delaunay Triangulation of Facial images from centroids

computed by Dlib face landmarks detection and Voronoi

FaceGraph Isomorphism-Subgraph Isomorphism Similarity

2.317 People Analytics - Merit of Large Scale Visuals

- Topological Face Recognition - Voronoi FaceGraph

Isomorphism-Subgraph Isomorphism Similarity - VF2 and

ISMAGS Isomorphic Subgraphs iteration

2.318 People Analytics - Merit of Large Scale Visuals

- Topological and Graph Theoretic Face Recognition -

Voronoi FaceGraph Isomorphism Similarity - Percentage

similarity

2.319 People Analytics - Merit of Large Scale Visuals

- Topological and Graph Theoretic Face Recognition -

Delaunay Triangulation Mesh isomorphism and Euler

Characteristic

2.320 Software Analytics – Cybercrime analytics -

Graph Isomorphism Similarity of Code Callgraphs and

Control Flow Graphs - Spark 3.0.1 + Python 3.7.5

2.321 Software Analytics - Cybercrime analytics -

Approximate Graph Isomorphism Similarity of Code

Callgraphs and Control Flow Graphs - Spark 3.0.1 +

Python 3.7.5

2.322 GIS Analytics - DBSCAN Density Clustering

implementation

2.323 Software Analytics - Degree Sequence Earth

Mover Distance Similarity of Code Callgraphs and Control

Flow Graphs - Spark 3.0.1 + Python 3.7.5

2.324 GIS Analytics - DBSCAN Density Clustering -

Neural Network Thresholding implementation

2.325 People Analytics - Merit of Large Scale Visuals

- Topological and Graph Theoretic Face Recognition -

Delaunay Triangulation Mesh Bezier Animation by PyVis

2.326 People Analytics - Merit of Large Scale Visuals

- Textual,Topological and Graph Theoretic Face Recognition - Physique Recognition From Quadrilateral mesh isomorphism and Face Distinguisher from ImageNet predictions

2.327 Computational Geometric Factorization - Matplotlib plot of benchmarks - 300 integers from 100 to 400 and 20 integers from 500 to 520

2.328 People Analytics - Social Network Analysis - Bipartite Social Networks (Matrimonial portals, Employer-Employee Professional networks) - Recommender Systems and Sentiment Analysis of Profiles for Ranking Preferences

2.329 GIS Urban Sprawl Analytics - Population Density computation from Contour Pixel color coding

2.330 GIS Urban Sprawl Analytics - Population count computation from HRSL Map Legend

2.331 Economic merit - Neuro Cryptocurrency mint -
Commodity linked Neuro Fictitious Cryptocurrency Proof
of Work Algorithm

2.332 Computational Geometric Integer Factorization -
20523 bits quadcore benchmarks - Factorization as a non-
trivial Proof of Work computation for Neuro
Cryptocurrency mining rig

2.333 Set Partition Analytics - Set Partition-
Rectangular Area to Factorization reduction and Integer
Linear Program solution of Frobenius Coin Diophantines
for Neuro Cryptocurrency mining rig Proof of Work, One
Time Password facility augmented for Voter Received
Encrypted Paper Audit Trail (VREPAT) Conceptual Drone
Electronic Voting Machine

2.335 Fictitious Neuro Cryptocurrency Mining Rig -
Random Integer Partition + Rectangular Area

Factorization + Money Changing Problem Proof of Work -
Digital Watermarking implementation for Visual Neuro
cryptocurrency

2.336 Music Synthesizer - Python 3.8 upgrade and
Fractal implementation

2.337 Music similarity - Dynamic Time Warping
Waveform Distance algorithm implementation

2.338 Music Information Retrieval (MIR) - Learning a
polynomial from music waveform

2.339 Fictitious Neuro Cryptocurrency NP-Hard Proof
of Work - Moving Mount Fuji - Single Bin Sorted LIFO
Histogram - Towers of Hanoi - implementation

2.340 Music Information Retrieval - AI Music
Synthesis from Sum of Damped Sinusoids

2.341 GIS Urban Sprawl Analytics - SEDAC Urban
Expansion Probabilities Projection for Year 2030 - a

casestudy of urban sprawls

2.342 GIS Urban Sprawl Analytics - Maximum Population Density and Area-Population Ratio as arguments

2.343 Set Partition Analytics - Neuro Cryptocurrency Towers of Hanoi NP-Hard Proof-of-Work - Set Partition Histogram to Lagrange Sum of Four Square theorem square tile cover of an Almost-Square rectangle by Integer Factorization

2.344 Kernel lift random walk in tree of TOP Category Shell Turing Machines and UNIX Shell Game Example

2.345 Intrinsic Merit of Texts - TAC 2010 code opensourced -

2.346 People Analytics - Social Network Analysis - CoronaVirus2019 analyzer - Recomputed Spearman,

Kendall Tau, Pearson Correlation coefficients of COVID19 dataset and Verhulste-Pearl-Reed Logistic in R - in GRAFIT repositories

2.347 GMSH Trimesh and Quadmesh - GEO OPT and MSH files from GMSH FLTK

2.348 Merit of Large Scale Visuals - Trimesh and Quadmesh generation from an image by GMSH

2.349 GIS Urban Sprawl Analytics - Polya Urn Urban Growth Model implementation for 4-colored Urban Sprawl Contour Segment Facegraphs

2.350 People Analytics - Social Network Analysis - CoronaVirus2019 analyzer - Recomputed Spearman, Kendall Tau, Pearson Correlation coefficients for revised COVID19 dataset (Lancet,WHO estimates) and corrections to Verhulste-Pearl-Reed and May Logistic code in R - in GRAFIT repositories

2.351 GIS Analytics - Archaeology - an archaeoastronomical and computer vision analysis of MODIS GIS image of Adam's Bridge (Ram Sethu)

2.352 Merit of Large Scale Visuals - Face recognition by Approximate Topological Match of GMSH Quadmesh images

2.353 Merit of Large Scale Visuals - Archaeology - ImageNet predictions from multiple models - Rebus Topological Script Recognition - Dholavira Signboard example deciphered by ResNet50 and ResNet50V2

2.354 Merit of Large Scale Visuals - Astronomy Analytics - DBSCAN Clustering and Segmentation analysis of sky imageries for 2 high magnitude seismic events

2.355 Merit of Large Scale Visuals - Archaeology - Rebus Decipherment of Indus Unicorn Seals and their validation by extracting common subgraphs in ImageNet

predictions textgraphs (GSpan Frequent Subgraph Mining algorithm)

2.356 Merit of Large Scale Visuals - GIS Urban Sprawl Analytics - Automatic Delineation of Urban Growth Boundaries from Transportation Network GIS - KMeans Clustering of Contour Polynomials

2.357 Computational Geometric Factorization - Quadcore benchmarks - Spark 3.0.1 + Python 3.7.x - Nanoseconds perf counter

2.358 Astronomical Dataset Analytics - Planetarium Ephemeris Search

2.359 GIS Urban Sprawl Analytics - Minimum Spanning Forest,HITS Facegraph complexity measures for Automatic Urban Delineation

2.360 Medical Imageing and Music Information Retrieval (MIR) - fMRI imagery analysis for music-evoked

autobiographical memories

2.361 Intrinsic Merit of Music - Kolmogorov

Complexity approximation by Normalized Compression
Distance (NCD)

2.362 People Analytics - Unique Identification -
Fingerprint recognition

2.363 Intrinsic merit of Music - Music Information
Retrieval - AI Music Synthesis from Weierstrass Fourier
Fractal

2.364 GIS Urban Sprawl Analytics - Moran's I
measure of Urban Sprawl Dispersion

2.365 People Analytics - Social Network Analysis
(Professional) - Chaotic Hidden Markov Model of Career
Transition - JSON persistence to MongoDB

2.366 Space Archaeology - Edge detection for
Airborne-Satellite GIS imagery (LIDAR,SAR,LandSat) by

Canny Edge Detector

2.367 GIS Urban Sprawl Analytics - Automatic
Delineation of Urban Growth Boundaries by KMeans
Clustering of Contour Polynomials and Canny Edge
Detector

2.368 Astronomical Dataset Analytics - Planetarium
Ephemeris Search - SkyField implementation
2.369 Astronomical Dataset Analytics - Planetarium
Ephemeris Search - Longitude-Latitude and AstroPy
support

2.370 Astronomy Dataset Analytics - Angular
separation of solar system planets during Extreme
Weather Events on Earth - an N-Body gravitational pull
correlation study

2.371 Majority Voting Versus Query complexity model
of Intrinsic Merit of Texts-Audio-Video-People - all

implementations of Intrinsic Merit compute Merit()
function by queries

2.372 Astronomy Dataset Analytics - NASA JPL
Horizons Ephemeris Lookup Service support and Hubble
Ultra Deep Field imagery RGB analytics

2.373 Astronomy Dataset Analytics - Computation of
N-Body Gravitational Acceleration for Solar System bodies
on days of Extreme Weather Events (Earthquakes and
Hurricanes)

2.374 Astronomy and Cosmology Datasets Analytics -
Hubble eXtreme Deep Field imagery RGB analysis and
Distance similarity of N-Body gravitational accelerations

2.375 Astronomy and Cosmology Datasets Analytics -
Wilkinson Microwave Anisotropy Probe-Cosmic Microwave
Background (WMAP CMB) and N-Body gravitational
accelerations from arbitrary vantage points

2.376 NeuronRainApps - Wikipedia Textgraph

Question-Answering Bot - WordNet Walk on answer textgraph

2.377 Economic Merit and People Analytics - Mechanism Design of Multi-round Majority Voting derived from Simultaneous Ascending Auction - for Drone EVM usecase implementation in NeuronRain

2.378 Merit of Large Scale Visuals - Archaeology - Rebus Decipherments of Indus Pashupathi Seals and mining frequent subgraphs from predictions

2.379 Astronomy and Cosmology Datasets Analytics - N-Body gravitational accelerations computed for USGS (1900-2012) 8+ magnitude Earthquakes and NOAA HURDAT2 (1851-2012) North Atlantic Hurricanes Datasets

2.380 GIS Urban Sprawl Analytics - Facebook (Meta)

High Resolution Settlement Layer (HRSL) Population

Density Maps - GDAL GeoTIFF to JPEG format Translation

2.381 GIS Urban Sprawl Analytics - Facebook (Meta)

High Resolution Settlement Layer (HRSL) Population

Estimation from GDAL-Rasterio Georeferencing

2.382 GIS Urban Sprawl Analytics - Facebook (Meta)

High Resolution Settlement Layer (HRSL) and LandSat 9

OLI2-TIRS2 GDAL-Rasterio Georeferencing - Window read

2.383 GIS Urban Sprawl Analytics - Facebook (Meta)

High Resolution Settlement Layer (HRSL) and Global

Human Settlement Layer (GHSL) GDAL-Rasterio

Georeferencing - Sampling and Mollweide-EPSG

transforms reprojections

2.384 GIS Urban Sprawl Analytics - Facebook (Meta)

High Resolution Settlement Layer (HRSL) and Global

Human Settlement Layer (GHSL) GDAL-Rasterio

Georeferencing - Gini Index of Population and Built-Surface

2.385 GIS Urban Sprawl Analytics - Facebook (Meta)

High Resolution Settlement Layer (HRSL) - Population estimation from GDAL-Rasterio Georeferencing

2.386 GIS Urban Sprawl Analytics - Polya Urn Urban Growth Model - Revised for N color segments

2.387 NeuronRainApps - Autonomous Driving - Obstacle Lattice from LIDAR Point Cloud Data - Python and C++ lattice walk obstacle avoidance usecases

2.388 Computational Geometric Integer Factorization - Python 3.10.4 upgrade - 2232 and 67 bits Quadcore Benchmarks and Mathematica-Pari/GP-FLINT performance numbers comparison

2.389 GIS Urban Sprawl Analytics - Polya Urn Urban Growth Model - Weights Learnt

- 2.390 Intrinsic Merit of Music - AI Music Synthesis
from Sum of Sinusoids - Signal synthesis from librosa
tone()
- 2.391 Intrinsic Merit of Music - AI Music Synthesis
from Polynomials Learnt from training data - PolyFeatures
and Carnatic-Hindustani notes support
- 2.392 Intrinsic Merit of Music - Virtual Piano
Implementation and Music Synthesis from 12-notes octave
- 2.393 Intrinsic Merit of Music - Virtual Piano
Implementation - Carnatic Music Synthesis from 12-notes
octave
- 2.394 Intrinsic Merit of Music - Deep Learnt
Automata and Music Synthesis - a Boolean Composition
and Learning Theory perspective
- 2.395 GIS Urban Sprawl Analytics - Polya Urn Urban
Growth Model - Urban sprawl area computation

2.396 GIS Urban Sprawl Analytics - Comparison of Raster Data Bounding Boxes between 2 dates - Chennai Metropolitan Area Sprawl - GHSL GHS SMOD Degree of Urbanization - R2019A and R2022A

2.397 Complement Diophantines - Lagrange and Barycentric interpolations

2.398 Intrinsic Merit of Music - Synthesized Bach from training music waveforms

2.399 Astronomy and Cosmology Datasets Analytics - Solar system N-Body Pairwise angular separations computed for NOAA HURDAT2 (1851-2012) North Atlantic Hurricanes Datasets

2.400 Computational Geometric Integer Factorization - Multiple Integers - Python 3.10.4 upgrade + Spark 3.0.1 Quadcore Benchmarks

2.401 GIS Weather Analytics and Climate Analytics -

NASA JPL DE421 Ephemeris N-Body analytics integration and correlation to extreme weather events - Syzygies and angular separations of Solar system bodies

2.402 Computational Geometric Integer Factorization - Multiple Integers - Python 3.10.4 upgrade + Spark 3.0.1 Quadcore Benchmarks - Numba JIT optimization - Configurable number of factors - Factorization- SquareRoot-Primality of Consecutive integers

2.403 GIS Urban Sprawl Analytics - GHSL R2019A and R2022A - Comparison of Raster Data Bounding Boxes for Chennai Metropolitan Area Sprawl

2.404 GIS Urban Sprawl Analytics - R2022A - 3D Urban Sprawl Growth Model from BUILT-V and BUILT-S datasets - Commercial segment (Central Business District) Delineation by average building height - Chennai Metropolitan Area Sprawl Example

2.405 Computational Geometric Integer Factorization -
Python 3.10.4,Python 3.7.5,Spark 3.0.1,Spark 3.3.0
Quadcore Benchmarks - Numba JIT enabled for joblib
parallelized tiles creation - Semiprimes,Cunningham's
Number

2.406 Computational Geometric Integer Factorization -
Python 3.10.4 + Spark 3.3.0 Quadcore Benchmarks -
Hardware Square root instruction optimization and
Semiprime Factorization

2.407 GIS Urban Sprawl Analytics - R2022A - 3D
Urban Sprawl Growth Model from BUILT-V and BUILT-S
datasets - Mean,Median,Standard Deviation of Building
Heights - Chennai Metropolitan Area Sprawl Example

2.408 GIS Weather Analytics and Climate Analytics -
Predictions of Extreme Weather Events

2.409 GIS Urban Sprawl Analytics - R2022A - 3D

Urban Sprawl Growth Model from BUILT-V and BUILT-S datasets - Digital Elevation Model from Matplotlib3D - Chennai Metropolitan Area Sprawl Example

2.410 GIS Urban Sprawl Analytics - R2022A - Verhulste Population Growth Model - Population Estimator - Chennai Metropolitan Area Sprawl Example

2.411 MultiFractal Detrended Fluctuation Analysis (MFDFA) - for unearthing Fractal structure within any timeseries data (including Music, Medical Imageing-ECG, Meteorology-Precipitation, Seismology-Tremors, NBody Gravitational Accelerations, Economics-Financial Markets)

2.412 GIS Urban Sprawl Analytics - Urban Sprawl Road Network Graph from OpenStreetMap (OSMnx)

2.413 GIS Urban Sprawl Analytics - Urban Sprawl Road Network Analytics (OSMnx) based delineation for periurban sprawl of Chennai Metropolitan Area - Road

density measures and Cheeger constant - a case study

**2.414 GIS Weather Analytics and Climate Analytics -
MFDFA model of precipitation**

**2.415 Intrinsic Merit of Music - MFDFA model of
Music Waveforms and Piano notes string generation from
mathematical functions**

**2.416 Computational Geometric Integer Factorization -
Python 3.10.6 + Spark 3.3.0 Quadcore Benchmarks -
Optional Square Root Optimization, Refactoring of
Multiple Integer Factorization, Linux Pollard-Rho Factor
Command Benchmarks comparison**

**2.417 Computational Geometric Integer Factorization -
Python 3.10.6 + Spark 3.3.0 Quadcore Benchmarks -
Multiple Consecutive Integers**

**2.418 GIS Weather Analytics and Climate Analytics -
Prediction of Extreme Weather Events based on angular**

separations-NBody gravitational accelerations of few more pairs of solar system bodies from Sequence Mining and Graphics plot of Gravities

2.419 GIS Weather Analytics and Climate Analytics - Prediction of Extreme Weather Events based on angular separations-NBody gravitational accelerations of solar system bodies from Sequence Mining - Python3.x upgrade of Sequence Mining (NeuronRain Green and Antariksh - GitHub and GitLab) and JSON persistence in NeuronRain Research (SourceForge)

2.420 GIS Weather Analytics and Climate Analytics - Prediction of Extreme Weather Events based on angular separations-NBody gravitational accelerations of solar system bodies from Sequence Mining - Arbitrary Celestial Conjunctions and Reading Sequence mined Class Association Rules from JSON

2.421 GIS Urban Sprawl Analytics - Overlay of GHSL
R2022A BUILT-V Visualization and Chennai Metropolitan
Area Map and its clustering

2.422 GIS Urban Sprawl Analytics - Overlay of GHSL
R2022A BUILT-V Visualization and Chennai Metropolitan
Area Map and its DBSCAN clustering

2.423 GIS Weather Analytics and Climate Analytics -
Prediction of Extreme Weather Events from Gaussian
Mixture Model - Multimodal Gaussian fit of the
precipitation timeseries

2.424 Computational Geometric Integer Factorization -
Python 3.7.5 + Spark 3.3.0 Quadcore Benchmarks - Linux
Elliptic Curve Factorization gmp-ecm comparison

2.425 Computational Geometric Integer Factorization -
Python 3.7.5 + Spark 3.3.0 Quadcore Benchmarks - Linux
General Number Field Sieve Factorization (CADO-NFS)

comparisons (multiple permutations and combinations of integers)

2.426 GIS Weather Analytics and Climate Analytics -
Prediction of Extreme Weather Events from Gaussian
Mixture Model - Multimodal Gaussian fit of the
precipitation timeseries - Modes set from N-Body
gravitations

2.427 GIS Weather Analytics and Climate Analytics -
Prediction of Extreme Weather Events from Gaussian
Mixture Model - Multimodal Gaussian fit of the
precipitation timeseries - Gaussian Ensemble Rainfall
Forecast from Timeseries Partition Enumeration

2.428 Granger causality of timeseries data (including
Music, Medical Imageing-ECG, Meteorology-Precipitation,
Seismology-Tremors, NBody Gravitational Accelerations,
Economics-Financial Markets)

2.429 Graphical Event Model (GEM) from Granger causality of timeseries data (including Music, Medical Imaging-ECG, Meteorology-Precipitation, Seismology-Tremors, NBody Gravitational Accelerations, Economics-Financial Markets) - Stock quotes GEM example

2.430 GIS Urban Sprawl Analytics - Maxflow-Minxut Bottleneck measure of OSMnx transport network graph - Overlay of GHSL R2022A BUILT-V Visualization and Chennai Metropolitan Area Map - a road network density analysis of expanded Chennai Metropolitan Area sprawl and mystery boom in class 21 pixel periurban regions of GHSL R2022A.

2.431 Computational Geometric Integer Factorization - Python 3.7.5 + Spark 3.3.0 Quadcore 4093 bits Benchmarks - Self Initializing Quadratic Sieve Factorization and Elliptic Curve Method (SIQS + ECM)

comparison

2.432 String Analytics and Intrinsic Merit of Music -
Fisher-Yates-Knuth Shuffle - permutation catalog of all
possible music waveforms - Mersenne Twister PRNG

2.433 String Analytics and Intrinsic Merit of Music -
Fisher-Yates-Knuth Shuffle augmented by Reservoir
sampling - permutation catalog of all possible music
waveforms - Reservoir sampling function implemented in
C++ is a multipurpose Opinion mining utility for
approximating majority voting on a population (Opinion
polls in psephology approximate majority gate or
compress a majority gate by sampling a fraction of
leaves)

2.434 String Analytics - Variable value swap by XOR
swap algorithm - Multiple return values within a Go
function, Goroutines and Gochannels for synchronization

2.435 String Analytics - String reversals by XOR swap algorithm - definition of separate Go function for reversing a string by XOR swap

2.435 String Analytics and Large Scale Visuals Analytics - Python 3.11 Matrix mirror utility primitive - String mirroring or String topological inversion

2.436 String Analytics and Large Scale Visual Analytics - Python 3.11 String Factorization - Vowelless text compression as a consonant-vowel vectors Matrix product

2.437 Rasterization or rectification of hyperbolic arc and Factor point location in Rust- Sequential version and iterative binary search

2.438 Computational Geometric Factorization - Rayon Parallel iterator Rasterization or rectification of hyperbolic arc and Factor point location in Rust- Sequential and

Parallel versions - a high performance supercomputing
(Nick class) alternative to PySpark cloud Computational
Geometric Factorization (DMRC MapReduce)

AstroInfer is the userspace Big Data Mining and Deep Learning facet of NeuronRain. Initially implemented for mining patterns in astronomical datasets(degrees of astronomical objects viz planets, constellations etc.,) and prediction based on rules and execution of those rules (SourceForge), has been generalized for any dataset (GitHub, GitLab). It is also used in USBmd and VIRGO codebases below for traffic analytics and kernel analytics. Design started in May 2003.

(Latest NeuronRain AsFer research version Code at
sourceforge repository: <http://asfer.sourceforge.net> and

periodically updated Design Details at:

<http://sourceforge.net/p/asfer/code/HEAD/tree/asfer-docs/AstroInferDesign.txt>

(Latest NeuronRain AsFer enterprise version Code at

github repository: <https://github.com/shrinivaasanka/asfer-github-code> and periodically updated Design Details
at:<https://github.com/shrinivaasanka/asfer-github-code/blob/master/asfer-docs/AstroInferDesign.txt>

3. NEURONRAIN - VIRGO - VIRTual Generic Os Linux

Kernel (compatible with 4.1.5 (32-bit) and 4.13.3 (64-bit)

mainline kernels) - Linux Kernel Extensions for cloud -

kernel modules, system calls for cloud rather than at

application level high up the stack with :

3.243 Config file support

3.244 Psuedorandom Generator Based

Loadbalancer

3.245 Kernel space remote execution

3.246 User space remote execution with kernel
upcall and pthread creation of userspace library function
or executable

3.247 Example unit test cases

3.248 Usermode output redirected logging

feature for Kernel upcall to Userspace

3.249 Intermodule Function Invocation in

Kernel Space - through which any machine on cloud can
be completely remote-controlled deep upto board and
hardware cards through function names or commands
sent through `virgo_clone()` calls.

3.250

CPU Pooling Driver and `virgo_clone`

system call - Multi-kernel-threaded VIRGO clouexec

Kernel Driver Module for unrestricted service of

`virgo_clone` or other client requests.

3.251

Memory Pooling Driver (and a key-

value store) and system calls userspace clients for it on

Cloud nodes - `virgo_malloc`, `virgo_get`, `virgo_set`,

`virgo_free`.

3.252

Queueing Driver - that implements a

wrapper over linux concurrent managed workqueue

(CMWQ) and also a native local queue - used for

KingCobra requests queuing with handler invocation. Also

implemented is a standalone kernel queueing service that

listens on requests rather than being forwarded by CPU

and Memory pooling drivers above.

3.253

VIRGO Cloud File Systems Driver

and system calls - implements distributed cloud file system calls and telnet userspace clients for - `virgo_open`, `virgo_close`, `virgo_read` and `virgo_write` of a file on remote cloud node

3.254 All the drivers above for CPU, Memory and File System on cloud have 3 paths each for telnet connection to remote driver kernel server socket and system call connection to remote driver kernel server socket - 1) parameter is executable in userspace 2) parameter is a function name which uses a kernel upcall plugin to execute in userspace 3) parameter is a function name executable in kernel space - configured by a boolean flag with in the driver binaries. Based on this flag either kernel upcall to userspace or kernel intermodule invocation is done.

3.255 A config driver - for exporting config

symbols

- 3.256 Experimental Bakery algorithm kernel module implementation - for synchronization in cloud
- 3.257 Utilities driver kernel module - a universal kernel module with exported utility function symbols that can be invoked across VIRGO Linux subsystems and Linux kernel including EvenNet logging kernel socket client and skbuff kernel socket debugger.
- 3.258 Experimental EventNet driver kernel service module - This listens on incoming EventNet log messages (Vertex and Edge) and writes to EventNet Vertex and Edge text files by VFS write. These files can then be massively processed by the boost::graph or pygraph GraphViz code in AsFer to create DOT files and graphics. Event vertices and edges can be logged by `virgo_eventnet_log()` utility function from any kernel

module on any VIRGO cloud node.

3.259 From the above EventNet graph, a logical time ordering can be obtained on the cloud events and partakers which is for example useful in establishing money trail in KingCobra MAC eletronic money.

3.260 Kernel Analytics Kernel Module - reads the config key-value pairs set by AsFer or any other machine learning software and exports them which can be looked-up in any other kernel module. The key-value config are analytics variables learnt by mining kernel or other logs and objects. With this an adaptive dynamic kernel which changes over a period of time depending on a machine learnt config is obtained. At present an Apache Spark usecase which mines kern.log and exports a config variable through kernel_analytics has been implemented.

3.261 Thus VIRGO has all the requisite

minimum functionalities for a linux variant cloud,
artificially intelligent, operating system

3.262 SATURN Program Analysis Module -

Intercepts linux build and extracts program analysis data in trees. SATURN logs are analyzable by AsFer (e.g null pointers, aliasing, memory etc.,).

3.263 VIRGO is suitable for analytics driven

embedded systems e.g kernelspace IoT

3.264 VIRGO KTLS config kernel module - only in VIRGO_KTLS branch of VIRGO64 repositories

3.265 64 bit version of VIRGO Linux kernel based on mainline 4.13.3 kernel (in a separate repository) is in separate repository and has been found to be better performing and stabler than VIRGO 32bit.

3.266 VIRGO 64-bit has lot of random panics related resolutions related mostly to i915, CIFS drivers

and SMB security. Though functionally similar to 32 bit VIRGO linux, 64 bit kernel is suited for large RAM installations, with CPU level security features and faster than 32 bit binaries. 4.13.3 kernel also supports Kernel Transport Layer Security (KTLS). VIRGO64 repositories implement KTLS setsockopt() in a separate branch - `VIRGO_KTLS`.

3.267 Streaming kernel analytics in VIRGO64 - for live reading of stream of analytic variable-value pairs from remote cloud node over network in `kernel_analytics` module and exporting them as global symbols in local cloud node kernel. Requires a webservice creating a stream of key-value analytics learnt from a dataset. Presently implemented only in VIRGO64

3.268 Kernel Analytics powered PXRC Flight Controller Driver - Kernel support pxrc flight controller

driver from 4.17. Changed PXRC driver in kernel 5.1.4 has been committed to VIRGO64 repositories which imports kernel analytics variables exported by kernel_analytics driver in VIRGO64.

3.269 Kernel Analytics powered UVC Video Driver - Changed UVC Video driver in kernel 5.1.4 has been committed to VIRGO64 repositories which imports kernel analytics variables exported by kernel_analytics driver in VIRGO64.

3.270 Read-Copy-Update in userspace - usecase (in GRAFIT) - wraps VIRGO System calls

3.271 Software Transactional Memory in userspace - Lockfree datastructures - usecase (in GRAFIT) - wraps VIRGO system calls

VIRGO enables viewing entire cloud as a single “logical

“machine” upto hardware level with machine learning support by implementing lowlevel primitives that can be invoked by highlevel cloud OS components viz., OpenStack Neutron. This differentiates from other cloud libraries which are in userspace mostly. AstroInfer together with KernelAnalytics-VIRGO-USBmd-KingCobra make it a Cloud OS kernel with Machine Learning and Analytics abilities. Support for mobile operating systems in cloud are, among other things, long term goals for VIRGO.

(Latest NeuronRain VIRGO research version Code at sourceforge repositories: <https://sourceforge.net/projects/virgo-linux/> , <https://sourceforge.net/projects/virgo64-linux/> and periodically updated Design details at: <https://sourceforge.net/p/virgo64-linux/code/ci/master/tree/virgo-docs/VirgoDesign.txt> , <http://sourceforge.net/p/virgo/>

[linux/code-0/HEAD/tree/trunk/virgo-docs/VirgoDesign.txt](https://github.com/shrinivaasanka/virgo-linux-github-code/blob/master/virgo-docs/VirgoDesign.txt)

(Latest NeuronRain VIRGO enterprise version Code at
github repositories: VIRGO -

<https://github.com/shrinivaasanka/virgo-linux-github-code> ,

<https://github.com/shrinivaasanka/virgo64-linux-github-code> and

periodically updated Design details at:

[https://github.com/shrinivaasanka/virgo64-linux-github-](https://github.com/shrinivaasanka/virgo64-linux-github)

[code/blob/master/virgo-docs/VirgoDesign.txt](https://github.com/shrinivaasanka/virgo64-linux-github-code/blob/master/virgo-docs/VirgoDesign.txt),

[https://github.com/shrinivaasanka/virgo-linux-github-code/bl
ob/master/virgo-docs/VirgoDesign.txt](https://github.com/shrinivaasanka/virgo-linux-github-code/blob/master/virgo-docs/VirgoDesign.txt))

Design started in 2008.

4. NEURONRAIN - USBmd- Linux kernel USB device

driver module for debugging and network analytics :

4.272 Contains a modified version of Linux kernel
mainline USB and USBWWAN broadband modem drivers

with debug statements and functions that capture and dump data transfer buffers in kern.logs.

4.273 These logs are parsed and processed with Apache Spark and other AsFer machine learning algorithms (e.g Streaming Algorithm implementations) to create a debug and traffic analytics for in and out data - a typical software analytics usecase with widespread necessity in anti-cybercrime, anti-theft and anti-virus products

4.274 Mainline USB drivers (USBWWAN, USB serial, USB storage) instrumented with debug statements in URB related functions

4.275 Apache Spark Python MapReduce Log Analyzer for kern.log - extracts patterns in USB requests and wireless LAN traffic. By enabling Kernel Address Sanitizer in linux kernel 4.10.3 64 bit code and Spark ETL, USB

related kernel function invocations have been profiled.

4.276 64-bit version of this driver is in separate repository based on 4.13.3 mainline linux kernel (32-bit version is based on mainline kernel 4.1.5)

4.277 USBmon and Ftrace kernel utility support for analyzing USB interface traffic and kernel functions invocation graph logs

4.278 This USB analyzer complements the Program Analysis - Software Analytics module in AstroInfer (for userspace applications) analyzing kernelspace function call graph and network activity among others

4.279 tcpdump and wireshark pcap format WLAN logs can also be analyzed by Spark Analyzer for regular expression patterns

4.280 Ftrace kernel function call graphs are written to a DOT file and analyzable by any Subgraph Miner

4.281 Ftrace call graphs have been analyzed by NetworkX for PageRank and Degree Centrality measures which provide insight into most active code in kernel for an executable

(Latest NeuronRain USBmd research version Code at sourceforge repositories: <https://sourceforge.net/p/usb-md/> , <https://sourceforge.net/p/usb-md64/code/ci/master/tree/> and periodically updated Design Details at:

https://sourceforge.net/p/usb-md64/code/ci/master/tree/USBmd_notes.txt,
<http://sourceforge.net/p/usb-md/code-0/HEAD/tree/>
[USBmd_notes.txt](#))

(Latest NeuronRain USBmd enterprise version Code at github repositories: <https://github.com/shrinivaasanka/usb-md-linux-github-code> , <https://github.com/shrinivaasanka/usb-md64-github-code> and periodically updated Design Details at:

[https://github.com/shrinivaasanka/virgo-linux-github-code/bl
ob/master/USBmd_notes.txt](https://github.com/shrinivaasanka/virgo-linux-github-code/blob/master/USBmd_notes.txt) ,
[https://github.com/shrinivaasanka/usb-md64-github-code/blo
b/master/USBmd_notes.txt](https://github.com/shrinivaasanka/usb-md64-github-code/blob/master/USBmd_notes.txt)).

5. NEURONRAIN - King Cobra - Kernelspace Messaging and Computational Economics Software that includes a new fictitious Electronic Message Currency - Neuro - which depends on Google Protocol Buffer based Cloud Object Move functionality in AsFer. Flow of such a fictional currency on AsFer EventNet implementation (actors in event vertices are buyers/sellers and direction of flow determines buy/sell) can simulate Economic Markets and Pricing. EventNet is thus equivalent to

hyperledgering in other cryptocurrencies and can capture money trail.

(NeuronRain KingCobra research version sourceforge

Repositories - <https://sourceforge.net/p/kcobra/> ,

<https://sourceforge.net/projects/kcobra64/> and periodically updated

design notes at

<https://sourceforge.net/p/kcobra64/code/ci/master/tree/KingCobraDesignNotes.txt>

, <https://sourceforge.net/p/kcobra/code-svn/KingCobraDesignNotes.txt>)

(NeuronRain KingCobra enterprise version sourceforge

Repositories - <https://github.com/shrinivaasanka/kingcobra-github-code> ,

<https://github.com/shrinivaasanka/kingcobra64-github-code> and

periodically updated design notes at

<https://github.com/shrinivaasanka/kingcobra64-github-code/blob/master/KingCobraDesignNotes.txt>,

<https://github.com/shrinivaasanka/kingcobra-github-code/blob/master/KingCobraDesignNotes.txt>)

Implements a minimal kernelspace and userspace messaging framework using VIRGO cpupooling (`virgo_clone`) and memory pooling drivers that in turn queues the requests into a kernel workqueue. There is also a standalone VIRGO queue kernel service that listens on the requests without dependencies on VIRGO cpupooling and memorypooling drivers to forward the requests. The VIRGO workqueue handler pops the request from workqueue and invokes KingCobra driver's `servicerequest` exported function and replies to the publisher and optionally disk persists the incoming requests to filesystem through VFS. Differentiator is KingCobra implements a cloud messaging with a decentralized queue with disk persistence and workflow in kernel level so that hardware is easily integrated into

cloud:

5.282 Linux Kernel workqueue based kernelspace pub-sub kernel module - Receives Messages from VIRGO Queuing

5.283 Atomic Cloud Move for Neuro (Message As Currency) electronic money is implemented In AsFer in C++ client-server perfect forwarding (overloaded std::move() and std::forward())

5.284 Userspace JMS pub-sub client-server

5.285 Pricing - CVXPY implementation - Eisenberg-Gale Market Equilibrium Convex Optimization for finding Market Clearing Prices for Goods and Services. Replacing Prices by Perceived Merit translates Market Equilibrium to Intrinsic Merit Versus Perceived Ranking Equilibrium.

5.286 Disk persistence of Request-Reply queues.

5.287 64-bit version of this driver is in separate

repository based on 4.13.3 mainline linux kernel (32-bit
version is based on mainline kernel 4.1.5)

5.288 Fictitious Neuro currency Buy-Sell transactions
are edges in an EventNet Event-Actor Graph implemented
in AsFer and can be mined for subgraph patterns -
EventNet is thus an economic network

5.334 Algorithmic Trading in Fictitious Neuro
Cryptocurrency - EventNet Graphical Event Model (GEM)
HyperLedger implementation

6. NEURONRAIN - acadpdrafts - All publications and
drafts with code (along with important documents and
Photo ID proofs) are uploaded at

<http://sourceforge.net/projects/acadpdrafts/>

The opensource codebases above are nonprofit academic research efforts. Premium Technical Support for above opensource codebases available. Premium Dual licensed Closedsource products derived from codebases in:

<https://github.com/shrinivaasanka>, <https://gitlab.com/shrinivaasanka> and
<https://sourceforge.net/u/userid-769929/profile/> are in development since 2010.

7. NeuronRain GRAFIT Open Learning :

https://github.com/shrinivaasanka/Grafit/tree/master/course_material, <https://gitlab.com/shrinivaasanka/Grafit>,

<https://sourceforge.net/u/userid-769929/Grafit/ci/master/tree/>

/ - May 2010 - Present - Free online courses based on

NeuronRain codebases and Design Documents,

Puzzles/Questions from Competitive examinations and

miscellaneous topics - non-linear material focusing on

Bigdata, Problem solving, Machine Learning,

Fundamentals, Programming - Creative Commons 4.0

NCND licensed - includes spillover implementations of

other NeuronRain repositories:

Virtual Classroom -

<https://classroom.github.com/classrooms/8086998-https->

github.com/shrinivaasanka-grafit

HAMSA - NeuronRain Lectures, Audio-Visuals related to NeuronRain repositories, Theory and Miscellaneous -

<https://kuja27.blogspot.com>

GRAFIT course materials (in .zip) are available from

Moodle GRAFIT website -

<https://moodle.org/pluginfile.php/4765687/user/private/Grafi>

<https://moodle.org/pluginfile.php/4765687/user/private/Grafit-master.zip?forcedownload=1>

BRIHASPATHI - Private Online Virtual Classrooms

Private repositories of virtual classrooms for various commercial online courses (BigData, Machine Learning, Topics in Mathematics and Computer Science,...) -

<https://github.com/Brihaspathi> - requires GitHub student

logins

April 2015 - September 2015 - CloudEnablers -

CusDelight - Corestack, Chennai - Architect

1. Design and PoC implementation in python for translating Topology Orchestration Specification for Cloud Applications (TOSCA) XML BPMN Plan Models to Stackforge/Mistral Workflow YAML (Mistral-translator - similar to Heat-translator project for Simple TOSCA YAML to Heat Orchestration Templates)
2. Design and PoC implementation in python for Jinja2

rendering template feature addition to TOSCA Plan XML and Mistral YAML based on JSON context for conditionals and control structures support

3. Creating a Capability matrix of different Cloud

Orchestration and template softwares:

AWS,Heat,Mistral,Puppet,Chef,Ansible,Saltstack,Twisted,Nunjucks

4. Study of OpenTOSCA implementation of TOSCA

(University of Stuttgart) and feasibility of its integration into Corestack (Cloudenablers)

5. Feasibility study of new features to Mistral Workflow

Engine Parser (if-else,nested for-each,with-items,concurrency)

6. Feasibility of Apache Brooklyn CAMP blueprints based cloud application deployment

7. Design and PoC Development of Service Dynamic Data

Discovery, MetaData and Actions REST API, Cloud Federation and Abstraction Layer (for abstracting providers, federated clouds and multiclouds like ComputeNext, Libcloud, Jclouds, Dasein, AWS etc.,) for provider abstraction and Directory Service REST API

8. Corestack code review

9. Study of feasibility of Jclouds integration into corestack

10. Design and PoC for Process Checkpoint and

Restoration with CRIU in baremetal, VMs and Docker

(with overlay and aufs union mount filesystems). Docker public repository -

<https://hub.docker.com/r/srinivasankannan/cloud-migration/>

11. Design and PoC sync scripts implementation for

Automated VM image synchronization with bup, rsync,

lsyncd-lua and inotifywait (inotify-tools).

12. Design and PoC implementation for Service Discovery

through above REST API and migrating them with `Jinja2`
`Heat` and `Mistral` `YAML` templates extraction and
Performance tunables for cloud.

June 2014 - July 2014 - Clockwork Interviews Pvt Ltd,
Chennai (Hiring product - <http://piqube.com>) -
Consultancy on Design and a Java based minimal
Implementation for computing the various statistical
measures from a job experience sequence: 1) Stability or
predictability of job experience sequence using Shannon
Entropy and Inversion number, 2) Estimating various
types of average experience metrics, 3) Mobility scoring,
4) Initial work on Skills based clustering of linkedin
profiles using String similarity and 5) Initial Hidden
Markov Model design for job change motivation inference

February 2014 - Clockwork Interviews Pvt Ltd, Chennai

(Hiring product - <http://piqube.com>) - In-office non-profit

Consultancy to PiQube for initial design of a Multivariate

Linear Regression model for Stability and Gap scoring in

Resume that includes entropy independent variables

using ENT sequence disorder measure, study of various

tools (including Stanford CoreNLP - Classifiers and

Recursive Tensor Neural Network Sentiment Analysis

Tools) for Sentiment Analysis of a Resume and Holistic

Resume scoring model using social network analysis from

multiple datasources

October 2013 - January 2014 - continued work on my opensource products (Krishna iResearch - https://sourceforge.net/users/ka_shrinivaasan) and non-profit machine learning remote consultancy to Clockwork Interviews Pvt Ltd, Chennai (Hiring product - <http://piqube.com>) for initial design of a Resume scoring, Named Entity Recognition in Resume using GATE, CRF+, IIT-Bombay CRF, study of Social Network Analysis Tools (SocVnet,SNAP etc.,) and informal mail interactions on my open source products among other things

September 2012 - February 2013 - Parttime Consultancy for Global Analytics(Chennai) and Work on my opensource products (Krishna iResearch)

1. Did the Python AMP workflow implementation as per the workflow specification for AMP using PyUtilib, RabbitMQ and Config files support. Wrote some python scripts for AMP workflow config files support and parsing.
2. Wrote Design spec analyzing workflow alternatives (like PyF, cascade) .
3. Wrote AMP workflow-over-cloud (AMP on Hadoop) specification in addition to AMP workflow spec.
4. Guided on GDP 2.7.x and 3.0 bug fixes and issues.
5. Helped in debugging GDP 3.0 timeout manager implementation in Feb 2013 which was found to be working.

Apart from the above worked on design and development

of my nonprofit open source products-

https://sourceforge.net/users/ka_shrinivaasan

January 2011 – March 2011 and July 2011 – August 2012

- Global Analytics(Chennai) - Senior Lead Software Architect (C/C++/Python)

1. Mentored and Managed a team of 3 people .

2. Released version 2.3.1 of Global Decision

Platform(GDP) with fixes for crashes in production environment.

3. Did refactoring, many bug fixes and enhancements to Global Decision Platform(GDP) version 2.5.0 for master demo lead passthroughs to work.

4. Implemented acceptor-worker thread model request handling in GDP 2.5.0 with single acceptor thread and configurable number of worker threads for Service Execution Manager(SEM) and Service components of GDP. This resulted in significant improvement in response time.

5. Designed and implemented session based request-response in GDP 2.5.0

6. Implemented modified timeout error logging with placeholders for errorcodes and request id and did fixes

for frequent crashes in production environment for GDP

2.5.0.

7. Released GDP version 2.5.1 for which 2 minor binaries were developed to clear the queue and test the creation of queues (for System V message queues)

8. Released GDP version 2.7 which has important bug fixes for crashes in python container(needed for AMP runtime described later), few features for graceful shutdown of the SEM, option to reconnect to MySQL during exception in Python container etc.,

9. Redesigned GDP 2.5.0 SEM loadbalancer for GDP 3.0 to route requests to service processes by periodically monitoring the service process load and removing a

bottleneck due to wait/notify code. This resulted in 10x speed up of the request-response throughput time.

10. For GDP version 3.0, designed and Implemented a new Session Timeout Manager algorithm to timeout the request session based on user configured timeout value(simplified version of Survival Index Based Transaction Timeout Manager mentioned later).

11. For GDP version 3.0, worked on embedding multiple python interpreters in Service process with Python C API replacing boost::python calls. This involves facility to have a global dictionary across multiple interpreters (or) local dictionary per request (or) per-thread dictionary for each service worker thread (using thread local storage) and also creation of pool of interpreters from which

interpreters are allocated per request and returned to the pool after the request is serviced. This obviated the need for boost::python since boost did not support multiple interpreters with multithreading and interpreter pooling. This involved fixing one of the most time consuming bug related to restricted mode python in 2.4.3. Since it was deprecated in python 2.7, fixed Python 2.4.3 source to circumvent restricted mode and also did another fix for the same problem with `PyImport_ImportModuleEx()`

12. Mentored the team to work on Dynamic Risk Tables (DRT) library of GDP to extend the database compatibility to MySQL and Oracle

13. Was involved in the Automated Modelling Platform(AMP) development to automate the predictive

modelling from the production data. Fixed many crashes in python runtime during testing of AMP prototype realtime design. Suggested an alternative design for AMP realtime based on external user address-space message queue, which populates the in-memory queue from on-disk request log, periodically replenishes it from disk, reads requests from message queue and posts them in parallel to Realtime-subset and Realtime-superset python services, and gives a virtual view of the single physical queue to each queue client(queue clients are located in SEM) - meant to replace an existing queueing module with msgsnd/msgrcv which involves user and kernel address-space copying to remedy huge queue backlog seen in AMP realtime prototype. This tightly couples GDP Python runtime with the AMP runtime

14. For GDP 3.0, implemented a GDP pluggable queue named GDAMPQ designed for AMP above – Global Decisioning and Automated Modeling Platform Queue which is a simple in-memory queue with optional disk persistence to replace an existing queueing module with msgsnd/msgrcv which involves user and kernel address-space copying. This in-memory queue in best testing circumstances is 30% better than the System V unix message queues.

15. Worked on Python 2.4.3 source code to analyze crashes in PyMalloc. Modified Python 2.4.3 source to add a flag DISABLEPYMALLOC to disable calls to PyMalloc and re-route them to unix malloc/free systemcalls. Also analyzed Python 2.4.3 source code for restricted mode related errors while designing multi-interpreters for GDP

3.0 and fixed them.

16. Worked on writing python configuration scripts for automating the build configuration and installation of GDP components like SEM, Service and Master .

17. Implemented a minor encrypt utility for master component of the GDP

18. Wrote Functional specification for GDP 3.0 Automated Installer design

19. Wrote Functional specification for AMP Workflow

20. Wrote Functional specification for AMP Cloud deployment

21. Studied python workflow packages for AMP workflow viz., `Pyutilib.workflow`, `PyF` and implemented a workflow prototype for AMP workflow in `Pyutilib.workflow`.

22. Implemented a small prototype stub using Dumbo Mapper-reducer for Cloud

Deployment of AMP

23. Did 2 one-week training sessions for GDP versions

24. Did one-week training session for Python.

25. Implemented a mini-GDP client and server for debugging socket related errors in production.

26. Developed a documentation knowledgebase intranet website for GDP documentation, design documents, installation troubleshooting documents and debugging documents.

27. Implemented a Build Version feature for GDP 2.7.1 and GDP 3.0 which dynamically generates a buildversion.h headerfile which contains #defines for build machine and architecture information,SVN source tree version information for the build and is built at runtime by SEM, Service and Master binaries which include this buildversion.h header file

(In August 2012 resigned for personal reasons.)

January 2011 - March 2011 - Consultant for Global Analytics(Chennai) and Work on my opensource products (Krishna iResearch) - (C++/Python)

Worked on my Open Source Products Design & Development (Krishna iResearch) and Did consulting and development for Global Analytics,Chennai (optimization, a smart-pointer reference-counted memory manager backed-up by an Object Pool, and refactoring) in Global Decision Platform (GDP) 2.3 for GPD 2.3.1 and 2.5 which is written in C++ with boost::python embedding.

August 2008 - June 2011 - Work on my opensource

products (Krishna iResearch)

Started working on open source project ASFER
(<http://asfer.sourceforge.net>) as part of my non-profit open
source initiative Krishna iResearch - ASFER is a rule
miner and executer- presently uses Vector space retrieval
and Support Vector Machines with added Support for
Naive Bayesian Multinomial Classifier and Decision Tree
Classifier

March 2006 - July 2008 - webMethods and webMethods

(now Software AG) (Bangalore) (C/C++/Java)

Worked on WebMethods Broker Server 5.x/6.x/7.x- a heavily multithreaded messaging product written in C/C++/Java based on publisher/subscriber model - Clients can publish and/or subscribe to certain predefined types of messages called “Events”. My responsibilities included development of new functionalities and fixing customer escalation issues in core areas of Broker (Connection Layer, Territories/Gateways, Broker Admin Tool etc.,) and writing knowledgebase documents.

November 2005 - February 2006 - Krishna iResearch (self-started, not-for-profit, open-source research initiative)

Initial design work for Krishna iResearch open source products focussing on algorithms for web and BigData

August 2005 - November 2005 - Verizon Data Services India, Chennai

In Verizon India for brief period and then self-study in Mathematics and Computer Science.

May 2003 - Krishna iResearch (self-started, not-for-profit, open-source research initiative)

Registered on SourceForge.net and Initial design work started for open source products focussing on algorithms for web and BigData

February 2000 - July 2005 - iPlanet (Sun Microsystems-Netscape Alliance) and Sun Microsystems (now Oracle) (Bangalore)

Ported SunONE Application Server to Red Hat Enterprise Linux

Advanced Server 2.1 (2005)

Certified JWSDP 1.3 webservices pack on Sun One Application Server 7.1 (2005)

Studied feasibility of supporting SunOne Web/Proxy Server on Solaris 10 zones (2005)

On Academic Sabbatical for 3 months (February 2005-May 2005)

Implementation of a threaded ICP server and porting
SOCKS server for Sun Java system web proxy server 4.0
(C/C++/Solaris/Windows/RHELinux) (2004-2005):

Implemented a threaded ICP(Internet Cache protocol)
server functionality using NSPR threads.

Also ported a legacy SOCKS server to proxy server 4.0.

Load Balancer Module for Apache Web Server 1.3.27 /
2.0.47

(C/C++) (2003-2004) :

Designed and implemented a Load balancer module for
Apache 1.3.27 which will route the

HTTP/HTTPS requests onto SunONE Application Server instances by Round Robin Algorithm.

This module is part of Sun ONE Application Server 7, Enterprise Edition. Also ported the module to Apache 2.0.47.

Optimizations and Features for iPlanet/Sun ONE Application

Server 6.x (Java) (open sourced at <http://glassfish.java.net>)
(2002) :

a) Fixed bugs related to performance, admin-tool and security in iPlanet Application Server 6.x. Also fixed few CTS bugs

in SunONE Application Server 7 Standard Edition.

b) Involved in optimization of SUN J2EE Reference Implementation (RI) to make the JTS transaction manager enterprise-class. This included introducing a new

Component Level

Transaction Attribute setting feature in iPlanet Application Server

6.5. With this, user can set the type of transaction (XA or Local)

at J2EE application component level. [Patents granted - details in the end]

c) Added a Trace functionality for the JTS transaction manager in SunONE Appserver 7.0

Enterprise Edition

(http://glassfish.sourceforge.com/documentation/1:2ur2-b04-1/dir_512e7fccb5ab465c594742cd72317a0e.html).

Debugging and Monitoring framework (MagicDraw) (2002)

:

Designed a Debugging and Monitoring framework for Sun

ONE

Application Server 7.0 Enterprise Edition.

Store Adapter prototype for High Availability(Java) (2002)

:

Implemented a store adapter prototype for Clustra High Availability Database. This was to test the feasibility of using

Clustra HADB as backend store for session data in SunONE Application Server 7.0 Enterprise Edition.

Minor Feature Addition to SunOne Application Server 6.5 SP1 (2002)

Added support for enabling/disabling TCP_NODELAY in SunOne Application Server 6.5 SP1

Survival Index based Transaction Timeout Manager for

iPlanet Application Server (Java)(2002):

Designed and implemented a new Transaction Timeout

Manager for iPlanet Application Server. This resulted in an

overall speed-up of 10% for the application server.

(Invention Disclosure done in 2002-2003)

Miscellaneous Bugfixes for customer escalations in the

BillerXpert product which was dependent on iPlanet

Application Server. This was one of the most involved

bugfixes that required many days of debugging with JVM

Profiling tools. (2001)

An automated Test suite for nightly build setup of

iPlanet Application Server 6.0 (2001):

Designed and implemented a test suite which ran a set of

regression tests on nightly builds of iPlanet Application Server - written on Korn shell.

JVM related bugfix for IBM's Java Virtual Machine team related to native thread and green thread implementations of JVM (2000)

Created a Resume Querying System using JDBC with Oracle Backend for Sun Microsystems Human Resources Department to track candidate profiles - named ResumeXpert (2000)

Official Training Courses (SunU)

1. Solaris internals
2. Multithreaded Application development in Solaris
3. SOLARIS DEVICE DRIVER INTERNALS

July 1999 - January 2000 - BaaN Infosystems (now SSA Global) (Hyderabad)

Developed a storefront for an e-commerce application called

E-Enterprise using Microsoft Internet Information Server/Site Server, Visual Basic, Active Server Pages and Visual Interdev

January 1999 - May 1999 - BE Thesis and Project(team)

- COBRA - Distributed Computing Framework based on CORBA(Visibroker) and JAVA.

April 1998 - June 1998 - BE Internship at Steel Authority of India Limited (Raurkela and Durgapur Steel Plants) - Enhancements to Automated Furnace Data Acquisition software through a GUI developed on VAX Fortran over VAX VMS

MISCELLANEOUS

1. Gold medal for proficiency from PSG Tech for ranking first in B.E.(CSE) - Received in February 2000
2. GRE (2002) - score 2040, TOEFL - 270
3. Joint Entrance Screening Test(JEST) 2006 in Theoretical Computer Science - Rank 21 (Reg No: 22123).
4. Participated in Winter School 2010 on Machine

Learning and Computer Vision, Microsoft Research & CIFAR, IISc

5. UGC Net July 2016 - 66 (I), 54 (II), 50.67 (III) (in percentage) (Roll No. 69004241)

EXTRA-CURRICULAR:

Painting, Writing, Reading non-fiction mostly on science, mathematics, religion and philosophy.

VIDEO RESUME(s):

<https://github.com/shrinivaasanka/asfer-github-code/tree/>

master/python-src/image_pattern_mining/ImageNet/testlogs/

(ExampleVideo*mp4)

<https://github.com/shrinivaasanka/Grafit/blob/master/>

course_material/NeuronRain/LinuxKernelAndCloud/code/

testlogs/Krishna_iResearch_NeuronRain.Repositories-2020-

07-10_13.17.20.mp4

K.Srinivasan

(also spelt as : *SrinivasanKannan, Ka.Shrinivaasan, ShrinivasKannan*)
(Research Website : <https://acadpdrafts.readthedocs.io>)

About Myself

Worked for various IT majors and startups from 1999 and did Doctoral research in theoretical computer science till 2011. Presently working on a non-funded and not-for-profit opensource initiative and pursuing independent unaffiliated academic research.

Academics

- B.A(Hindi)-Praveen Uttarardh-Dakshin Bharat Hindi Prachar Sabha-Chennai - 1988-1992
- B.E(Computer Science)-PSG College of Technology,Coimbatore- 1995-99 - Gold Medalist for Proficiency
- MSc(Computer Science)-Chennai Mathematical Institute(CMI),Chennai- 2008-10
- Junior Research Fellow (PhD-Computer Science)-CMI,Chennai-Incomplete- 2010-11

Work

- Associate Software Engineer - BaaN Infosystems (now SSA Global),Hyderabad - 1999-2000
- Member Tech Staff - iPlanet (Sun Microsystems-Netscape Alliance), Bangalore - 2000-2002
- Member Tech Staff - Sun Microsystems (now Oracle) - Bangalore - 2002-2005
- System Analyst - Verizon - Chennai - 2005
- Senior Software Engineer - webMethods - Bangalore - 2006-2007
- Engineering Specialist - webMethods (now Software AG) - Bangalore - 2007-2008
- Consultant and Architect - Global Analytics (now GAIN credit) - Chennai - 2011-2013
- Consultant - PiQube Analytics (Clockwork Interviews) - Chennai - 2013-2014
- Architect - Cusdelight-CloudEnablers - Chennai - 2015

Research Publications - CMI/IMSc/IIT,Chennai

- Decidability of Complementation - 2011 - <http://arxiv.org/abs/1106.4102>
- Algorithms for Intrinsic Merit - 2010 - <http://arxiv.org/abs/1006.4458>
- NIST TAC 2010 version of Algorithms for Intrinsic Merit -
http://www.nist.gov/tac/publications/2010/participant.papers/CMI_IIT.proceedings.pdf

Research Profiles

- Google Scholar - <https://scholar.google.co.in/citations?user=eLZY7CIAAAJ&hl=en>
- DBLP - <http://dblp.dagstuhl.de/pers/hd/s/Shrinivaasan:Ka>
- arXiv - ORCID - <https://orcid.org/0000-0003-1822-4697>
- Microsoft Academic - [https://academic.microsoft.com/search?q=ka%20shrinivaasan&qe=%40%40%40Composite\(AA.AuN%3D%3D%27ka%20shrinivaasan%27\)&f=&orderBy=4&skip=0&take=10](https://academic.microsoft.com/search?q=ka%20shrinivaasan&qe=%40%40%40Composite(AA.AuN%3D%3D%27ka%20shrinivaasan%27)&f=&orderBy=4&skip=0&take=10)
- Researchgate - https://www.researchgate.net/profile/Srinivasan_Kannan5
- Semantic Scholar - <https://www.semanticscholar.org/author/Ka.-Shrinivaasan/1861803>
- CiteSeerX - <https://citeseerk.ist.psu.edu/search?q=Ka.+Shrinivaasan>
- NASA/ADS - https://ui.adsabs.harvard.edu/search/q=author%3A%22Shrinivaasan%2C%20Ka.%22&sort=date%20desc%2C%20bibcode%20desc&p_=0

Alumni Profiles

- CMI Alumni - 2008-10 - <https://www.cmi.ac.in/people/alumni-profile.php?id=shriniwas>
- CMI JRF - 2010-11 - [http://www.cmi.ac.in/people/fac-profile.php?id=shriniwas](https://www.cmi.ac.in/people/fac-profile.php?id=shriniwas)
- PSG Tech - 1995-99 - <http://alumni.psgtech.ac.in/profile/view/srinivasan-kannan-1>

Publication Drafts - Unguided and Unreviewed - 2012-present

Independent academic research publication drafts expanded on previous publications - <https://acadpdrafts.readthedocs.io>

Open Source Initiative - Krishna iResearch - 2003-present

Presently working individually on research and development of non-commercial, non-funded open source copyleft dual-licensed initiative (no team or sponsor involved) - cloud, bigdata analytics and machine learning augmented new Linux Kernel fork-off :

NeuronRain Research - http://sourceforge.net/users/ka_shrinivaasan

NeuronRain Green - <https://github.com/shrinivaasanka>/Krishna_iResearch GitHub Organization - <https://github.com/Krishna-iResearch>
NeuronRain Antariksh - <https://gitlab.com/shrinivaasanka>/NeuronRain Documentation, FAQ and Licensing - <http://neuronrain-documentation.readthedocs.io/en/latest/>
Previous repositories include an open learning free courseware (https://github.com/shrinivaasanka/Grafit/tree/master/course_material) replicated in SourceForge, GitLab and implementations of publications and drafts in <https://acadpdrafts.readthedocs.io>.

Brihaspathi - Private Online Virtual Classrooms and JAIMINI Closed Source Private Repositories

GitHub - Private repositories of virtual classrooms for various commercial online courses (BigData, Machine Learning, Topics in Mathematics and Computer Science,...) and JAIMINI Closed Source Derivative of NeuronRain - <https://github.com/Brihaspathi> - requires GitHub student logins

SourceForge - <https://sourceforge.net/projects/jaimini/>

GitLab - <https://gitlab.com/shrinivaasanka/jaimini>

Detailed CV

Details on work and academics - https://github.com/shrinivaasanka/Krishna_iResearch_DoxyxygenDocs/blob/master/kuja27_website_mirrored/site/kuja27/CV_of_SrinivasanKannan_alias_KaShrinivaasan_alias_SrinivasKannan.pdf, https://github.com/shrinivaasanka/Krishna_iResearch_DoxyxygenDocs/blob/master/kuja27_website_mirrored/site/kuja27/BITSPilaniAV.pdf

Contact Address

172, Gandhi Adigal Salai,
Kumbakonam-612001
Ph: 9789346927
ka.shrinivaasan@gmail.com, shrinivas.kannan@gmail.com, kashrinivaasan@live.com

Domain of Work - Development and Architecture

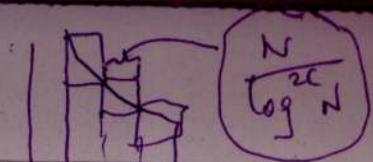
Middleware(Web, Application, Messaging etc.,), Machine Learning, Bigdata Analytics, Linux Kernel, Cloud (Linux Kernelspace RPC, Hadoop, Spark, CloudOSes), C/C++/Java/Python.

Research-Theory and Engineering

Computational Number Theory Algorithms, Computational Geometry, Computational Linguistics and Natural Language Processing, Computational Economics, Algorithms for Massive Datasets and Machine Learning, Fame and Intrinsic Fitness/Merit, Computational Complexity of Majority Voting, Satisfiability and related, Pseudorandomness, Program Analysis.

24/6/2013
29/6/2013

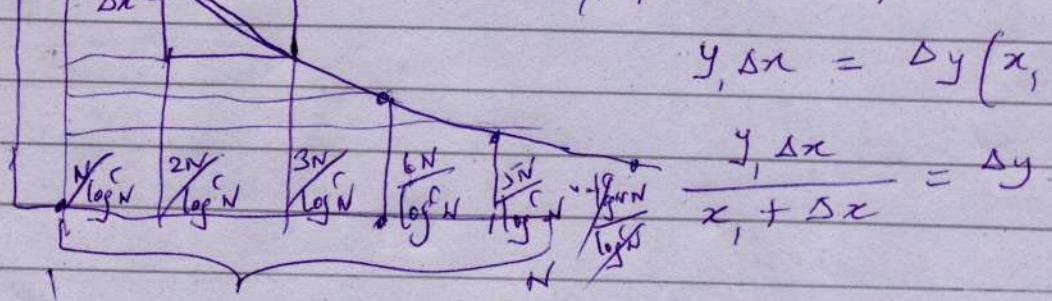
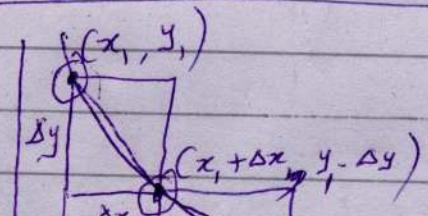
$$T_{\text{to log}} \frac{N}{\log^c N}$$



$$(\log^c N) \log \log \left(\frac{N}{\log^2c N} \right) = O \left(\log^2(N) \right).$$

$$\left(\log^c \left(\frac{N}{\log^c N} \right) \right) \times \frac{N}{\log^2c N}$$

30/6/2013



$$y_1 = \frac{N}{x_1} \Rightarrow \Delta y = \frac{N \Delta x}{x_1 (x_1 + \Delta x)} \quad \Delta x = \frac{N}{\log^c N}$$

$$\sum \Delta y_i = \Delta y = \frac{N \times N}{\log^c N} = \frac{N^2}{\log^c N} \quad \log^c N$$

$$\frac{N^2}{x_1 (x_1 \log^c N + N)}$$

$$\Delta y_1 = \frac{N^2}{x_1 (x_1 \log^c N + N)}$$

$$\sum_{N / \log^c N} \Delta y_i = N^2 \left[\frac{1}{\frac{N}{\log^c N} \left(\frac{N}{\log^c N} \log^c N + N \right)} + \frac{1}{\frac{2N}{\log^c N} \left(\frac{2N}{\log^c N} \log^c N + N \right)} + \dots \right]$$

$$\begin{aligned}
 \sum_{\substack{i=1 \\ \log^c N}}^N \Delta y_i &= \frac{N^2 \log^c N}{N} \left[\frac{1}{(N+N)} + \frac{1}{2(2N+N)} + \frac{1}{3(3N+N)} + \dots + \frac{1}{\log^c N (\log^c N N + N)} \right] \\
 &= \cancel{\frac{N^2 \log^c N}{N \times R!}} \left[\frac{1}{1(1+1)} + \frac{1}{2(2+1)} + \frac{1}{3(3+1)} + \dots + \frac{1}{\log^c N (\log^c N + 1)} \right] \\
 &\leq \log^c N \left[\frac{1}{1^2} + \frac{1}{2^2} + \frac{1}{3^2} + \dots + \frac{1}{\log^c N} \right] \\
 &\leq \left[\log^c N \left(\frac{1}{1} + \frac{1}{2} + \dots + \frac{1}{b} \right) \right] \text{Riemann Zeta function } (s=2)
 \end{aligned}$$

Each row slice is $\log \log N$

Total time for $\frac{N}{\log^c N}$ to N is $O(\log^c N \log \log N)$

$$= O(\log^{c+1} N)$$

Recursion stops when

$$N = \log^{2^c} N$$

$$\log N = q_c \log \log N$$

$$q = \frac{\log N}{c \log \log N}$$

1 to $\frac{N}{\log^c N}$ is searched recursively with above algorithm

$$O(\log^{c+1} N) + O\left(\log^{c+1} \frac{N}{\log^c N}\right) + O\left(\log^{c+1} \frac{N}{\log^{2^c} N}\right)$$

$$\begin{aligned}
 & \left((q+1) \log^{\frac{1}{c+1}} N \right)^{c+1} \\
 & \leq \left(\frac{\log N}{c \log \log N} + 1 \right)^{c+1} \log^{c+1} N \\
 & \leq (\log N + 1)^{c+1} \log^{c+1} N \\
 & = \Theta \left(\log^{\frac{c+1}{2}} N \log^{c+1} N \right) = \Theta \left(\log^{2c+2} N \right)
 \end{aligned}$$

Should we bound theta of

Big O

(or) it can be

$$\Theta \left(\frac{\log^{2c+2} N}{(\log \log N)^{c+1}} \right)$$

Ka. Srinivasan
30/6/2013

Discrete Hyperbolic Polylogarithmic Sieve For Integer Factorization (Draft)

Ka.Shrinivaasan (ka.shrinivaasan@gmail.com)

June 16, 2013

Abstract

Best known algorithm for factorization is the General number field sieve algorithm which takes time exponential in number bits in the input or simply called exponential algorithm. In this article, a polylogarithmic time (in size of input) sieve for factorization is discussed that uses elementary principles.

1 Discretization of a hyperbolic arc

1. A plot of a hyperbolic function is ($xy = N$) drawn on a 2-dimensional plane. This plot of hyperbola is continuous. This hyperbola can be discretized (or tesselated) with set of rectangles that cover the hyperbolic arc.
2. Discretization step divides the x-axis of the hyperbola into $\log^c N$ intervals where each interval is of length $N/\log^c N$. Infact it suffices to calculate upto \sqrt{N} length along the x-axis since atleast 1 factor must lie within \sqrt{N} . Correspondingly the y-axis is also split into intervals whose width is $y * \sqrt{N}/(x \log^c N + \sqrt{N})$ where (x, y) are coordinates of bottom left corners of each rectangle.
3. The constant c is chosen to be quite large so as to shrink the size of each rectangle as much as possible. Thus each rectangle is of length $\sqrt{N}/\log^c N$ and width $y * \sqrt{N}/(x \log^c N + \sqrt{N})$ where (x, y) are coordinates of topleft corners of each rectangle. The factors (say p,q) of N are sure to be present within one of the rectangles if N is composite. Algorithm then is to sieve out rectangles to get to the factors. Looping through $\log^c N$ rectangles takes time $O(\log^c N * \text{TimePerRectangle})$.
4. Within each rectangle, the values of coordinates are hypothetically considered to be products of x and y coordinates. This is just supposed to exist and no lookup table or memoization is needed (though it can be made into a dynamic programming recurrence algorithm if knowledge of factors of numbers lower than N is assumed). To find the coordinates where N is located within the rectangle, binary search is applied on the coordinate products with some specialized modifications as described in next section below.
5. The line equation corresponding to leading diagonal within each rectangle obtained previously, intersects with the hyperbola, thereby creating a bow-shaped area (hemmed between hyperbolic arc and diagonal) of coordinate products within each rectangle. This bow shaped area is only a small percentage of the area of each rectangle.

6. The row slices of coordinate products within this bow shaped area are in sorted order from left to right. Also the coordinate products in diagonal of the rectangle from bottom-left to top-right of each rectangle are in sorted ascending order.

2 Binary search on the above bow-shaped area within each rectangle

1. Since coordinate products along bottom-left to top-right diagonal are in ascending order and border coordinates for each rectangle are already known, doing a binary search along the diagonal helps in reaching to the closest point within the rectangle to the factor coordinates (p, q) such that $pq = N$. Closest point (x, y) is where $xy - pq$ is least.
2. Once the closest point (x, y) is reached on the diagonal, from that row x , remaining row slices within bow-shaped area can be sequentially scanned and each row slice within the bow-shaped area can be binary searched for factor coordinates. This is repeated for each rectangle. Time per rectangle is $O(\log^2 N)$ since each side of the rectangle is a function of $N/\log^c N$ which is polylog in input size. As mentioned in a later section below for suitably chosen c , $N/\log^c N$ is upperbounded by $\log(N)$ and thus area of each rectangle is $O(\log^2 N)$.

```

3. Functions BinarySearchSubroutine(xleft, yleft, xright, yright)
{
  if(yleft == yright) //horizontal scan
  {
    if (xleft+xright)/2 * yleft == N
    {
      return (xleft+xright)/2, yleft as factors of N
    }
    else if (xleft + xright)/2 * yleft > N
      BinarySearchSubroutine(xleft, yleft, (xleft+xright)/2, yright)
    else
      BinarySearchSubroutine((xleft+xright)/2, yleft, xright, yright)
    }
    else //diagonal scan
    {
      same as above except that midpoint is computed as
      ([xleft+xright]/2 , [yleft+yright]/2)
      and proceed as above outer if clause
    }
  }
}

```

3 Discrete Hyperbolic Polylogarithmic Sieve for Integer Factorization - Algorithm Steps

1. Before the factorization loop starts for a number N to be factorized, the constant c has to be chosen in such a way that $N/\log^c N$ is less than $\log N$.
2. It can be derived that if c is greater than $\log N/\log\log N - 1$ then it makes sure that $N/\log^c N$ is less than $\log N$.

3. Thus a choice of value of c is made adhering to above inequality to make the sides of rectangles sublogarithmic in N .
4. Split interval $[0 - N]$ into $\log^c N$ intervals of width $N/\log^c N$ each.
5. For each of $\log^c N$ rectangle of width $N/\log^c N$ and height $y * \sqrt{N}/(x \log^c N + \sqrt{N})$ where (x, y) is the coordinate of bottom-left corner of each rectangle, compute the following loop:
 - (a) By Binary Search, using BinarySearchSubroutine described above, along the bottom-left-topright diagonal, reach to the closest point (x, y) to the factor coordinates (p, q) such that $pq = N$ and $xy - pq$ is least within the rectangle.
 - (b) From point (x, y) and within the bow shaped area confined within the hyperbolic area and the diagonal of the rectangle, do horizontal binary search for coordinate of $N(p, q)$ such that $pq = N$.
 - (c) Output $p, q(pq = N)$ if found in previous step and break; else continue loop

Thus total time taken is $O(\log^c N * \log N * \log N) = O(\log^{c+2} N)$ for choice of c described below.

4 $\log(N)$ Upperbounds $N/\log^c N$ for suitable choice of c

Key to this algorithm is that side of each rectangle $N/\log^c N$ is upperbounded by $\log N$ for suitable choice of c . The constant c has to be chosen in such a way that $N/\log^c N$ is less than $\log N$. It can be derived that if c is greater than $\log N / \log(\log(N)) - 1$ then it makes sure that $N/\log^c N$ is less than $\log N$. This is just a lowerbound on what values c can have and c is not a function of N as it might seem. For example if N is 10^{200} then c has to be greater than $200 \log(10) / \log(200 \log(10)) - 1$ which is approximately 85. Choosing c as 100 satisfies this lowerbound for c to be greater than 85. Since sides of rectangle are upperbounded by $\log(N)$, areas of all rectangles are upperbounded by $\log^2 N$. Since choice of c is predecided before the factorization loop starts, c is invariant in the loop and it depends only on number to be factorized. Hence even if the complete rectangle is scanned instead of bow-shaped area, time per rectangle is still $O(\log^2 N)$. For arbitrarily large values of c , the upperbound widens between $\log N$ and $N/\log^c N$. Though by fixing c as $\log(N)/\log(\log(N))$ would be sufficient for $\log N$ to upperbound $N/\log^{\log(N)/\log(\log(N))} N$ for all N , leaving it as a lowerbound minimum value for c gives some flexibility in deciding the number of rectangles that tessellate the hyperbola. If we have $\log^c N$ number of processors which would imply an NC circuit, above factorization loop can in fact be done using a logdepth, bounded width NC1 circuit with each rectangle assigned to one processor if binary search per rectangle is also in logdepth and constant c decides the extent of parallelism.

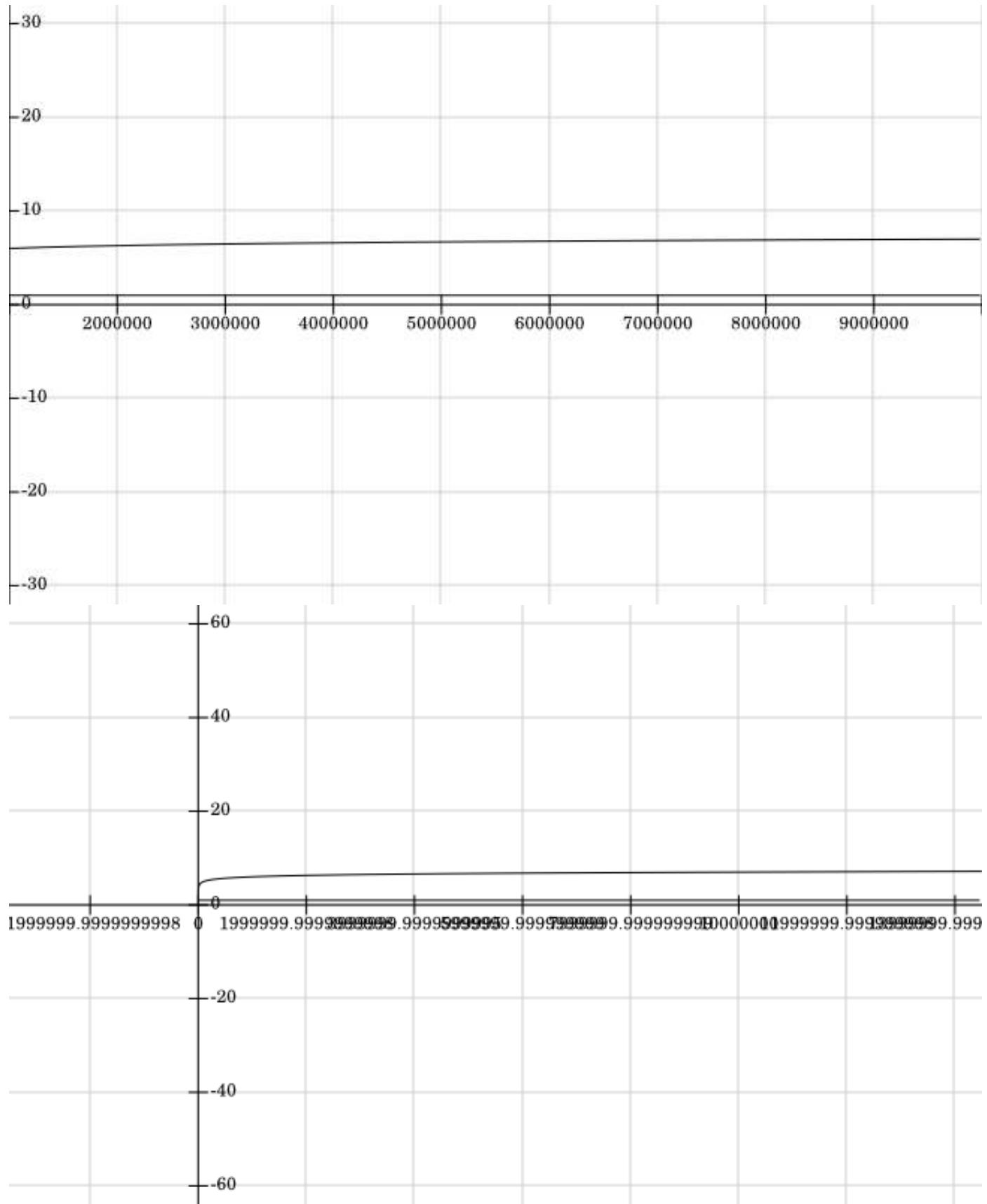
5 Implications

As a consequence of this polylog algorithm for factorization, RSA cryptosystem's large prime key-pair can be found in polynomial time by factorization for which till now there are only BQP quantum polynomial algorithms available. Hence factorization in P implies probably that $BQP = P$ (if factorization is BQP-complete) since there is only a quantum polynomial algorithm for factorization and implies therefore that $BPP = P$ since BQP contains BPP (implying pseudorandomness and determinism are equivalent in computational power). This algorithm outputs all factors of an integer within polylogarithmic time in input bits and as a special case this is also a polynomial time primality test similar to PRIMES in P (AKS).

6 Acknowledgement

I dedicate this article to God.

7 Appendix 1 - function plots for $\log N$ upperbounding $N/\log^{\log(N)/\log(\log(N))} N$



8 Appendix 2 - Indefinite Elliptic Integral for Hyperbolic Arc Length which is attained when number of rectangles tend to infinity

$$\int \sqrt{1 + \frac{N^2}{x^4}} dx = -x \sqrt{\frac{N^2}{x^4} + 1} +$$
$$\frac{2N x^2 \sqrt{\frac{N^2}{x^4} + 1} \sqrt{1 - \frac{i x^2}{N}} \sqrt{1 + \frac{i x^2}{N}} \left(E\left(i \sinh^{-1} \left(\sqrt{\frac{i}{N}} x \right) \middle| -1 \right) - F\left(i \sinh^{-1} \left(\sqrt{\frac{i}{N}} x \right) \middle| -1 \right) \right)}{\sqrt{\frac{i}{N}} (N^2 + x^4)}$$

+ constant

Computed by Wolfram|Alpha

9 Bibliography

References

- [1] Elementary algebraic principles
- [2] Binary Search
- [3] Various Function plotter software
- [4] Python function plot (text) code written by self
- [5] Scanned rough notes containing illustrations - <https://docs.google.com/file/d/0B8TCub8qrCY8czZJZFJMT1V>
- [6] Scanned rough notes containing illustrations(annexures):
 - 1. <https://docs.google.com/file/d/0B8TCub8qrCY8Ykt4cUFDZmpDX1U/edit>
 - 2. <https://docs.google.com/file/d/0B8TCub8qrCY8U1pEeUxrUENfcFU/edit>
 - 3. <https://docs.google.com/file/d/0B8TCub8qrCY8SkpwcjBDYkMtSXM/edit>

Discrete Hyperbolic Polylogarithmic Sieve For Integer Factorization - using Interpolation Search (Draft)

Ka.Shrinivaasan (ka.shrinivaasan@gmail.com)

June 30, 2013

Abstract

Best known algorithm for factorization is the General number field sieve algorithm which takes time exponential in number bits in the input or simply called exponential algorithm. In this article, a polylogarithmic time (in size of input) sieve for factorization is discussed that uses elementary principles.

1 Discretization of a hyperbolic arc

1. A plot of a hyperbolic function is ($xy = N$) drawn on a 2-dimensional plane. This plot of hyperbola is continuous. This hyperbola can be discretized (or tesselated) with set of rectangles that cover the hyperbolic arc.
2. Discretization step divides the x-axis of the hyperbola into $\log^c N$ intervals where each interval is of length $N/\log^c N$. Infact it suffices to calculate upto \sqrt{N} length along the x-axis since atleast 1 factor must lie within \sqrt{N} . Correspondingly the y-axis is also split into intervals whose width is $y * \sqrt{N}/(x \log^c N + \sqrt{N})$ where (x, y) are coordinates of topleft corners of each rectangle.
3. The constant c is chosen to be quite large so as to shrink the size of each rectangle as much as possible. Thus each rectangle is of length $\sqrt{N}/\log^c N$ and width $y * \sqrt{N}/(x \log^c N + \sqrt{N})$ where (x, y) are coordinates of topleft corners of each rectangle. The factors (say p,q) of N are sure to be present within one of the rectangles if N is composite. Algorithm then is to sieve out rectangles to get to the factors. Looping through $\log^c N$ rectangles takes time $O(\log^c N * \text{TimePerRectangle})$.
4. Within each rectangle, the values of coordinates are hypothetically considered to be products of x and y coordinates. This is just supposed to exist and no lookup table or memoization is needed (though it can be made into a dynamic programming recurrence algorithm if knowledge of factors of numbers lower than N is assumed). To find the coordinates where N is located within the rectangle, Interpolation search, a variant of binary search that takes $\log \log N$ time on the average, is applied on the coordinate products with some specialized modifications as described in next section below.
5. The line equation corresponding to leading diagonal within each rectangle obtained previously, intersects with the hyperbola, thereby creating a bow-shaped area (hemmed between hyperbolic arc and diagonal) of coordinate products within each rectangle. This bow shaped area is only a small percentage of the area of each rectangle.

6. The row slices of coordinate products within this bow shaped area are in sorted order from left to right. Also the coordinate products in diagonal of the rectangle from bottom-left to top-right of each rectangle are in sorted ascending order. Thus searching these row slices of coordinate products (with some constant extension to each slice on leftside) within each rectangle is sufficient to sieve out the factors.

2 Interpolation search on the above bow-shaped area within each rectangle

1. The hyperbolic bow arc area obtained above can be searched with interpolation search on each of the row slice within a rectangle (along the y-axis).
2. Functions `InterpolationSearchSubroutine(xleft, yleft, xright, yright)`

```

{
  if(yleft == yright) //horizontal scan on a row slice yleft==yright
  {
    if (xleft+xright)/2 * yleft == N
    {
      return (xleft+xright)/2, yleft as factors of N
    }
    else if (xleft + xright)/2 * yleft > N
      InterpolationSearchSubroutine(xleft, yleft, (xleft+xright)/2, yright)
    else
      InterpolationSearchSubroutine((xleft+xright)/2, yleft, xright, yright)
    }
  }
}
```

3 Discrete Hyperbolic Polylogarithmic Sieve for Integer Factorization - Algorithm Steps

1. Before the factorization loop starts for a number N to be factorized, the constant c has to be chosen suitably. This constant decides the parallelism as described later.
2. Split interval $[0 - N]$ into $\log^c N$ intervals of width $N/\log^c N$ each.
3. For each of $\log^c N$ rectangle of width $N/\log^c N$ and height $y * \sqrt{N} / (x \log^c N + \sqrt{N})$ where (x, y) is the coordinate of top-left corner of each rectangle, compute the following loop:
 - (a) By Interpolation Search, using `InterpolationSearchSubroutine` described above, search each row slice within the bow shaped area confined within the hyperbolic area and the diagonal of the rectangle.
 - (b) Output $p, q (pq = N)$ if found in previous step and break; else continue loop

Low level calculations are derived later and also described in attached handwritten notes in appendix. This algorithm takes average upperbound time of $O(\log^c N * \log \log N)$.

4 Implications

As a consequence of this polylog algorithm for factorization, RSA cryptosystem's large prime key-pair can be found in polynomial time by factorization for which till now there are only BQP quantum polynomial algorithms available. Hence factorization in P implies probably that $BQP = P$ (if factorization is BQP-complete) since there is only a quantum polynomial algorithm for factorization and implies therefore that $BPP = P$ since BQP contains BPP (implying pseudorandomness and determinism are equivalent in computational power). This algorithm outputs all factors of an integer within polylogarithmic time in input bits and as a special case this is also a polynomial time primality test similar to PRIMES in P (AKS).

5 Derivation of the upperbound time for discrete hyperbolic factorization

Theorem 1. *Discrete Hyperbolic Polylogarithmic Factorization has an average upperbound running time of $O(\log^{2c+2} N)$*

Proof. Area of hyperbolic bow area per rectangle = Area of rectangle - Area of hyperbolic arc (obtained by area definite integral)

$$= (1/2 * N/\log^c N * y_1 \sqrt{N}) / (2 * \log^c N * (x_1 \log^c + \sqrt{N})) - N \log(x_1 + N/\log^c N) + N \log x_1$$

Dividing above bow area by y-axis side of the rectangle deltay gives average length of each hyperbolic row slice within bow: $((1/2 * N/\log^c N * y_1 \sqrt{N}) / (2 * \log^c N * (x_1 \log^c + \sqrt{N})) - N \log(x_1 + N/\log^c N) + N \log x_1) / (y_1 * \sqrt{N}) / (x_1 * (\log^c N) + \sqrt{N}))$

The above is average length of each hyperbolic row slice within bow area that needs to be interpolation searched. Time taken for interpolation search of row slice that is in ascending order from left to right will be $\log\log(\text{aboveexpression})$:

$$\log\log((1/2 * N/\log^c N * y_1 \sqrt{N}) / (2 * \log^c N * (x_1 \log^c + \sqrt{N})) - N \log(x_1 + N/\log^c N) + N \log x_1) / (y_1 * \sqrt{N}) / (x_1 * (\log^c N) + \sqrt{N}))$$

Above term can be upperbounded as follows ignoring some subtraction and division terms:

$$\text{Aboveexpression} \leq \log\log((1/2 * N/\log^c N * y_1 \sqrt{N}) / (2 * \log^c N * (x_1 \log^c + \sqrt{N})) + N \log x_1) / (y_1 * \sqrt{N}) / (x_1 * (\log^c N) + \sqrt{N}))$$

$$\text{Aboveexpression} \leq \log\log((N * y_1 \sqrt{N}) + N \log x_1) * 2 * \log^c N * (x_1 \log^c N + \sqrt{N}) / (2 * \log^c N * y_1 * \sqrt{N}))$$

$$\text{Aboveexpression} \leq \log\log((N/2 \log^c N) + (N \log x_1 (x_1 \log^c N + \sqrt{N}))) / (y_1 * \sqrt{N})) \text{ Since } y_1 = N/x_1, \text{ Aboveexpression} \leq \log\log((N/2 \log^c N) + (x_1 * \log x_1 (x_1 \log^c N + \sqrt{N}))) / (\sqrt{N}))$$

x_1 can take \sqrt{N} as maximum (if only \sqrt{N} fraction of x-axis is scanned with rectangular sieves):

$$\text{Aboveexpression} \leq \log\log((N/2 \log^c N) + (\log \sqrt{N} (\sqrt{N} \log^c N + \sqrt{N})))$$

$$\text{Aboveexpression} \leq \log\log(N + (\sqrt{N}/2)(\log^{c+1} N + \log N))$$

$$\text{Aboveexpression} \leq \log\log(2 * N * \log^{c+1} N)$$

$$\text{Aboveexpression} \leq \log\log(2 * N * \log^{c+1} N) \text{ or } \log\log(2 * N * (c+1) * \log N)$$

Thus time per row slice interpolation search per rectangle is upperbounded by:

$$\log\log(q * N * \log N) \text{ for } q = 2c + 2$$

Multiplying above upperbound for deltay_i rows within each rectangle_i and for all $\log^c N$ rectangles, gives the total running time:

$$\sum_{N/\log^c N}^N N \sqrt{N} / (x_1 (x_1 \log^c N + \sqrt{N})) \text{ with } \text{deltax} = N/\log^c N$$

Ignoring $x_i = 1$ consecutive $x_i(s)$ are: (each rectangle is $N/\log^c N$ wide)

$$N/\log^c N, 2N/\log^c N, 3N/\log^c N, \dots, (\log^c N)N/\log^c N$$

Above summation can be reduced as:

$$\begin{aligned}
& N\sqrt{N} * [(1/(N/\log^c N(N/\log^c N * \log^c N + \sqrt{N}))) + \dots] \\
& N\sqrt{N} * \log^c N / N * (1/(N + \sqrt{N}) + 1/(2(2N + \sqrt{N})) + \dots + 1/(\log^c N(\log^c N * N + \sqrt{N}))) \\
& \text{aboveexpression} \leq \log^c N(1/\sqrt{N} + 1/(2^2\sqrt{N}) + 1/(3^2\sqrt{N}) + \dots + 1/(\log^{2c} N\sqrt{N})) \\
& \text{aboveexpression} \leq \log^c N(1 + 1/(2^2) + 1/(3^2) + \dots + 1/(\log^{2c} N)) \\
& \text{aboveexpression} \leq (\log^c N/\sqrt{N}) * \text{RiemannZetaFunction}(s = 2) \\
& \text{aboveexpression} \leq (\log^c N/\sqrt{N}) * (\pi^2/6)
\end{aligned}$$

Thus total time for interpolation search of all row slices within all rectangles is : $O((\log^c N/\sqrt{N}) * 3\log\log(q * N * \log N)$ or simply $O((\log^c N * \log\log(N))$

In the calculations above the part of x-axis only upto \sqrt{N} is divided into $\log^c N$ intervals of width. The calculations for dividing x-axis upto N are below:

$$1. \text{deltax} = N/\log^c N \text{ and } \text{deltay} = N^2/(x_i(x_i\log^c N + N)) \text{ where } x_i(s) \text{ are } N/\log^c N, 2N/\log^c N, 3N/\log^c N, \dots, (\log^c N)$$

Area of hyperbolic bow area per rectangle = Area of rectangle - Area of hyperbolic arc(obtained by area definite integral)

$$= (1/2 * N/\log^c N * y_1 * N) / (2 * \log^c N * (x_1\log^c N + N)) - N\log(x_1 + N/\log^c N) + N\log x_1$$

Dividing above bow area by y-axis side of the rectangle deltay gives average length of each hyperbolic row slice within bow: $((1/2 * N/\log^c N * y_1 * N) / (2 * \log^c N * (x_1\log^c N + N)) - N\log(x_1 + N/\log^c N) + N\log x_1) / (y_1 * N / (x_1 * (\log^c N) + N))$

The above is average length of each hyperbolic row slice within bow area that needs to be interpolation searched. Time taken for interpolation search of row slice that is in ascending order from left to right will be $\log\log(\text{aboveexpression})$:

$$\log\log(((1/2 * N/\log^c N * y_1 * N) / (2 * \log^c N * (x_1\log^c N + N)) - N\log(x_1 + N/\log^c N) + N\log x_1) / (y_1 * N / (x_1 * (\log^c N) + N)))$$

Above term can be upperbounded as follows ignoring some subtraction and division terms:

$$\text{Aboveexpression} \leq \log\log(((1/2 * N/\log^c N * y_1 * N) / (2 * \log^c N * (x_1\log^c N + N)) + N\log x_1) / (y_1 * N / (x_1 * (\log^c N) + N)))$$

$$\text{Aboveexpression} \leq \log\log((N * y_1 * N) + N\log x_1) * 2 * \log^c N * (x_1\log^c N + N) / (2 * \log^c N * y_1 * N))$$

$$\text{Aboveexpression} \leq \log\log((N/2\log^c N) + (N\log x_1(x_1\log^c N + N)) / (y_1 * N)) \text{ Since } y_1 = N/x_1, \text{Aboveexpression} \leq \log\log((N/2\log^c N) + (x_1 * \log x_1(x_1\log^c N + N)) / (N))$$

x_1 can take \sqrt{N} as maximum (if only \sqrt{N} fraction of x-axis is scanned with rectangular sieves):

$$\text{Aboveexpression} \leq \log\log((N/2\log^c N) + (\log N(N\log^c N + N)))$$

$$\text{Aboveexpression} \leq \log\log(N + (N/2)(\log^{c+1} N + \log(N)))$$

$$\text{Aboveexpression} \leq \log\log(2 * N * \log^{c+1} N)$$

$$\text{Aboveexpression} \leq \log\log(2 * N * \log^{c+1} N) \text{ or } \log\log(2 * N(c + 1) * \log N)$$

Thus time per row slice interpolation search per rectangle is upperbounded by:

$$\log\log(q * N * \log N) \text{ for } q = 2c + 2$$

Multiplying above upperbound for deltay_i rows within each rectangle_i and for all $\log^c N$ rectangles, gives the total running time:

$$\sum_{N/\log^c N}^N N * N / (x_1(x_1\log^c N + N)) \text{ with } \text{deltax} = N/\log^c N$$

Ignoring $x_i = 1$ consecutive $x_i(s)$ are: (each rectangle is $N/\log^c N$ wide)

$$N/\log^c N, 2N/\log^c N, 3N/\log^c N, \dots, (\log^c N)/\log^c N$$

Above summation can be reduced as:

$$N * N * [(1/(N/\log^c N(N/\log^c N * \log^c N + N)) + \dots]$$

$$N * N * \log^c N / N * (1/(N + N) + 1/(2(2N + N) + \dots + 1/(\log^c N(\log^c N * N + N)))$$

$$\text{above expression} \leq \log^c N (1/(N) + 1/(2^2 * (N)) + 1/(3^2 * (N)) + \dots + 1/(\log^{2c} N * (N)))$$

$$\text{above expression} \leq \log^c N (1 + 1/(2^2) + 1/(3^2) + \dots + 1/(\log^{2c} N))$$

$$\text{above expression} \leq (\log^c N) * \text{RiemannZetaFunction}(s = 2)$$

$$\text{above expression} \leq (\log^c N) * (\pi^2/6)$$

Thus total time for interpolation search of all row slices within all rectangles (from $N/\log^c N$ to N) is : $O((\log^c N / \sqrt{N}) * 3\log\log(q * N * \log N))$ or simply $O((\log^c N * \log\log(N))$

Above has excluded the strip of x-axis from 1 to $N/\log^c N$. This is just because it suffices to start the tesselation from some point on x-axis that is less than \sqrt{N} rather than from 1 and upto N. Also it helps in getting to Riemann Zeta Function with $s = 2$ to upperbound the summation. With this atleast 1 factor can be sieved out and to extract all factors above algorithm is recursively applied by dividing with already found factors. This process is only polynomial in $\log N$. Such a point can be found by suitably approximating the square root (actual square root finding is factorization-hard) and finding c . In other words:

Find c such that, $\sqrt{N} > N/\log^c N$ or $\log^c N > \sqrt{N}$ which reduces to: $c > \log N / 2\log\log N$. Before starting factorization such a constant c can be found satisfying above inequality.

If approximate square root finding is not preferable, above algorithm can be recursively applied to the strip of x-axis from 1 to $N/\log^c N$. This would give a series summation of upperbounds as: $S = O(\log^c N) + O(\log^c(N/\log^c N)) + O(\log^c(N/\log^{2c} N)) + \dots + O(\log^c(N/\log^{qc} N))$

$$S = O(\log(N) + \log(N/\log^c N) + \dots + \log(N/\log^{qc} N))^{c+1}$$

$$S = O((q+1)\log(N))^{c+1}$$

The above recursion stops when $\log(N/\log^{qc} N) = \log 1 = 0$.

Solving for q :

$$N = \log^{qc} N$$

$$\log(N) = qc(\log\log(N))$$

$q = \log N / (c\log\log N)$. Thus recursion stops after q iterations.

$$S = O((\log N / c\log\log N + 1)\log N)^{c+1}$$

Hence total running time is $O(((\log N / c\log\log N + 1)^{c+1}(\log N)^{c+1}))$ or $O(\log^{2c+2} N)$ with no restriction on choice of constant c . Value for c gives some flexibility in deciding the number of rectangles that tesselate the hyperbola. If we have $\log^c N$ number of processors which would imply an NC circuit, above factorization loop can infact be done using a logdepth, bounded width NC1 circuit with each rectangle assigned to one processor (if interpolation search per bow arc area within each rectangle is also in logdepth) and constant c decides the extent of parallelism.

Thus above series obviously converges with only polylog recursive iterations and thus total time taken is still polylog in input size. This proves the correctness of the polylog upperbound for Discrete Hyperbolic Polylogarithmic Sieve For Integer Factorization.

□

6 Acknowledgement

I dedicate this article to God.

7 Bibliography

References

- [1] Elementary algebraic and geometric principles
- [2] Binary Search, Interpolation Search
- [3] Riemann Zeta Function

Discrete Hyperbolic Polylogarithmic Sieve For Integer Factorization - using Interpolation Search (Draft)

Ka.Shrinivaasan (ka.shrinivaasan@gmail.com)

June 25, 2013

Abstract

Best known algorithm for factorization is the General number field sieve algorithm which takes time exponential in number bits in the input or simply called exponential algorithm. In this article, a polylogarithmic time (in size of input) sieve for factorization is discussed that uses elementary principles.

1 Discretization of a hyperbolic arc

1. A plot of a hyperbolic function is ($xy = N$) drawn on a 2-dimensional plane. This plot of hyperbola is continuous. This hyperbola can be discretized (or tesselated) with set of rectangles that cover the hyperbolic arc.
2. Discretization step divides the x-axis of the hyperbola into $\log^c N$ intervals where each interval is of length $N/\log^c N$. Infact it suffices to calculate upto \sqrt{N} length along the x-axis since atleast 1 factor must lie within \sqrt{N} . Correspondingly the y-axis is also split into intervals whose width is $y * \sqrt{N}/(x \log^c N + \sqrt{N})$ where (x, y) are coordinates of topleft corners of each rectangle.
3. The constant c is chosen to be quite large so as to shrink the size of each rectangle as much as possible. Thus each rectangle is of length $\sqrt{N}/\log^c N$ and width $y * \sqrt{N}/(x \log^c N + \sqrt{N})$ where (x, y) are coordinates of topleft corners of each rectangle. The factors (say p,q) of N are sure to be present within one of the rectangles if N is composite. Algorithm then is to sieve out rectangles to get to the factors. Looping through $\log^c N$ rectangles takes time $O(\log^c N * \text{TimePerRectangle})$.
4. Within each rectangle, the values of coordinates are hypothetically considered to be products of x and y coordinates. This is just supposed to exist and no lookup table or memoization is needed (though it can be made into a dynamic programming recurrence algorithm if knowledge of factors of numbers lower than N is assumed). To find the coordinates where N is located within the rectangle, Interpolation search, a variant of binary search that takes $\log \log N$ time on the average, is applied on the coordinate products with some specialized modifications as described in next section below.
5. The line equation corresponding to leading diagonal within each rectangle obtained previously, intersects with the hyperbola, thereby creating a bow-shaped area (hemmed between hyperbolic arc and diagonal) of coordinate products within each rectangle. This bow shaped area is only a small percentage of the area of each rectangle.

6. The row slices of coordinate products within this bow shaped area are in sorted order from left to right. Also the coordinate products in diagonal of the rectangle from bottom-left to top-right of each rectangle are in sorted ascending order. Thus searching these row slices of coordinate products (with some constant extension to each slice on leftside) within each rectangle is sufficient to sieve out the factors.

2 Interpolation search on the above bow-shaped area within each rectangle

1. The hyperbolic bow arc area obtained above can be searched with interpolation search on each of the row slice within a rectangle (along the y-axis).
2. Functions `InterpolationSearchSubroutine(xleft, yleft, xright, yright)`

```

{
  if(yleft == yright) //horizontal scan on a row slice yleft==yright
  {
    if (xleft+xright)/2 * yleft == N
    {
      return (xleft+xright)/2, yleft as factors of N
    }
    else if (xleft + xright)/2 * yleft > N
      InterpolationSearchSubroutine(xleft, yleft, (xleft+xright)/2, yright)
    else
      InterpolationSearchSubroutine((xleft+xright)/2, yleft, xright, yright)
    }
  }
}
```

3 Discrete Hyperbolic Polylogarithmic Sieve for Integer Factorization - Algorithm Steps

1. Before the factorization loop starts for a number N to be factorized, the constant c has to be chosen suitably. This constant decides the parallelism as described later.
2. Split interval $[0 - N]$ into $\log^c N$ intervals of width $N/\log^c N$ each.
3. For each of $\log^c N$ rectangle of width $N/\log^c N$ and height $y * \sqrt{N} / (x \log^c N + \sqrt{N})$ where (x, y) is the coordinate of top-left corner of each rectangle, compute the following loop:
 - (a) By Interpolation Search, using `InterpolationSearchSubroutine` described above, search each row slice within the bow shaped area confined within the hyperbolic area and the diagonal of the rectangle.
 - (b) Output $p, q (pq = N)$ if found in previous step and break; else continue loop

Low level calculations are derived later and also described in attached handwritten notes in appendix. This algorithm takes average upperbound time of $O(\log^c N * \log \log N)$.

4 Implications

As a consequence of this polylog algorithm for factorization, RSA cryptosystem's large prime key-pair can be found in polynomial time by factorization for which till now there are only BQP quantum polynomial algorithms available. Hence factorization in P implies probably that $BQP = P$ (if factorization is BQP-complete) since there is only a quantum polynomial algorithm for factorization and implies therefore that $BPP = P$ since BQP contains BPP (implying pseudorandomness and determinism are equivalent in computational power). This algorithm outputs all factors of an integer within polylogarithmic time in input bits and as a special case this is also a polynomial time primality test similar to PRIMES in P (AKS).

5 Derivation of the upperbound time for discrete hyperbolic factorization

Theorem 1. *Discrete Hyperbolic Polylogarithmic Factorization has an average upperbound running time of $O(\log^{2c+2} N)$*

Proof. Area of hyperbolic bow area per rectangle = Area of rectangle - Area of hyperbolic arc (obtained by area definite integral)

$$= (1/2 * N/\log^c N * y_1 \sqrt{N}) / (2 * \log^c N * (x_1 \log^c + \sqrt{N})) - N \log(x_1 + N/\log^c N) + N \log x_1$$

Dividing above bow area by y-axis side of the rectangle deltay gives average length of each hyperbolic row slice within bow: $((1/2 * N/\log^c N * y_1 \sqrt{N}) / (2 * \log^c N * (x_1 \log^c + \sqrt{N})) - N \log(x_1 + N/\log^c N) + N \log x_1) / (y_1 * \sqrt{N}) / (x_1 * (\log^c N) + \sqrt{N}))$

The above is average length of each hyperbolic row slice within bow area that needs to be interpolation searched. Time taken for interpolation search of row slice that is in ascending order from left to right will be $\log\log(\text{aboveexpression})$:

$$\log\log((1/2 * N/\log^c N * y_1 \sqrt{N}) / (2 * \log^c N * (x_1 \log^c + \sqrt{N})) - N \log(x_1 + N/\log^c N) + N \log x_1) / (y_1 * \sqrt{N}) / (x_1 * (\log^c N) + \sqrt{N}))$$

Above term can be upperbounded as follows ignoring some subtraction and division terms:

$$\text{Aboveexpression} \leq \log\log((1/2 * N/\log^c N * y_1 \sqrt{N}) / (2 * \log^c N * (x_1 \log^c + \sqrt{N})) + N \log x_1) / (y_1 * \sqrt{N}) / (x_1 * (\log^c N) + \sqrt{N}))$$

$$\text{Aboveexpression} \leq \log\log((N * y_1 \sqrt{N}) + N \log x_1) * 2 * \log^c N * (x_1 \log^c N + \sqrt{N}) / (2 * \log^c N * y_1 * \sqrt{N}))$$

$$\text{Aboveexpression} \leq \log\log((N/2 \log^c N) + (N \log x_1 (x_1 \log^c N + \sqrt{N}))) / (y_1 * \sqrt{N})) \text{ Since } y_1 = N/x_1, \text{ Aboveexpression} \leq \log\log((N/2 \log^c N) + (x_1 * \log x_1 (x_1 \log^c N + \sqrt{N}))) / (\sqrt{N}))$$

x_1 can take \sqrt{N} as maximum (if only \sqrt{N} fraction of x-axis is scanned with rectangular sieves):

$$\text{Aboveexpression} \leq \log\log((N/2 \log^c N) + (\log \sqrt{N} (\sqrt{N} \log^c N + \sqrt{N})))$$

$$\text{Aboveexpression} \leq \log\log(N + (\sqrt{N}/2)(\log^{c+1} N + \log N))$$

$$\text{Aboveexpression} \leq \log\log(2 * N * \log^{c+1} N)$$

$$\text{Aboveexpression} \leq \log\log(2 * N * \log^{c+1} N) \text{ or } \log\log(2 * N * (c+1) * \log N)$$

Thus time per row slice interpolation search per rectangle is upperbounded by:

$$\log\log(q * N * \log N) \text{ for } q = 2c + 2$$

Multiplying above upperbound for deltay_i rows within each rectangle_i and for all $\log^c N$ rectangles, gives the total running time:

$$\sum_{N/\log^c N}^N N \sqrt{N} / (x_1 (x_1 \log^c N + \sqrt{N})) \text{ with } \text{deltax} = N/\log^c N$$

Ignoring $x_i = 1$ consecutive $x_i(s)$ are: (each rectangle is $N/\log^c N$ wide)

$$N/\log^c N, 2N/\log^c N, 3N/\log^c N, \dots, (\log^c N)N/\log^c N$$

Above summation can be reduced as:

$$\begin{aligned}
& N\sqrt{N} * [(1/(N/\log^c N(N/\log^c N * \log^c N + \sqrt{N}))) + \dots] \\
& N\sqrt{N} * \log^c N / N * (1/(N + \sqrt{N}) + 1/(2(2N + \sqrt{N})) + \dots + 1/(\log^c N(\log^c N * N + \sqrt{N}))) \\
& \text{above expression} \leq \log^c N(1/\sqrt{N} + 1/(2^2\sqrt{N}) + 1/(3^2\sqrt{N}) + \dots + 1/(\log^{2c} N\sqrt{N})) \\
& \text{above expression} \leq \log^c N(1 + 1/(2^2) + 1/(3^2) + \dots + 1/(\log^{2c} N)) \\
& \text{above expression} \leq (\log^c N/\sqrt{N}) * \text{RiemannZetaFunction}(s = 2) \\
& \text{above expression} \leq (\log^c N/\sqrt{N}) * (\pi^2/6)
\end{aligned}$$

Thus total time for interpolation search of all row slices within all rectangles is : $O((\log^c N/\sqrt{N}) * 3\log\log(q * N * \log N))$ or simply $O((\log^c N * \log\log(N))$

Above has excluded the strip of x-axis from 1 to $N/\log^c N$. This is just because it suffices to start the tesselation from some point on x-axis that is less than \sqrt{N} rather than from 1 and upto N. Also it helps in getting to Riemann Zeta Function with $s = 2$ to upperbound the summation. With this atleast 1 factor can be sieved out and to extract all factors above algorithm is recursively applied by dividing with already found factors. This process is only polynomial in $\log N$. Such a point can be found by suitably approximating the square root (actual square root finding is factorization-hard) and finding c . In other words:

Find c such that, $\sqrt{N} > N/\log^c N$ or $\log^c N > \sqrt{N}$ which reduces to: $c > \log N/2\log\log N$. Before starting factorization such a constant c can be found satisfying above inequality.

If approximate square root finding is not preferable, above algorithm can be recursively applied to the strip of x-axis from 1 to $N/\log^c N$. This would give a series summation of upperbounds as:

$$\begin{aligned}
S &= O(\log^c N) + O(\log^c(N/\log^c N)) + O(\log^c(N/\log^{2c} N)) + \dots + O(\log^c(N/\log^{qc} N)) \\
S &= O(\log(N) + \log(N/\log^c N) + \dots + \log(N/\log^{qc} N))^{c+1} \\
S &= O((q+1)\log(N))^{c+1}
\end{aligned}$$

The above recursion stops when $\log(N/\log^{qc} N) = \log 1 = 0$.

Solving for q :

$$N = \log^{qc} N$$

$$\log(N) = qc(\log\log(N))$$

$q = \log N / (\log\log N)$. Thus recursion stops after q iterations.

$$S = O((\log N / \log\log N + 1)\log N)^{c+1}$$

Hence total running time is $O(((\log N / \log\log N + 1)^{c+1}(\log N)^{c+1}))$ or $O(\log^{2c+2} N)$ with no restriction on choice of constant c . Value for c gives some flexibility in deciding the number of rectangles that tessellate the hyperbola. If we have $\log^c N$ number of processors which would imply an NC circuit, above factorization loop can infact be done using a logdepth, bounded width NC1 circuit with each rectangle assigned to one processor (if interpolation search per bow arc area within each rectangle is also in logdepth) and constant c decides the extent of parallelism.

Thus above series obviously converges with only polylog recursive iterations and thus total time taken is still polylog in input size. This proves the correctness of the polylog upperbound for Discrete Hyperbolic Polylogarithmic Sieve For Integer Factorization.

□

6 Acknowledgement

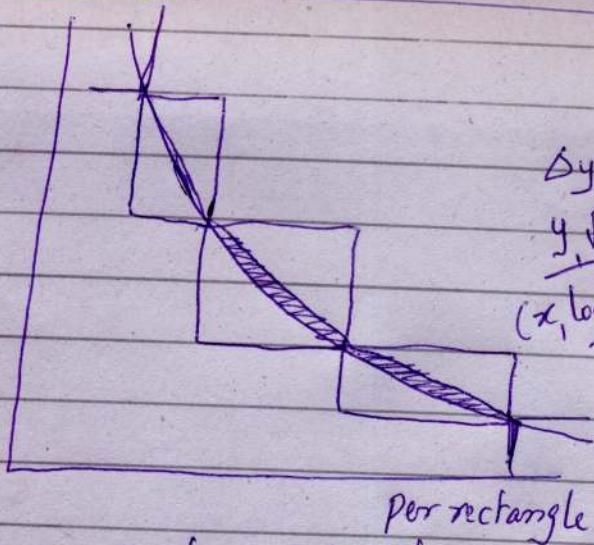
I dedicate this article to God.

7 Appendix 1 - Handwritten calculations for lowerbound derivation

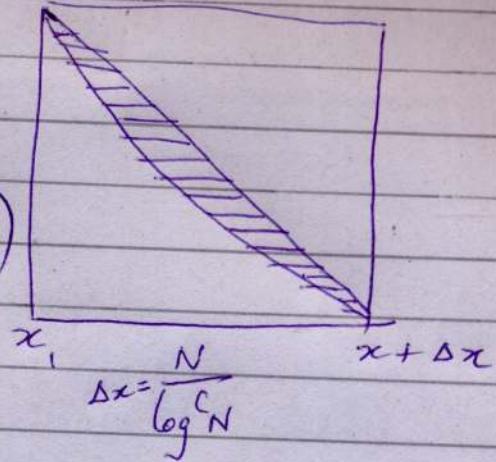
Bow shaped area - interpolation search - Discrete Hyperbolic
 Polylogarithmic Factorization

20/6/2013 Ka-Shinnirae

(Re-Re-Derived to verify)



$$\Delta y = \frac{y_i \sqrt{N}}{(x_i \log^c N + \sqrt{N})}$$



Area of hyperbolic bow : Area of rectangle - Area of hyperbolic bow

$$= \frac{1}{2} \frac{N}{\log^c N} \times \frac{y_i \sqrt{N}}{(x_i \log^c N + \sqrt{N})} - N \log \left(x_i + \frac{N}{\log^c N} \right) + N \log x_i$$

Dividing above bow area by Δy gives average length of each hyperbolic row slice within bow : (Straightened bow and divide)

$$\left[\frac{N y_i \sqrt{N}}{2 \log^c N (x_i \log^c N + \sqrt{N})} - N \log \left(x_i + \frac{N}{\log^c N} \right) + N \log x_i \right] \frac{y_i \sqrt{N}}{(x_i \log^c N + \sqrt{N})}$$

$$\leq \log \log \left[\frac{N}{2 \log^c N} + \frac{x \sqrt{N} (\log x)}{\sqrt{N}} (\log^c N + \sqrt{N}) \right]$$

x can take \sqrt{N} as maximum (it is sufficient upto \sqrt{N})

$$\leq \log \log \left[\frac{N}{2 \log^c N} + \frac{\sqrt{N}}{\sqrt{N}} \log \sqrt{N} (\sqrt{N} \log^c N + \sqrt{N}) \right]$$

$$\leq \log \log \left[\frac{N}{2 \log^c N} + \sqrt{N} \log \sqrt{N} (\log^c N + 1) \right]$$

$$\leq \log \log \left[N + \frac{\sqrt{N}}{2} \left(\log^{(c+1)} N \right) + \frac{\sqrt{N}}{2} \log N \right]$$

$$\leq \log \log \left[N + \frac{\sqrt{N}}{2} \left(\log^{(c+1)} N + \log N \right) \right] \leq \log \log \left[N + \frac{\sqrt{N}}{2} \right]$$

$$\leq \log \log \left((2c+2)N^{q^2} \right) \quad \log N < N \quad N \log N < N$$

$$\leq \log \log (q^2 N^2) \leq \boxed{\log_2 \log (qN)} \quad \text{if } q^2 =$$

Thus time per row slice interpolation search is upper bounded by :

$O(\log(2 \log qN))$ and thus poly-loglog of

Multiplying above upperbound for Δy_i rows/rec and $\log^c N$ rectangles are :

$$\left[\sum_{i=1}^{\log^c N} \left[\frac{(y_i \sqrt{N}) \log (2 \log qN)}{(x_i \log^c N + \sqrt{N})} \right] \right] \times \left[\log (2 \log qN) \right]$$

$$y_i = \frac{N}{x_i}$$

Summation above is expanded as :

$$\log^c N$$

Thus consecutive $x_i(s)$ are : (each rectangle $\rightarrow \frac{N}{\log^c N}$ width)

$$\frac{N}{\log^c N}, \frac{2N}{\log^c N}, \frac{3N}{\log^c N}, \dots, \frac{(\log^c N)N}{(\log^c N)}$$

$$N\sqrt{N} \left[\frac{1}{\frac{N}{\log^c N} \left(\frac{N}{\log^c N} \log^c N + \sqrt{N} \right)} + \frac{1}{\frac{2N}{\log^c N} \left(\frac{2N}{\log^c N} \log^c N + \sqrt{N} \right)} + \frac{1}{\frac{3N}{\log^c N} \left(\frac{3N}{\log^c N} \log^c N + \sqrt{N} \right)} + \dots + \frac{1}{\frac{(\log^c N)N}{\log^c N} \left(\frac{(\log^c N)N}{\log^c N} \log^c N + \sqrt{N} \right)} \right]$$

$$= \frac{N\sqrt{N} \log^c N}{N} \left[\frac{1}{(N + \sqrt{N})} + \frac{1}{2(N + \sqrt{N})} + \frac{1}{3(3N + \sqrt{N})} + \dots + \frac{1}{\log^c N} \right]$$

$$= \frac{N\sqrt{N} \log^c N}{N\sqrt{N}} \left[\frac{1}{(\sqrt{N} + 1)} + \frac{1}{2(\sqrt{N} + 1)} + \frac{1}{3(\sqrt{N} + 1)} + \dots + \frac{1}{\log^c N} \right]$$

$$\leq \frac{\log^c N}{\sqrt{N}} \left(\frac{\pi^2}{6} \right)$$

RZF($s=2$) converges to $\frac{\pi^2}{6}$
 is upper bound for sum
 all Δy_i $i=1, 2, \dots$

Thus Total time for all row slices' interpolation
 is: $\underbrace{\text{Sum of } \Delta y_i}_{\text{per row slice search upperbound.}}$

$$\leq \left[\frac{\log^c N}{\sqrt{N}} (3) \right] \left[\log \left(2 \log(g_N) \right) \right] \quad \left[\frac{\pi^2}{6} < 3 \right]$$

$$\leq \log^c N \log \left(2 \log(g_N) \right) \quad \left[O \left((\log^c N) (\log \log(N)) \right) \right]$$

$$\leq \log^c N \log N \quad (\text{since } O(N) > 2 \log(g_N)).$$

$$\leq \log^{c+1} N = \boxed{O(\log^{c+1} N)}$$

Thus polylog upper bound for Discrete hyperbolic is proved correct.

$$S^{\frac{1}{c+1}} < \log(N) + \log\left(\frac{N}{\log^c N}\right) + \log\left(\frac{N}{\log^{2c} N}\right) + \dots$$

$$S^{\frac{1}{c+1}} < \log(N) + \frac{\log(N)}{\log N} + \frac{\log(N)}{\log^{2c} N} + \dots$$

$$S^{\frac{1}{c+1}} < \frac{\log(N)}{1 - \frac{1}{\log N}}$$

$$S^{\frac{1}{c+1}} <$$

$$\frac{\log(N) \log^c N}{\log^c N - 1}$$

$$a^b < a^b$$

$$a^{b-1} < b^{b-1}$$

$$\Rightarrow \left(\frac{1}{c+1}\right) \log S < \log\left(\frac{N \log^c N}{\log^c N - 1}\right)$$

$$\log S < (c+1) \log\left(\frac{N \log^c N}{\log^c N - 1}\right)$$

$$\text{if } a < b$$

$$\left(\frac{a}{b}\right)^{b-1} < 1$$

$$\frac{a}{r^b (r-1)}$$

23/6/2013

Recursive Discrete Hyperbolic Factorisation
sufficient for polylog:

$$S = \log^{c+1} N + \log^{c+1}\left(\frac{N}{\log^c N}\right) + \log^{c+1}\left(\frac{N}{\log^{2c} N}\right) + \dots$$

$$S < (1 + 1/N + \dots) \cdot N^{c+1}$$

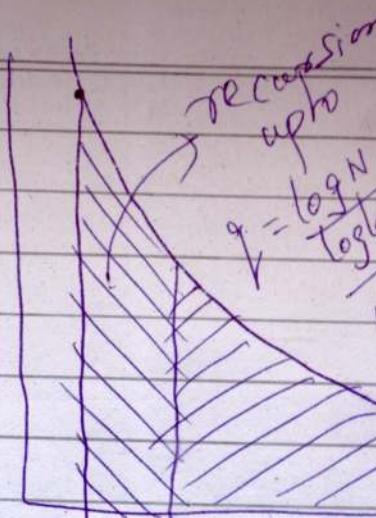
$$S \leq \left((q+1) \log N \right)^{c+1}$$

$$\text{But } \log \frac{N}{\log \frac{N}{\log N}} = \log 1$$

$$\frac{N}{\log N} = \frac{N}{\log \frac{N}{\log N}}$$

$$\log N = (q+1) \log \log N$$

$$\left[\frac{\log N}{\log \frac{\log N}{C}} = q \right] \quad \begin{array}{l} \text{recursion} \\ \text{stops after} \\ q \text{ levels within } O\left(\frac{\log N}{\log \log N}\right) \end{array}$$



$$\Rightarrow S \leq \left(\frac{\log N}{\log \frac{\log N}{C}} \log N \right)^{c+1}$$

with no rest
in choice of

$$\Rightarrow S \leq \left(\log^2 N \right)^{c+1}$$

$$S \leq \log^{2c+2} N$$

$$\left(\frac{\log N}{\log \frac{\log N}{C}} \right)^{c+1} \log^{c+2}$$

$$S = O\left(\log^{2c+2} N\right)$$

V Shrin

8 Bibliography

References

- [1] Elementary algebraic and geometric principles
- [2] Binary Search, Interpolation Search
- [3] Riemann Zeta Function

Document Summarization from WordNet Subgraph obtained by Recursive Gloss Overlap - Draft

*SrinivasanKannan(alias)Ka.Shrinivaasan(alias)ShrinivasKannan
IndependentOpenSourceDeveloper, Researcher and Consultant*

Ph : 9789346927, 9003082186, 9791165980

*KrishnaiResearchOpenSourceProducts : http://sourceforge.net/users/ka_shrinivaasan,
https://www.ohloh.net/accounts/ka_shrinivaasan
ResearchWebsite : https://sites.google.com/site/kuja27/
(ka.shrinivaasan@gmail.com, shrinivas.kannan@gmail.com, kashrinivaasan@live.com)*

July 25, 2014

Abstract

This article expands on the idea of construction of WordNet subgraph of a text document mentioned in references and studies the spectral properties of the WordNet subgraph and its Summarizability

1 Document Summarization and Recursive Gloss Overlap Algorithm

1. WordNet Subgraph is created for a document using Recursive Gloss Overlap algorithm as mentioned in the references below. Nodes of this graph are words and edges are relations between words. Breadth First Search or Depth First Search of this graph yields a flattened out meaning contained in the document. This just preserves the approximate meaning. The BFS or DFS prints "word1", "relation12" and "word2" iteratively for all nodes and edges. The grammar used for summary is just a concatenation of texts "word1", "relation12" and "word2".
2. Above BFS and DFS would find all the connected components of the WordNet subgraph. Thus summary of the document can be obtained by just concentrating on top few connected components (in terms of size). This is because size of a connected component in the WordNet subgraph implies the centrality or importance of that component to the document.
3. WordNet subgraph from Recursive Gloss Overlap is quite flexible as to what depth the definition graph or tree has to be grown. Once definition tree has been constructed, removal of isomorphic nodes or subtrees does away with duplication and gives a pruned wordnet subgraph with unique vertices and edges.
4. Mapping a text document to a graph as above is quite useful in applying lot of graph theorems and techniques on the wordnet subgraph.

5. As Document Summarization is basically an effort to highlight most important parts of a larger document, above graph precisely gives importance of a word to a document in terms of degree (incoming + outgoing edges) of that word vertex. More edges emanating from or connecting to a word implies that the word is a hub with edges as spokes to the document. This is an alternative measure to Term Frequency and Inverse Document Frequency for importance of a keyword. Moreover the degree is arrived after indepth growing of the Definition Tree which is tantamount to "human reasoning" or "studying in depth".Degrees of words could change if the depth to which the definition tree grown is altered which is commonsensical to human thinking process of in depth studying of a text yielding new insights.
6. Sorting the WordNet Subgraph nodes on the degree and sieving out nodes with less degrees is an effective summarization technique.
7. Random Walks on the WordNet subgraph of a document and their mixing time is a plausible measure of effort needed to understand a document (to get to Stationary Distribution)
8. Graph Spectra on the Laplacian of the WordNet Subgraph of a document could yield interesting details on the number of connected components (multiplicity of zero eigen value), spectral gap and algebraic connectivity (smallest and second smallest eigen values). Summarizability is inversely proportional to number of connected components and directly proportional to Algebraic Connectivity. If the WordNet Subgraph is more "Expanding" (i.e minimum of ratio of neighbours of all subset of vertices to the size of all subset of vertices) then that implies that the document has more internal connected structures and is summarizable.

2 Algorithm for Document Summarization from WordNet Subgraph

9. Algorithm for a summary (Theory only, without any experimental data):
 - (a) Find word vertices of high degree above a threshold and their adjacent words and all paths amongst these high degree word vertex hubs. Add them to a new graph.
 - (b) This gives a subgraph of high degree vertex hubs of WordNet Subgraph (Subgraph of a WordNet Subgraph).
 - (c) Do BFS or DFS traversal and output sentences as "word(x)" "relation-xy" "word(y)" for all edges (x,y) iteratively for this high degree subgraph. This is the typical Hub-Spokes Model.
 - (d) Without loss of generality, the spokes could have length of more than one edge. This captures the most important parts of a document and prunes others because high degree word vertices affect lot of other nodes in WordNet subgraph.
 - (e) Hub words have more relations to adjacent words than non-Hub words in this graph.
10. Above algorithm can also optionally use PageRank algorithm to compute the ranks of highly important word vertices and choose only vertices above a threshold rank in constructing the subgraph of WordNet subgraph as above instead of degrees. (Importance of a word increases if it is pointed to by another important word). The graph is WordNet subgraph instead of hyperlink graph.

11. If the WordNet gives Parts-of-Speech information also, then more complex sentences can be constructed from the BFS or DFS traversal output. Probably this might need 2 passes - First pass creates simple sentences as above and creates a mapping table of the words to parts of speech during DFS or BFS and the second pass uses more complex sentence PoS template (e.g "Noun Pronoun Adjective Verb Obj") and looks up the table constructed in previous iteration to coalesce two or more simple sentences into complex sentences based on common substrings between two sentences. This coalescing can be hierarchically continued as a tree bottom-up to form more complex sentences from simple sentences.
12. Accuracy of the summary is dependent on the WordNet and also on the Summarizability measured by spectral properties above. If the WordNet is replaced by some other domain specific Ontology, relevance or meaningfulness of summary might increase. Above algorithm does not give much importance to very complex sentence formations. Only sentences of the form "word1 relates to word2" and their hierarchical coalitions are focussed. Complex sentences can be also obtained by transitive closure of the word-word relations. The Kleene closure of the above graph would give all possible such sentence formations. Above notwithstanding, the crucial meaning of the larger document is conveyed by the Summarization algorithm previously described.

3 Acknowledgement

I dedicate this article to God.

4 Bibliography

References

- [1] Algorithms for Intrinsic Merit of a Document - Recursive Gloss Overlap Algorithm for wordnet subgraph - <http://arxiv.org/abs/1006.4458>
- [2] Slides illustrating WordNet subgraph or Definition Graph Construction using Recursive Gloss Overlap Algorithm - <https://sites.google.com/site/kuja27/PresentationTAC2010.pdf?attredirects=0>
- [3] Primitive implementation of the above at: <http://sourceforge.net/p/asfer/code/HEAD/tree/python-src/InterviewAlgorithm/>
- [4] TAC 2010 - Update Summarization using Interview Algorithm - <http://www.nist.gov/tac/publications/2010/participant.papers/CMIIIT.proceedings.pdf>
- [5] WordNet - <http://wordnet.princeton.edu/>
- [6] Other publication drafts on Majority Voting and Interview Algorithm related to the above in <https://sites.google.com/site/kuja27/>

2 January 1994

DEC '12

half or more SAT inputs January 3

2013 CAP Theorem with
Wednesday for Kim & Cebra [In progress]

10/1/2014

AsFeX - optional to do - add Apache Pig Hive, Tez support for astronomical datasets.
(and) HBase, NoSQL support.

The minimum cover will of all probability

Random growth networks (in previous page) cannot be of polynomial size if $P \neq NP$. (This intuitively implies that minimum learning needed which is symbolized by Random growth network implication graph cannot be a polynomial in length of proof geodesic that is contained within minimum convex hull.)

LHS is in P. In RHS each SAT voter has to be satisfied which is the democracy problem making it NP-Complete. If all voters are zero error RHS is NP-Complete since NC1 majority circuit takes inputs from SAT voters. Thus LHS is P as RHS is in NP if voters are zero error.

For zero-error [the entropy has to be zero] the voter has to make perfect decision in all possible scenarios -

Does such a voter exist?

That

$\sum_i (Total_i - Observed_i) = 0$ (Decision entropy is zero)

Number of all possible scenarios is infinite.
So above question itself could be undecidable and is RE & not recursive. Does that imply

So above question itself could be undecidable and is RE & not recursive. Does that imply $P \neq NP$ is undecidable?

Nc1 majority

Circuit with SAT inputs

BPP or BPNC

BPP or BPNC

Scenarios of decision

	S	M	T	W	T	F	S
30	31	1	2	3	4	5	6
9	10	11	12	13	14	15	16
16	17	18	19	20	21	22	23
23	24	25	26	27	28	29	

	S	M	T	W	T	F	S
3	4	5	6	7	8	9	
10	11	12	13	14	15	16	
17	18	19	20	21	22	23	
24	25	26	27	28	29		

Saturday

$P \stackrel{?}{=} NP$ is decidable.

Pseudorandom choice:

If $x\%$ of voters are imperfect, probability that selected choice is good

$$100 = [x]\% = (100 - x)\%$$

Majority Voting:

If $x\%$ of voters are imperfect, probability that elected choice is good

$$= (100 - x)\%$$

If ideal perfect — making with function

$f(\cancel{d}, \text{Scenario}) = \text{decision}$ and actual decision making is the function

$$f(\text{Scenario}) = \text{decision}$$

$$\text{then the integral } \int f(g) - g(g) \neq 0$$

for all perfect voters. Or

$\int |f(g) - g(g)| > 0$ for $x\%$ of voters

An imperfect vote is an imperfect assignment to a Voter-SAT CNF. That is Voter CNFSAT is probabilistic.

Impossibility lemma: (Proof as in previous page)

Election of perfect leader is possible iff voters are all perfect and zero-error in both pseudorandom choice and majority voting.

(To verify: If this is Kenneth Arrow's impossibility theorem rephrased.)

Above counterexample $P = NP$ conclusion is for the question "Is perfect election of leader possible?" and has P (prob) equation circuit as its basis.

Sunday 6.

Argument based on law of thermodynamics:

Entropy in a system is non-decreasing if \mathcal{D} only at Kelvin, which is unattainable. If non-zero entropy (disorder)

7 January
2013

Monday

in a system) is thought of as source of any error of judgement: thus the rules out

perfect judgement. Thus thermodynamics $\Rightarrow P \neq NP$

hopkins entropy in

Ka Shrinivasan

10/01/2014

Both arguments below:

- 1) Implication graphs - implication is undecidable
- 2) Perfect Vicker existence is undecidable (using P(Geo) circuits)
lead to $P \neq NP$ undecidability (in infinite case of sumnos and in most fragments of logic).
Some fragments have decidable implications.
Both paths are independent of each other.

Above is a non-conventional non-textbook unnatural proof approach to $P \neq NP$. Since diagram proofs won't work, above unnatural paths are worth exploring.

Ka Shrinivasan

11/1/2014

2013

Tuesday

S	M	T	W	T	F	S
30	31					
2	3	4	5	6	7	1
9	10	11	12	13	14	8
16	17	18	19	20	21	15
23	24	25	26	27	28	29
24	25	26	27	28	29	

S	M	T	W	T	F	S
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	

January
2013

Tuesday

2013

Wednesday

Indepth analysis of a variant of Majority Voting and relation to Zermelo-Fraenkel Set Thoery With Axiom of Choice (ZFC) - Updated Draft

*SrinivasanKannan(alias)Ka.Shrinivaasan(alias)ShrinivasKannan
IndependentOpenSourceDeveloper, ResearcherandConsultant*

Ph : 9789346927, 9003082186, 9791165980

*KrishnaiResearchOpenSourceProducts : http://sourceforge.net/users/ka_shrinivaasan,
https://www.ohloh.net/accounts/ka_shrinivaasan
ResearchWebsite : <https://sites.google.com/site/kuja27/>
(ka.shrinivaasan@gmail.com, shrinivas.kannan@gmail.com, kashrinivaasan@live.com)*

February 8, 2014

Abstract

As a generalization on Majority voting and pseudorandom choice, a BPNC majority voting circuit with oracle access to a choice function is presented and the choice function (which in the infinite case would be a choice function for ZFC) is defined in this article.

1 Majority voting circuit and pseudorandom choice are in BPNC

It is a known result that $BPNC^k$ is contained in BPP. Earlier a majority voting circuit with BPP voter oracle circuit lying in $NC1^{BPP}$ was explained. It was also mentioned that instead of an oracle voter, the majority voting circuit can be formulated alternatively as a non-uniform BPNC circuit. But pseudorandom choice was shown to be in BPP using Nisan's Pseudo random generator. Since BPNC is contained in BPP, this probably gives a counterintuitive result that majority voting is more efficient compared to a pseudorandom choice but both have a same bounded error derived using P(Good) expression. But recently there are pseudorandom generators constructed in NC0 with linear stretch. Thus there is a symmetry for the LHS and RHS of the P(good) equation with both Pseudorandom choice with bounded error and Majority voting with bounded error lying in BPNC class. But the caveat is that the majority voting is non-uniform. This is an improved bound as against $NC1^{BPP}$ oracle circuit for majority voting and BPP machine for pseudorandom choice.

2 Majority and Parity

By the result of Furst,Saxe and Sipser Parity is not in AC0. It is also proved by Hastad's switching lemma implying Parity cannot be computed by constant depth circuits of polynomial size and unbounded fanin. There is a simple reduction from parity to majority function where sum of bits are computed by a 3-to-2 way addition of 3 numbers and parity (XOR) is used in the computation. Thus parity function forms the building block of entire majority voting circuit. Since parity is not computable in AC0, the majority circuit with bounded error cannot be computed in constant depth and this justifies the BPNC bound obtained above.

3 Zermelo-Fraenkel-Axiom-of-Choice (ZFC) and its relevance to majority voting

Axiom of choice in set theory postulates that given a set X (possibly infinite) of non-empty sets, there always exists a choice function that chooses one element from each element-set in X . Though existence of choice function for infinite case is controversial, AOC has been pushed into ZermeloFraenkel Set theory to make it ZFC set theory (ZF with AOC). Now let us consider a majority voting scenario where the voting population is partitioned into subsets called constituencies and there exists a choice function that chooses one representative element from each constituency. Thus set of representatives from each constituency chosen by choice function form an electoral collegium denoted as EC. This EC can be modelled as a majority voting circuit with each chosen representative of EC as an input to the majority circuit. Thus we have a voting system with two levels.

4 Definition of a choice function and circuits for choice function computation

Let m be the size of voting citizens in a constituency and $q < m$ be the number of candidates.

Each of the q candidates get votes $v(i)$ for $i=1,2,3,\dots,q$.
Thus $v(1) + v(2) + v(3) + \dots + v(q) = m$

Above is a partition of voting population. The choice function C is defined as follows:

$C(m,q) = \text{index } \text{maxI of the biggest valued part in the integer partition of } m \text{ as defined above , where } 1 < \text{maxI} < q$

By Axiom of Choice, there exists a choice function for choosing a representative from each constituency in infinite case. But it needs to be well-defined that the above choice function always returns a unique index of biggest part in the partition. Unfortunately partition of votes as defined by the choice function can have multiple similar valued biggest parts (which requires a tie- breaker criterion) and might return a set in the absence of a tiebreaker. Thus any voting system similar to above choice function allowing a tie has an inherent limitation. A turing machine computing the choice function for a constituency takes as input the voting population and candidates, initiates a voting on candidates, computes a partition of the voting population into votes for individual candidates, sorts the parts in the partition and outputs the index of the candidate with largest vote (assuming a tiebreaker). Aggregating votes for candidates involves addition NC1 circuits for each of the candidates where voters cast their vote by giving inputs into the NC1 circuit of the corresponding candidate. Outputs of each of the addition circuits form a partition of the voting population. These outputs are parts in the partition which can be sorted using a sorting network to get the largest indexed candidate part. For voting with bounded error these addition NC1 circuits are BPNC circuits with error and feed to a sorting network of polynomial size and logdepth (AKS sorting circuits). Thus the choice function oracle circuit is in BPNC of polylog depth and polynomial size.

5 Circuit for majority voting with constituencies as described above

The non-uniform NC1 majority voting circuit can be modified to query a choice function oracle described above and get the inputs(representatives) for majority gate collegium. Using BPNC choice function described above as oracle for electoral collegium, we obtain a $BPNC^{BPNC}$ circuit. It is not known if BPNC is low for itself like BPP. Thus a BPNC circuit with oracle access to a BPNC choice function has been arrived at for the special case of majority voting with constituencies and electoral collegium.

6 Future directions

Handwritten notes on perfect voter decidability and relation to Arrow's Impossibility Theorem, NP-Completeness of Democracy, Relation between hash functions and integer partitions by the author in references below might further strengthen the above result. For instance, for above Choice function, a hash function that unequally distributes is needed (no two keys have equally long buckets of values hashed into them). Thus number of such hash functions indirectly give all possible voting patterns with unique elected choice. If coalitions are also mathematically defined, more fine-grained complexity model for practical multiparty majority voting can be obtained.

7 Acknowledgements

I dedicate this article to God.

8 Bibliography

References

- [1] On Type-2 Probabilistic Quantifiers, Automata, Languages and Programming: 23rd international colloquium, ICALP
- [2] Pseudorandom generators with linear stretch in NC0, Berni Applebaum, Yuval Ishai, Eyal Kushilevitz, Computer science department, Technion, Israel.
- [3] Various resources on Choice functions and Zermelo-Fraenkel-Axiom-of-Choice set theory
- [4] TAC 2010 – Update summarization through interview algorithm - http://www.nist.gov/tac/publications/2010/participant.papers/CMI_IT.proceedings.pdf
- [5] Few Algorithms for ascertaining merit of a document and their applications - <http://arxiv.org/abs/1006.4458>
- [6] Various resources on Integer Partitioning
- [7] Parity not in AC0 - Furst, Saxe and Sipser
- [8] Hastad's Switching Lemma, John Hastad's PhD thesis, MIT
- [9] Majority is in non-uniform NC1, Mix Barrington
- [10] Sorting networks of Ajtai-Komlos-Szemeredi

- [11] Integer Partitions and Hash Functions - <https://sites.google.com/site/kuja27/IntegerPartitionAndHashF0>
- [12] Informal notes - 3 : on Minimum Convex Hulls of Implication Random Growth Networks and Perfect Voter Decidability - <https://sites.google.com/site/kuja27/ImplicationRandomGraphConvexHullsAndPerfectVoterProblem2014-01-11.pdf?attredirects=0>
- [13] Informal notes - 2 : on Minimum Convex Hulls of Implication Graphs and Hidden Markov Model on class nodes of Concept Hypergraph - <https://sites.google.com/site/kuja27/NotesOnConceptHypergraphHMMandImplicationGraphConvexHulls2012-30.pdf?attredirects=0>
- [14] Informal notes - 1 : on Implication Graphs, Error probability of Majority Voting and P Versus NP Question <http://sourceforge.net/projects/acadpdrafts/files/ImplicationGraphsPGoodEquationAndPNotEqualToNP0>
- [15] Lower Bounds for Majority Voting and Pseudorandom Choice - <https://sites.google.com/site/kuja27/LowerBoundsForMajorityVotingPseudorandomChoice.pdf?attredirects=0>
- [16] Circuit For Computing Error Probability of Majority Voting and Pseudorandom Choice - <https://sites.google.com/site/kuja27/CircuitForComputingErrorProbabilityOfMajorityVoting.pdf?attredirects=0>
- [17] Interview Algorithm is in IP=PSPACE - <https://sites.google.com/site/kuja27/InterviewAlgorithmInPSPACE0>

Indepth analysis of a variant of Majority Voting and relation to Zermelo-Fraenkel Set Theory With Axiom of Choice (ZFC)

Ka.Shrinivaasan (ka.shrinivaasan@gmail.com)

March 8, 2013

Abstract

As a generalization on Majority voting and pseudorandom choice, a BPNC majority voting circuit with oracle access to a choice function is presented and the choice function (which in the infinite case would be a choice function for ZFC) is defined in this article.

1 Majority voting circuit and pseudorandom choice are in BPNC

It is a known result that $BPNC^k$ is contained in BPP. Earlier a majority voting circuit with BPP voter oracle circuit lying in $NC1^{BPP}$ was explained. It was also mentioned that instead of an oracle voter, the majority voting circuit can be formulated alternatively as a non-uniform BPNC circuit. But pseudorandom choice was shown to be in BPP using Nisan's Pseudo random generator. Since BPNC is contained in BPP, this probably gives a counterintuitive result that majority voting is more efficient compared to a pseudorandom choice but both have a same bounded error derived using P(Good) expression. But recently there are pseudorandom generators constructed in NC0 with linear stretch. Thus there is a symmetry for the LHS and RHS of the P(good) equation with both Pseudorandom choice with bounded error and Majority voting with bounded error lying in BPNC class. But the caveat is that the majority voting is non-uniform. This is an improved bound as against $NC1^{BPP}$ oracle circuit for majority voting and BPP machine for pseudorandom choice.

2 Majority and Parity

By the result of Furst,Saxe and Sipser Parity is not in AC0. It is also proved by Hastad's switching lemma implying Parity cannot be computed by constant depth circuits of polynomial size and unbounded fanin. There is a simple reduction from parity to majority function where sum of bits are computed by a 3-to-2 way addition of 3 numbers and parity (XOR) is used in the computation. Thus parity function forms the building block of entire majority voting circuit. Since parity is not computable in AC0, the majority circuit with bounded error cannot be computed in constant depth and this justifies the BPNC bound obtained above.

3 Zermelo-Fraenkel-Axiom-of-Choice (ZFC) and its relevance to majority voting

Axiom of choice in set theory postulates that given a set X (possibly infinite) of non-empty sets, there always exists a choice function that chooses one element from each element-set in X . Though existence of choice function for infinite case is controversial, AOC has been pushed into ZermeloFraenkel

Set theory to make it ZFC set theory (ZF with AOC). Now let us consider a majority voting scenario where the voting population is partitioned into subsets called constituencies and there exists a choice function that chooses one representative element from each constituency. Thus set of representatives from each constituency chosen by choice function form an electoral collegium denoted as EC. This EC can be modelled as a majority voting circuit with each chosen representative of EC as an input to the majority circuit. Thus we have a voting system with two levels.

4 Definition of a choice function and circuits for choice function computation

Let m be the size of voting citizens in a constituency and $q \ll m$ be the number of candidates.

Each of the q candidates get votes $v(i)$ for $i=1,2,3,\dots,q$.
Thus $v(1) + v(2) + v(3) + \dots + v(q) = m$

Above is a partition of voting population. The choice function C is defined as follows:

$C(m,q) = \text{index maxI of the biggest valued part in the integer partition of } m \text{ as defined above , where } 1 < \text{maxI} < q$

By Axiom of Choice, there exists a choice function for choosing a representative from each constituency in infinite case. But it needs to be well-defined that the above choice function always returns a unique index of biggest part in the partition. Unfortunately partition of votes as defined by the choice function can have multiple similar valued biggest parts (which requires a tie-breaker criterion) and might return a set in the absence of a tiebreaker. Thus any voting system similar to above choice function allowing a tie has an inherent limitation. A turing machine computing the choice function for a constituency takes as input the voting population and candidates, initiates a voting on candidates, computes a partition of the voting population into votes for individual candidates, sorts the parts in the partition and outputs the index of the candidate with largest vote (assuming a tiebreaker). Aggregating votes for candidates involves addition NC1 circuits for each of the candidates where voters cast their vote by giving inputs into the NC1 circuit of the corresponding candidate. Outputs of each of the addition circuits form a partition of the voting population. These outputs are parts in the partition which can be sorted using a sorting network to get the largest indexed candidate part. For voting with bounded error these addition NC1 circuits are BPNC circuits with error and feed to a sorting network of polynomial size and logdepth (AKS sorting circuits). Thus the choice function oracle circuit is in BPNC of polylog depth and polynomial size.

5 Circuit for majority voting with constituencies as described above

The non-uniform NC1 majority voting circuit can be modified to query a choice function oracle described above and get the inputs(representatives) for majority gate collegium. Using BPNC choice function described above as oracle for electoral collegium, we obtain a $BPNC^{BPNC}$ circuit. It is not known if BPNC is low for itself like BPP. Thus a BPNC circuit with oracle access to a

BPNC choice function has been arrived at for the special case of majority voting with constituencies and electoral collegium.

6 Acknowledgements

I dedicate this article to God.

7 Bibliography

References

- [1] On Type-2 Probabilistic Quantifiers, Automata, Languages and Programming: 23rd international colloquium, ICALP
- [2] Pseudorandom generators with linear stretch in NC0, Berman Applebaum, Yuval Ishai, Eyal Kushilevitz, Computer science department, Technion, Israel.
- [3] Various resources on Choice functions and Zermelo-Fraenkel-Axiom-of-Choice set theory
- [4] TAC 2010 → Update summarization through interview algorithm - <http://www.nist.gov/tac/publications/2010/participant.papers/CMITproceedings.pdf>
- [5] Various resources on Integer Partitioning
- [6] Parity not in AC0 - Furst, Saxe and Sipser
- [7] Hastad's Switching Lemma, John Hastad's PhD thesis, MIT
- [8] Majority is in non-uniform NC1, Mix Barrington
- [9] Sorting networks of Ajtai-Komlos-Szemeredi
- [10] Lower Bounds for Majority Voting and Pseudorandom Choice (<https://sites.google.com/site/kuja27/LowerBoundsForMajorityVotingPseudorandomChoice.pdf?attredirects=0>)
- [11] Circuit For Computing Error Probability of Majority Voting and Pseudorandom Choice (<https://sites.google.com/site/kuja27/CircuitForComputingErrorProbabilityOfMajorityVoting.pdf?attredirects=0>)
- [12] Interview Algorithm is in IP=PSPACE (<https://sites.google.com/site/kuja27/InterviewAlgorithmInPSPACE.pdf?attredirects=0>)

Integer partitions and their mapping to hash functions - Updated Draft

SrinivasanKannan(alias)Ka.Shrinivaasan(alias)ShrinivasKannan

IndependentOpenSourceDeveloper, Researcher and Consultant

Ph : 9789346927, 9003082186, 9791165980

KrishnaiResearchOpenSourceProducts : http://sourceforge.net/users/ka_shrinivaasan,

https://www.ohloh.net/accounts/ka_shrinivaasan

ResearchWebsite : https://sites.google.com/site/kuja27/

(ka.shrinivaasan@gmail.com, shrinivas.kannan@gmail.com, kashrinivaasan@live.com)

April 17, 2014

Abstract

This article is a short observation of an interesting relation between Integer Partitions and Hash Functions and derives a number of possible hash functions based on this relation.

1 Introduction

A widely used notion of hash function maps a key to value as $h(x) = y$. If x_1 and x_2 are two key values and if $h(x_1)$ and $h(x_2)$ are equal then x_1 and x_2 are placed in same bucket. Thus a hash table partitions the set of keys to be hashed into sets of buckets of keys having same hash value.

2 Generating functions to represent Integer Partitions and Euler's theorem

First we study how integer partitions are represented and later their mapping to buckets of hash functions. One way is through generating function and applying Euler's theorem. Consider the product:

$$(1 + x + x^2 + x^3 \dots)(1 + x^2 + x^4 + x^6 \dots)(1 + x^3 + x^6 \dots)(1 + x^4 + x^8 \dots) \dots \quad (1)$$

To illustrate, consider the coefficient of x^3 . By choosing x from the first parenthesis, x^2 from the second, and 1 from the remaining parentheses, we obtain a contribution of 1 to the coefficient of x^3 . Let the monomial chosen from the i -th parenthesis $1 + x^i + x^{2i} + x^{3i}$ in (1) represent the number of times the part i appears in the partition. In particular, if we choose the monomial x^{i*c_i} from the i -th parenthesis, then the value i will appear c_i times in the partition. Each selection of monomials makes one contribution to the coefficient of x^n and in general, each contribution must be of the form $x^{1*c_1}x^{2*c_2}x^{3*c_3} \dots = x^{1*c_1+2*c_2+3*c_3\dots}$. Thus the coefficient of x^n is the number of ways of writing $n = c_1 + 2 * c_2 + 3 * c_3 + \dots$ where each $c_i \geq 0$. Notice that this is just another way to represent an integer partition. As an example $5 = 1 + 2 + 2$ can be written as $5 = 1 * 1 + 2 * 2$. Above generating function is an infinite product of geometric series $p(n)$ which is the Euler's partition theorem.

3 Number of possible hash functions

Having arrived at a way to express the integer partitions and parts in a partition, we analyse how integer partitions and hash functions are related. Each hash table partitions the hashed elements into sets of buckets. We can map each of these buckets to a part in an integer partition. Thus if there are x parts in a partition of n elements then there will be x non-empty buckets in the hash table where size of each bucket is equal to the value of the corresponding part in the partition and is thus a one-to-one and onto mapping. If there are m possible hash values then each of these x parts or buckets can be arranged in mP_x ways for each partition of n elements (permutations instead of combinations for order of elements). If we aggregate it over all the partitions we get all possible ways of placing an element in a bucket which is nothing but all possible hash functions.

1. Let m be the number of possible values of hash function $h(x)$
2. Let n be the total number of elements which will be hashed and placed in buckets
3. Each hash entry would have a linked list of elements hashed on to a hash value for that entry
4. Let $\lambda(i)$ be the number of parts in partition i
5. Let $p(n)$ be the partition function $[\lambda(i) \leq m \text{ and } m \geq n]$
6. Then number of possible hash functions =

$$\sum_{i=1}^{p(n)} mP_{\lambda(i)} \quad (2)$$

where $\lambda(i)$ which is the number of parts in partition i can be obtained from above generating function for integer partitions as sum of all c_i 's,

$$\sum_{i=1}^q c_i \quad (3)$$

where the value i will appear c_i times in the partition and q is total number of distinct integer in the partition

4 Restricted Partitions, Compositions and Hash Collision Chaining

1. Riemann sums (discrete approximation of Riemann integral) of all the functions corresponding to the hash functions are same. Thus all such functions form an equivalence class. (Assuming each partition created by the hash functions as a function plot)
2. Hardy-Ramanujan asymptotic bound for partition function $p(n)$ is $O(e^{pi*sqrt(0.66*n)}/(4 * 1.732*n))$ which places a bound on number of hash functions also: ([http://en.wikipedia.org/wiki/Partition_\(number_theory\)](http://en.wikipedia.org/wiki/Partition_(number_theory)))
3. If m -sized subsets of the above $O(m! * e^{sqrt(n)}/n)$ number of hash functions are considered as a (k, u) -universal or (k, u) -independent family of functions - $(Pr(f(x1) = y1...)) < u/m^k$, then following the notation above, this m -sized subset family of hash functions follow the $Pr(f(x1) = y1...) < u/m^n$ where n is number of keys and m is the number of values. ($m!$ is for summation over $(m, \lambda(i))$ for all partitions)

4. Thus deriving a bound for number of possible hash functions in terms of number of keys and values could have bearing on almost all hashes including MD5 and SHA.
5. Birthday problem and Balls and Bins problem - Since randomly populating m bins with n balls and probability of people in a congregation to have same birthday are a variant of Integer partitioning and thus hash table bucket chaining, bounds for birthday problem and Chernoff bounds derived for balls and bins could be used for Hash tables also: (http://en.wikipedia.org/wiki/Birthday_problem, <http://www.cs.ubc.ca/nickhar/W12/Lecture3Notes.pdf>)
6. Restricted partitions which is the special case of integer partitions has some problems which are NP-complete. Money changing problem which is finding number of ways of partitioning a given amount of money with fixed denominations(Frobenius number) is NP-complete (<http://citeseer.ub.edu:8080/citeseerx/showciting;jsessionid=92CBF53F1D9823C47F64AAC119D30FC3509754>, *Naoki Abe* 1987). Number of partitions with distinct and non-repeating parts follow Roger-Ramanujan identities (2 kinds of generating functions).
7. The special of case of majority voting which involves integer partitions described in: <https://sites.google.com/site/kuja27/IndepthAnalysisOfVariantOfMajorityVotingwithZFAOC2014.pdf> requires a hash function that non-uniformly distributes the keys into hashes so that no two chains are equal in size (to simulate voting patterns without ties between candidates). This is the special case of restricted partitions with distinct and non-repeating parts of which money changing is the special case and finding a single solution is itself NP-complete.
8. Thus Majority voting can be shown to be NP-complete in 2 ways: a) by Democracy circuit (Majority with SAT) in

$$\begin{aligned} & \text{http://sourceforge.net/projects/acadpdrafts/files/} \\ & \text{ImplicationGraphsPGoodEquationAndPNotEqualToNPQuestion_excerpts.pdf/download} \text{ and} \\ & \text{http://sites.google.com/site/kuja27/} \\ & \text{PhilosophicalAnalysisOfDemocracyCircuitAndPRGChoice}_{2014-03-26}.pdf \end{aligned}$$
b) by reduction from an NP-hard instance of Restricted Partition problem like Money changing problem (MCP) for Majority voting with constituencies described in <https://sites.google.com/site/kuja27/IndepthAnalysisOfVariantOfMajorityVotingwithZFAOC2014.pdf>
9. Infact the above two ways occur in two places in the process of democratic voting: The democracy circuit is needed when a single candidate is elected while the restricted partition in second point is needed in a multi-partisan voting where multiple candidates are voted for.
10. Point 8b above requires a restricted partition with distinct non-repeating parts. There are many results on this like Roger-Ramanujan identities, Glaisher theorem and its special case Euler's theorem which equate number of partitions with parts divisible by a constant and distinctiveness of the parts (odd, differing by some constant etc.,). Such a restricted partition is needed for a tiebreaker and hence correspond bijectively to hash collision chaining.
11. An interesting manifestation of point 10 is that nothing in real-life voting precludes a tie and enforces a restricted partition, with no two candidates getting equal votes, where all voters take decisions independent of one another(voter independence is questionable to some extent if swayed by phenomena like "votebank", "herd mentality" etc.,) thereby theoretically invalidating the whole electoral process.
12. Counting Number of such restricted partitions is a *SharpP-complete* problem - <https://www.math.ucdavis.edu/deloera/TALKS/denumerant.pdf>

13. If a Hash table is recursive i.e the chains themselves are hashtables and so on... then this bijectively corresponds to a recurrence relation for partition function (expressing a partition of a higher integer in terms of lower integer).
14. If the hash table chains are alternatively viewed as Compositions of an integer (ordered partitions) then there are 2^{n-1} maximum possible compositions.
 $(\text{http://en.wikipedia.org/wiki/Composition_(numbertheory)})$
15. In the summation over all parts of partitions derived in
 $\text{https://sites.google.com/site/kuja27/IntegerPartitionAndHashFunctions.pdf}$
if $m == n$ then it is the composition in point 14 above and thus summation over all parts of partitions is greater than or equal to 2^{n-1} as some permutations might get repeated across partitions. Thus the summation expresses generalized restricted composition :
- $$\sum_{i=1}^{p(n)} n P_{\lambda(i)} >= 2^{n-1} (\text{where } m = n) \quad (4)$$
16. For large n the above summation asymptotically equals $e * n!$. Logarithm of above summation then is greater than or equal to $(n - 1)$ and thus can be equated to any partition of n. Thus any partition can be written as a series which is the combinatorial function of parts in all individual partitions.
17. As a special case when all the collision chains are of equal size or have greatest common divisor, a bijection to factors of n and hash function collision chains is obtained. Number of such Hash functions is a function of number of non-trivial factors of n.
18. Generating function for number of distinct partitions is given by $\prod_{k=1}^{\infty} (1+x^k) = \sum_{n=1}^{\infty} q(n) * x^n$
19. Thus number of valid multipartisan majority voting patterns = number of distinct partitions of n($q(n)$) = number of hash functions with distinct collision hash chain sizes (without permutations)
20. If the denominations are fixed as $1, 2, 3, 4, 5, \dots, n$ then the denumerants to be found are from the diophantine equation: $a_1 * 1 + a_2 * 2 + a_3 * 3 + a_4 * 4 + a_5 * 5 + \dots + a_n * n$ (with $a_i = 0$ or 1). GCD of all $a_i(s)$ is 1. Thus Schur's theorem for MCP or Coin Problem applies. Finding one such distinct partition and hence a majority voting pattern without tie is NP-complete by this reduction. This can also be proved to be NP-complete considering the above diophantine as a 0-1 Integer Linear Programming(ILP) NP-complete instance. This partition is guaranteed to have a biggest part always and resolves tie if any.
21. Maximizing Collision in hashes is necessary in finding similarity (e.g DNA, voice and data) and is hence a *pattern matching algorithm* whereas minimizing collision results in a special case of *one way function with collision resistance*.
22. Restricted partitions can be classified into two sets as follows when mapped to hash function collision chains:
- Set of partitions in which no part exceeds a constant size (more Collision Free)
 - Set of partitions in which no part is less than a constant size (more Collision Prone)

23. Recurrence relation for restricted partitions in which no part exceeds N and each partition is of size M is given by: $p(N, M; n) = p(N, M - 1; n) + p(N - 1, M; n - M)$
24. Above recurrence has to be subtracted from partition function to get restricted partitions in which no part is less than N
25. Above points on Schur's theorem are also described in handwritten notes at:
 $https://sites.google.com/site/kuja27/SchurTheoremMCPAndDistinctPartitions_2014-04-17.pdf$

5 Acknowledgement

I dedicate this article to God.

6 Bibliography

References

- [1] Lectures on Integer Partitions by Herbert Wilf, University of Pennsylvania
- [2] This idea was first mentioned by the author in some informal internal email communications at Sun Microsystems when the author worked at Sun Microsystems from 2000-2005(mailid: kannan.srinivasan@sun.com)
- [3] Various lecture notes on Generating Functions(Combinatorics)
- [4] Various website resources for Restricted Partitions
- [5] $http://sourceforge.net/p/asfer/code/HEAD/tree/AstroInferDesign.txt$
- [6] Majority voting related drafts and handwritten notes in $http://sites.google.com/site/kuja27$
- [7] Generating function - $http://math.berkeley.edu/~mhaman/math172/spring10/partitions.pdf$
- [8] Schur's theorem for asymptotic bound for number of denumerants - $http://en.wikipedia.org/wiki/Schur's_theorem$
- [9] Frobenius problem - $http://www.math.univ-montp2.fr/ramirez/Tenerif3.pdf$
- [10] Hash collision chains and Schur Theorem - $https://sites.google.com/site/kuja27/SchurTheoremMCPAndDistinctPartitions_2014-04-17.pdf$
- [11] Standard NP-Completeness reductions - $http://www.cs.cmu.edu/afs/cs/academic/class/15451-s10/www/recitations/rec0408.txt$

Integer partitions and their mapping to hash functions

Ka.Shrinivaasan (ka.shrinivaasan@gmail.com)

June 17, 2012

Abstract

This article is a short observation of an interesting relation between Integer Partitions and Hash Functions and derives a number of possible hash functions based on this relation.

1 Introduction

A widely used notion of hash function maps a key to value as $h(x) = y$. If x_1 and x_2 are two key values and if $h(x_1)$ and $h(x_2)$ are equal then x_1 and x_2 are placed in same bucket. Thus a hash table partitions the set of keys to be hashed into sets of buckets of keys having same hash value.

2 Generating functions to represent Integer Partitions and Euler's theorem

First we study how integer partitions are represented and later their mapping to buckets of hash functions . One way is through generating function and applying Euler's theorem. Consider the product:

$$(1 + x + x^2 + x^3 \dots)(1 + x^2 + x^4 + x^6 \dots)(1 + x^3 + x^6 \dots)(1 + x^4 + x^8 \dots) \dots \quad (1)$$

To illustrate, consider the coefficient of x^3 . By choosing x from the first parenthesis, x^2 from the second, and 1 from the remaining parentheses, we obtain a contribution of 1 to the coefficient of x^3 . Let the monomial chosen from the i -th parenthesis $1 + x^i + x^{2i} + x^{3i}$ in (1) represent the number of times the part i appears in the partition. In particular, if we choose the monomial x^{i*c_i} from the i -th parenthesis, then the value i will appear c_i times in the partition. Each selection of monomials makes one contribution to the coefficient of x^n and in general, each contribution must be of the form $x^{1*c_1}x^{2*c_2}x^{3*c_3} \dots = x^{1*c_1+2*c_2+3*c_3} \dots$. Thus the coefficient of x^n is the number of ways of writing $n = c_1 + 2 * c_2 + 3 * c_3 + \dots$ where each $c_i \geq 0$. Notice that this is just another way to represent an integer partition. As an example $5 = 1 + 2 + 2$ can be written as $5 = 1 * 1 + 2 * 2$. Above generating function is an infinite product of geometric series $p(n)$ which is the Euler's partition theorem.

3 Number of possible hash functions

Having arrived at a way to express the integer partitions and parts in a partition, we analyse how integer partitions and hash functions are related. Each hash table partitions the hashed elements into sets of buckets. We can map each of these buckets to a part in an integer partition. Thus if there are x parts in a partition of n elements then there will be x non-empty buckets in the hash

table where size of each bucket is equal to the value of the corresponding part in the partition and is thus a one-to-one and onto mapping. If there are m possible hash values then each of these x parts or buckets can be arranged in mC_x ways for each partition of n elements. If we aggregate it over all the partitions we get all possible ways of placing an element in a bucket which is nothing but all possible hash functions.

1. Let m be the number of possible values of hash function $h(x)$
2. Let n be the total number of elements which will be hashed and placed in buckets
3. Each hash entry would have a linked list of elements hashed on to a hash value for that entry
4. Let $\lambda(i)$ be the number of parts in partition i
5. Let $p(n)$ be the partition function [$\lambda(i) \leq m$ and $m \geq n$]
6. Then number of possible hash functions =

$$\sum_{i=1}^{p(n)} mC_{\lambda(i)} \quad (2)$$

where $\lambda(i)$ which is the number of parts in partition i can be obtained from above generating function for integer partitions as sum of all c_i 's,

$$\sum_{i=1}^q c_i \quad (3)$$

where the value i will appear c_i times in the partition and q is total number of distinct integer in the partition

4 Acknowledgement

I dedicate this article to God.

5 Bibliography

References

- [1] Lectures on Integer Partitions by Herbert Wilf, University of Pennsylvania
- [2] This idea was first mentioned by the author in some informal internal email communications at Sun Microsystems when the author worked at Sun Microsystems from 2000-2005(mailid: kannan.srinivasan@sun.com)
- [3] Various lecture notes on Generating Functions(Combinatorics)

Interview Algorithm is in IP=PSPACE

Ka.Shrinivaasan (ka.shrinivaasan@gmail.com)

July 9, 2012

Abstract

This article is a short follow-up to the Interview Algorithm published earlier. An attempt is made to perceive interview with a complexity theory spectacle. Interview is mapped to Arthur-Merlin class of languages and its generalization as interactive proof class (IP) through a reduction. Moreover Interview algorithm is executed on a probabilistic turing machine allowing a bounded error probability.

1 Introduction

Article 1) defined a concept of interview and presented an algorithm for interview of one document by other document through question-answering using some automatically obtained questions or through some predefined reference questions. This extends 1) by analyzing some interesting complexity theoretic properties of an interview.

1.1 Motivation for Interview algorithm (reproduced here for continuity)

Here we map the real world scenario of an interview being conducted on a candidate where a panel asks questions and judges the candidate based on the quality of answers by candidate - candidate is a document and it is "interviewed" by a reference set of authorities. Each document x is interviewed/evaluated by set of reference documents which will decide on the merit of the document x . Reference set initially consists of n user chosen authorities on the subject. Interview is set of queries made by reference set on the document and evaluating the answer to the queries. If x passes the interview it is inducted into reference set. Next document will be interviewed by $n+1$ documents including last selected document and so on. Hierarchy of interviews can be built. For example Document x interviews documents y and z . Document y interviews w and document z interviews p . Thus we get a tree of interviews (it could be a directed acyclic graph too, if a candidate is interviewed by more than one reference, one of which itself was a candidate earlier). The interview scores can be weighted and summed bottom-up to get the merit of the root (Analogy: hierarchy in an organization).

1.2 Steps of the Interview algorithm

1. Relevance of the document to the reference set is measured by a classifier (NaiveBayesian or SVM or search engine results for a query)
2. Intrinsic merit score of the document is computed either by Recursive Gloss Overlap algorithm (measures the meaningfulness/sanity of the candidate) (or) by citation digraph

3. Reference set interviews the candidate and gets the score
4. Value addition of the candidate document is measured (what extra value candidate brings over and above reference set)
5. Candidate is inducted into reference set based on the above criteria if candidate is above a threshold.

2 Interview Algorithm and Interactive Proofs

In complexity theory, Arthur-Merlin or AM is the class of languages with an interactive proof that consists of the verifier sending a random string, the prover responding with a message, and where the decision to accept is obtained by applying a deterministic polynomial-time function to the transcript. The class Merlin-Arthur or MA denotes the class of languages with 2-round public coins (verifier reveals the random string to prover) interactive proof with the prover sending the first message. Interactive Proof or IP class introduces polynomial rounds of Prover-Verifier transcripts. Both Public-coins and Private-coins are being used depending on whether verifier reveals everything to prover or not with not too much of difference in computing power. By intuition there is a notable similarity between Interview Algorithm above which operates on question-answering and Interactive Proofs.

3 Interview algorithm is in $IP=PSAPCE$ - proof by reduction

Formalizing the above intuitive similarity, we can map a prover to the interviewee document and verifier to the interviewer document. If interview algorithm in 1) is relaxed to allow a probabilistic turing machine executing the interview, then by soundness, verifier(interviewer) accepts the prover's(interviewee) right answer by probability of atleast $2/3$ and by completeness, verifier(interviewer) accepts the prover's(interviewee) wrong answer by probability of not more than $1/3$. Thus interview algorithm gets less stringent. We assume that Interview algorithm has polynomial rounds. The interview transcript is evaluated by a polytime function which decides on accepting or rejecting the candidate document. Thus we have obtained a reduction from Interactive Proof to Interview algorithm which is executed by a probabilistic turing machine. By this reduction Interview algorithm has been shown to be in $IP=PSAPCE$. This can also be proved to be PSPACE-hard by turing reduction from a PSPACE-complete problem. For example an interview can be represented by a totally quantified boolean formula (TQBF) written as

$$\forall question_1 \ \exists answer_1 \ \forall question_2 \ \exists answer_2 \dots \dots \quad (1)$$

and the formula is satisfied based on correctness of answers to the questions

4 Acknowledgement

I dedicate this article to God.

5 Bibliography

References

- [1] Few Algorithms for ascertaining merit of a document - <http://arxiv.org/pdf/1006.4458.pdf>

- [2] TAC 2010 Update summarization by Interview Algorithm (<http://www.nist.gov/tac/publications/2010/appendices/Summarization/guided/CMIIT.pdf>)
- [3] Complexity theory by Sanjeev Arora and Boaz Barak

Lower Bounds for Majority Voting and Pseudorandom choice

Ka.Shrinivaasan (ka.shrinivaasan@gmail.com)

March 8, 2013

Abstract

In this article philosophical ramifications of the $P(\text{good})$ equation are described and circuits for the LHS and RHS of the $P(\text{good})$ equation are modelled and their lower bounds are analyzed.

1 Some circuit models for majority voting

In a majority voting milieu, each voter has a criterion to judge a candidate based on which the voter decides to vote for or against. Since satisfying the criterion is akin to a SAT problem, a voter's criterion can be specified as a CNF where each clause of the CNF is a sub-criterion rule that is part of the criterion the voter has and every rule clause is dependent on some variables. The probability that the voter makes a good decision can be defined with conditional probability as follows,

$\Pr(\text{Voter makes good decision}) =$

$\Pr(\text{Criterion CNF is good}) *$

$\Pr(\text{CNF is populated with correct inputs/ Criterion CNF is good})$

$\Pr[\text{Criterion CNF is good}] = p = c_1 * c_2 * c_3 * \dots * c_k$

where each c_i gives the probability that i (th) clause is good.

$\Pr[\text{CNF is populated with correct inputs}] = q = q_1 * q_2 * q_3 * \dots * q_l$

where each q_i gives the probability that i (th) variable is correctly populated.

From the above, $\Pr(\text{Voter makes a good decision}) = p * q$

2 Voter as an Oracle Turing Machine

Voter with CNFSAT having bounded error can be modelled as a Polytime Turing machine with bounded error. The Voter Turing machine accepts a CNFSAT criterion and variable assignments to the CNFSAT in its input tape and computes the CNFSAT on those assignments on a worktape and writes the output to an output tape. This computation takes only polynomial time as it is only a verification step. But this computation can have an error as described previously as the CNFSAT might output 1 when it has to output 0 and viceversa. Such a Voter Turing machine with error can be found in real life applications. Since it is in polynomial time with bounded error Voter Turing machine is in BPP. This BPP Voter turing machine can be used as an Oracle to other Turing machines and circuits. The CNFSAT criterion can be probabilistic also and thus it could be in Probabilistic SAT(PSAT) - Probabilistic SAT has to satisfy probability distribution constraints

on the clauses of the PSAT. But PSAT does not involve bounded error of a voter. Thus BPP model which allows bounded error of a voter is a better oracle. Moreover the bounded error in BPP can be brought down (derandomized in a sense) to a required threshold by repeated runs.

3 Error probability derivation for majority voting

For even number of voters($2n$),

$\Pr(\text{majority decision is good}) =$

$$(2n)!/(4^n) [1/(n+1)!(n-1)! + 1/(n-2)!(n+2)! + \dots + 1/(2n)!]$$

For odd number of voters(m)

$\Pr(\text{majority decision is good}) =$

$$m!/(2^m) [1/x!(m-x)! + \dots + 1/m!]$$

where x is the ceiling($m/2$)

Assumption here is that a good majority decision with high probability also results in a good outcome in the majority voting (rather it is trivial and is stated without proof).

4 Ways to simulate majority voting

Majority can be computed by non-uniform NC1 circuit or through a sorting network of polynomial size (Ajtai et al) or by Valiant's non-constructive majority circuit. As described previously, each voter has a criterion while voting which can be modelled as a CNF SAT with error probability. Thus a majority gate computing majority can be given access to oracle BPP Voter Turing machine as described in previous section. So such a majority voting circuit is in the non-uniform circuit complexity class $NC1^{BPP}$. NC1 is in P which inturn is in BPP. Thus majority voting with error by voter nodes can be computed in $NC1^{BPP}$. The error for this circuit is defined by the RHS of the P(good) equation. Alternatively, instead of having a BPP oracle access, the majority voting circuit can also be designed as a non-uniform Bounded Error NC1 circuit placing it in complexity class BPNC.

5 Ways to simulate junta or a random choice

LHS of the P(good) is a junta which is not the result of a majority vote and it arises by itself. This random choice can be one among the voters. Since there is no perfect randomness, only a pseudorandom generator has to be resorted to for choosing a junta. Pseudorandom generators fool a statistical test into believing as though it is truly random with high probability and form the basis of cryptographic key generators. Hypothesis of existence of pseudorandom generators is yet to be proved. Existence of one-way functions (hard to invert) imply pseudorandom generators. There are prominent PRGs like Nisan which run in polynomial time. Thus a junta can be output by following algorithm:

1. Source for a PRG which runs in polytime
2. Query the PRG to get the (pseudo)random choice

Junta output by the above algorithm can be either good or bad with a probability distribution and thus has a bounded error. This algorithm runs in polynomial time but with bounded error described by LHS of $P(\text{good})$ equation and thus in BPP.

6 Consequences of above bounds for pseudorandom choice and majority voting

As described above pseudorandom choice is in BPP and majority voting with error is in non-uniform NC1 with oracle access to BPP. Both a pseudorandom choice and majority voted choice have the same error probability of being correct as derived by $P(\text{good})$ equation. Though this is against conventional wisdom, the $P(\text{good})$ equation is not necessarily a correct representation of realworld majority voting where individual voters can have differing judgemental abilities with unequal error probability which is a result of varying human intelligence amongst the population and hence is a mootpoint. Thus LHS of the $P(\text{good})$ equation may be more or less compared to the RHS and as a result either a pseudorandom choice junta or a majority voting may be favorable depending on individual judging error probabilities in the population. Thus in real world this error probability derivation may have no relevance at all. But in a theoretical world, a distributed computing system where all nodes are equally intelligent(or have same algorithm running in them), then LHS of $P(\text{good})$ is always equal to RHS since there is zero bias because of uniform distribution. Application of the above separation result for future complexity class separations could be interesting.

7 Acknowledgement

I dedicate this article to God.

8 Bibliography

References

- [1] Few Algorithms for ascertaining merit of a document - <http://arxiv.org/pdf/1006.4458.pdf>
- [2] TAC 2010 Update summarization by Interview Algorithm (http://www.nist.gov/tac/publications/2010/appendices/Summarization/guided/CMI_IT.pdf)
- [3] Interview algorithm is in IP=PSpace
- [4] Nisan PRG
- [5] Complexity - Arora Barak
- [6] Leader Election Algorithms in distributed systems
- [7] Majority in non-uniform NC1 - Barrington

Few algorithms for ascertaining merit of documents and their applications

Ka.Shrinivaasan, Chennai Mathematical Institute (CMI)
(shrinivas@cmi.ac.in)

advised by

Dr.B.Ravindran, IIT Madras and Dr.Madhavan Mukund, CMI

(submitted for the requirement of thesis for M.Sc(Computer Science))

April 30, 2010

Contents

1	Introduction	4
2	Motivation	4
3	Three algorithms presented hereunder	5
4	Directed Graph of Citations	5
4.1	Average Maxflow and Path lengths of Directed Graph of Citations	5
4.2	Polarity of citation edge	6
5	Definition Graph Convergence(or)Generalized recursive gloss overlap	6
5.1	Motivation for computing Intrinsic Merit of a document	6
5.2	Definition tree of a document	6
5.3	Definition graph convergence and steps of Recursive Gloss Overlap algorithm	8
5.4	Definition of shrink	9
5.5	Comparison of two documents for relative merit - two examples .	9
5.6	Intrinsic merit score, Convergence factor and Relatedness	10
5.7	Intuition captured by above intrinsic merit score	12
5.8	Breadth/Depth first search of definition graph and why it is not a good choice for computing merit score	12
5.9	Sentiment analysis applying Recursive gloss overlap	12
5.10	False negatives	13
5.11	False positives	13
5.12	Normalization	13
5.13	Ordering and Relative Merit	13
5.14	Semantic relatedness or Meaningfulness of a document	14
5.15	Formal proof of correctness of Convergence and Intrinsic Merit Score	14
5.16	Parallelizability	15
6	Interview Algorithm (applying (1) and/or (2) for computing intrinsic merit)	15
6.1	Motivation for Interview algorithm	15
6.2	Steps of the Interview algorithm	15
6.3	Mathematical formulation of an interview	16
6.4	Searching for answer to a query within the document (as implemented)	17
6.5	Value addition measure	17
6.6	Update summarization through Interview algorithm (applying algorithm given in 6.2)	17
6.7	Application to Topic Detection, Link detection and Tracking . .	18
6.8	Implementation in Python	18

7 Results	19
7.1 Results - Intrinsic Merit score with quadratic overlap for top 10 Google ranked documents for query 'data mining' sorted ascending	19
7.2 Results - Intrinsic Merit score with quadratic overlap for top 10 Google ranked documents for query 'philosophy' sorted ascending	20
7.3 Results - Intrinsic Merit score with quadratic overlap for human (2 judges) judged documents on topic 'democracy' - sorted ascending	21
7.4 Results - Intrinsic Merit score with quadratic overlap for human (1 judge) judged documents on topic 'soap' - sorted ascending	21
7.5 Results - Intrinsic Merit score with quadratic overlap for top 7 Google news stories for query 'haiti earthquake' - sorted ascending	21
7.6 Results - Intrinsic Merit score with quadratic overlap for top 10 Google ranked documents for query 'literary' sorted ascending	22
7.7 Results Excerpt- Intrinsic Merit Ranking of Reuters corpus in 'earn' category	23
7.8 Results - Spearman ranking coefficient and Pearson coefficient for the above rankings	23
7.9 Results - Update Summarization applying Interview algorithm with Recursive Gloss Overlap - Example - Summary size is 12.5%	24
7.10 Results Excerpt - Topic Detection and Tracking applying Interview algorithm with Recursive Gloss Overlap	26
7.11 Results Excerpt - Applying Sentiment Analysis with Recursive Gloss Overlap for finding polarity of an edge in Citation Graph Maxflow (1)	30
7.12 Results Excerpt - Citation graph maxflow with a simple link graph example - polarity determined by Recursive Gloss Overlap	30
7.13 Conclusion	31

1 Introduction

Existing models for ranking documents(mostly in world wide web) are prestige based. In this thesis, alternative schemes to objectively judge the merit of a document independent of any external factors (like link graph) are proposed

2 Motivation

Motivation for objective, independent judgement of a document is founded on the following example:

Judge X decides about the merit of an entity Z purely by what other entities opine about Z without interacting with Z; Judge Y decides about the merit of Z by interacting only with Z. Question now is who is better judge - X or Y.

Probability of judgmental error of judge X is equal to probability of collective error of entities opining about Z while probability of judgemental error of judge Y is 0.5 as the following elementary arithmetic shows. Let us assume there are $2n$ voters and they need to decide/vote on whether a candidate is good or bad. A candidate getting majority ($n + 1$ good votes) will be winner.

Question: What is the probability that people have made a good decision?

Answer: Probability of each voter making a good decision is p and bad decision is $1 - p$ ($0 \leq p \leq 1$). Let $p = 0.5$ for an unbiased voter.

So for a candidate to be judged 'good', atleast $n + 1$ people should have made a good decision. Probability of a good choice for these $2n$ voters, skipping the calculations, is :

$$\begin{aligned} P(\text{good}) = & ((2n)!/4^n) * ((1/((n+1)!(n-1)!)) + \\ & 1/((n+2)!(n-2)!)) \\ & + \dots + 1/((n+n)!(n-n)!)) \quad (1) \end{aligned}$$

If there is an objective judgement without voting, probability of good decision is 0.5. It is interesting to see that above series tends to 0.5 as n grows infinitely. Thus, the judgement-through-majority-vote error probability is equal to the error probability of judge X who uses only the inputs from witnesses to judge Z while judgement-through-interaction(without election) error probability is equal to the error probability of judge Y (i.e. 0.5) who does not use witnesses. Thus, both judges X and Y are equally fallible but the cost incurred in a real world scenario for simulating X far outweighs that of Y. Thus it is worth delving into schemes for objective judgement like Y.

3 Three algorithms presented hereunder

1. Maxflow and Path lengths of Citation graphs - objective judgement (differs from Pagerank since it is Maxflow based and not prestige based)
2. Generalized Recursive Gloss Overlap - objective judgement (simulates judge Y with a 'white-box', invasive, intrinsic merit scoring) - covers majority of this report
3. Interview algorithm - objective judgement (simulates judge Y; Uses questions and answers to judge a candidate - 'black-box' and less-invasive - and also incorporates intrinsic merit score obtained from either MaxFlow of Citation graph or Generalized Recursive Gloss Overlap)

4 Directed Graph of Citations

4.1 Average Maxflow and Path lengths of Directed Graph of Citations

Given a corpus, algorithm constructs directed graph of incoming links to a document x from those documents chronologically later than x . Thus corpus is partitioned into set of digraphs. Indegree of a vertex in this digraph reflects the importance of a document represented by a vertex. This digraph can be thought of as a flow network where concept flows from a document to others which cite. Each edge has a weight. Flow/weight for an (u,v) edge is defined as number of references v makes while citing u though there could be other ways to weight an edge. Assigning polarity to this weight is discussed in 4.2. MinCut of the digraph is the set of documents which are "potentially most influenced by the source document" (because maximum flow of concept from source occurs through this set to outside world/sink). Thus size of maxflow/mincut, averaged over all vertex-pairwise maxflow values, is a measure of influence of a source document in a community and thus points to its merit. (E.g., Chronology for web documents can be found by 'Last-modified' HTTP header which every dynamic document server is mandated to send to client). Alternative way to get the merit is to count the number of vertices in a predefined radius from source (i.e set of paths of some fixed length from source) which can be less accurate and sometimes misleading. Thus documents can be ranked using average Maxflow values. Advantage of this scheme is that it quantifies the extent of percolation of a concept within a community through Maxflow, without giving importance to the prestige measure of the vertices(documents) involved. So, this is one way of objectively assessing the merit of a vertex(document). Implementation applies Ford-Fulkerson algorithm to each s, t distinct pair and finds the average maxflow out of each vertex.

4.2 Polarity of citation edge

Parse the document/sentence containing the citation/link into tokens and find polarity. Whether a word is positive or negative can be decided by:

1. looking up a sentiment annotated ontology (e.g positivity/negativity of a lemma in SentiWordNet) or
2. entropy analysis - using $\sum_{i=0}^1 (-P(i)\log P(i))$ where $P(0)$ = percentage of positive words and $P(1)$ = percentage of negative words. Closer the entropy to zero, clearer the sentence/document on its viewpoint (very good or very bad) or
3. recursive gloss overlap algorithm to the citing document to get the polarity/sentiment of context citing the document.

Implementation tries all the three above. If the polarity/sentiment is negative, the weight for edge (u,v) is made negative in citation digraph, indicating a negative flow of concept to vertex v from the cited vertex u .

5 Definition Graph Convergence(or)Generalized recursive gloss overlap

5.1 Motivation for computing Intrinsic Merit of a document

Intrinsic merit is defined as the amount of intellectual effort put forth by the reader of a document and we try to quantize this effort. It is important to note that this quantized effort is independent of any observer/link-graph. Any document goes through some human understanding and we try to model it through what can be called Iceberg/Convergence/Generalized recursive gloss overlap algorithm (named so because a web document contains only a tip of the knowledge a document represents and understanding the document requires deeper recursive understanding of the facts or definitions the document is home to.). For example, going through a research paper requires the understanding of the concepts which draw a logical graph in our mind. Thus time spent on grasping the concepts and hence the intrinsic merit is proportional to the size and complexity of this graph and points to its merit (which is equal to the intellectual effort of the human reader). Since WordNet is the existing model for semantic relationship, we will try to establish that a text document can be mapped to a graph which is a subgraph of WordNet and merit can be derived applying some metrics on this graph. This is the intuition behind the algorithms that follow.

5.2 Definition tree of a document

Given a document its definition tree is recursively defined as

Definition 1. $\text{definitiontree}(\text{all keywords of document}) = \text{definitiontree}(\text{term1}) \text{definitiontree}(\text{term2}) \dots \text{definitiontree}(\text{termn})$ where $\text{term1}, \text{term2}, \dots, \text{termn}$ occur in the definition of keywords of a document.

For example, let us consider the following document which talks about Kuratowski theorem

Document1 = Every K5,K3,3-free graph is planar

This document contains key terms like "K5,K3,3-free", "graph" and "planar". Now we recursively construct the definition tree for these terms. Key terms are decided after filtering out stopwords and by computing TF-IDF and only terms above a threshold tfidf are chosen for constructing the definition graph.

definitions at level 1:

1. K5 = Complete graph of 5 vertices (key terms: graph, vertices)
2. K3,3 = graph of two sets of 3 vertices each interconnected (key terms: graph, two sets, vertices, interconnected)
3. graph = set of vertices and edges among them (key words: vertices, edges, set)
4. planar = graph embedded on a plane (key words: graph, embedded, plane)

Thus the definition tree goes deeper as each keyword/concept is dissected and understood. Given above is level-1 grasping of the document. Important thing to note is that intersection of the sets of keywords in the definition of K5, K3,3, graph and planar is not an empty set (glosses for two or more keywords overlap). For example, intersection of definitions of K5 and K3,3 is the set {graph, vertices}. Thus the overlap of the terms "graph" and "vertices" in two definitions of K5 and K3,3 is an indication of deeper cohesion/interrelatedness of the terms in the document. Thus the replicated terms (represented by vertices) in the definition tree can be merged to get convergence (gloss overlap generalized to more than two glosses). Thus the definition tree is transformed into definition graph (since a vertex can have more than one parent) by merging replicated keyword vertices into 1 vertex. Synset definitions in WordNet gloss are used for getting keyword definitions in the implementation. But WordNet Gloss does not work for terms specialized for a domain (e.g. gloss for "graph" does not have a synset for graph theory as part of its senses set). This requires ontologies for the class the document belongs to. Thus recursive gloss overlap algorithm is limited by WordNet in present implementation. At each level, word sense disambiguation is done by following Lesk's algorithm adapted to Generalized Recursive Gloss overlap to choose the synset definition fitting the context. It is important to note that 1) only one relation ("is in definition of") is used and 2) only keywords within the document are considered 3) gloss overlap is computed recursively at each level of understanding till required depth is reached.

5.3 Definition graph convergence and steps of Recursive Gloss Overlap algorithm

Convergence of a document is defined as the decrease in the number of unique vertices of the set of definition trees of its keywords from level k to level $k+1$. For example definition tree of the above document converges to {edges, vertices} after expanding the definition tree further down. Thus the above document has "edges" and "vertices" as its undercurrent. Thus the Convergence algorithm takes no labelled examples for inference. Only requirement is to have a dictionary/gloss/ontology of terms and their corresponding definitions. If a documents definition tree does not converge within a threshold called "depth" number of levels then the document is most likely less meaningful or of low merit. Thus the Convergence algorithms strikingly adapts an iceberg which has seemingly unconnected set of "tips" at the top but as we go deeper get unified. Level where this unification happens is a differentiator of merit. If while recursively expanding the definition tree, a vertex results in a child vertex which is same as some sibling of the parent then we compute and remove the intersection of keywords at present and previous level - since these common vertices have already been grasped. Accordingly, number of edges, vertices and relatedness are updated for each level. Number of vertices are adjusted for removal of common tokens, but number of edges remain same since they just point to a different vertex at that level. This process continues top-down till required depth is reached.

Steps:

1. Get the document as input
2. currentlevel = 1
3. keywordsatthislevel = {keywords from the document through tfidf filter (e.g > 0.02)}
4. While (currentlevel < depthrequired) {
 - For each keyword from keywordsatthislevel lookup the best matching definition for the keyword and add to a set of tokens in next level - requires WordSenseDisambiguation - implementation uses Lesk's algorithm
 - Remove common tokens with previous levels since they have been grasped in previous level (this is an optimization)
 - Update the number of vertices, edges and relatedness (vertices correspond to unique tokens, edges correspond to the single relation 'y is in definition of x' and relatedness is linear overlap or quadratic overlap) and Update tokensofthislevel
 - currentlevel = currentlevel + 1}

5. Output the Intrinsic merit score =

$$|vertices| \cdot |edges| \cdot |relatedness| / \text{firstconvergencellevel}$$

Where

- Relatedness = $Number\text{OfOverlaps}$ (linear, also called as convergence factor) (or) Relatedness = $Number\text{OfOverlappingParents} * Number\text{OfOverlaps}^2$ (quadratic)
- firstconvergencellevel = level of first gloss overlap

At the end of recursive gloss overlap, nodes with high number of indegrees(parents) are indicators of the class of the document since greater the indegree, greater is the number of keywords overlapping (voting for an underlying theme). From graph theoretic view, Definition Graph constructed above is a multipartite graph since vertices can be partitioned into sets with no edges within a set and edges only across sets (without removal of common tokens between levels - which is only an optimization since by removing common tokens we redirect edges to vertices within the same set and multipartiteness is lost). Preserving multipartiteness is useful since it groups the tokens at each level of recursion into single set with edges across these sets - multipartite cliques of this multipartite graph can be analyzed to get the robustness. Moreover, this algorithm ignores grammatical structure. Reason is that principal differentiator in analyzing relative merit of two documents is the quality of content and complexity of content and both documents are equally grammatical. Quality of content is proportional to the vertices of the definition graph and complexity of the content is proportional to the relatedness and edges of definition graph. In spite of ignoring grammatical structure, the graph constructed above is context-sensitive since word sense disambiguation is done while choosing the synset matching a keyword. This way, the definition graph is a graph representation of the knowledge in the document sans the grammatical connectives.

5.4 Definition of shrink

Definition 2. Let us define "shrink" to be the amount of decrease in the number of unique vertices between levels k and $k + 1$ during convergence (gloss overlap)

5.5 Comparison of two documents for relative merit - two examples

Document1 : Car plies on sky

Constructing definition graph for level-1 we get,

1. Car - automobile used for surface transport

2. plies - is flexible; goes on a surface; moves

3. sky - atmosphere; not on earth;

As can be readily seen there is overlap of 2 key terms at level 1 of the tree and thus there is less gloss overlap. Thus at level-1 document looks less meaningful.

Document2 : Cars and buses ply on road

Constructing definition graph for level-1 we get,

1. Car - automobile used for surface transport
2. Buses - automobile used for surface transport
3. ply - flexible; go on a surface; move
4. road - asphalted surface used for transport

All 4 keywords overlap giving surface as common token in their respective glosses. Overlap is better than Document1, since more keywords contribute to overlap.

5.6 Intrinsic merit score, Convergence factor and Relatedness

Definition 3. Let us define *Intrinsic merit I* to be the product of number of vertices(V), number of edges(E) and *Convergence factor(C)* of the definition graph of the document.

$$I = V * E * C \quad (2)$$

Convergence factor (C) is the difference between number of vertices in definition tree and number of vertices in definition graph (V). Number of vertices in definition tree includes overlapping vertices without coalescing them (since after coalescence we get the definition graph).Number of vertices in the definition tree = $x^d - 1$ where x is the average number of keywords per term definition and d is the depth of the definition tree of the document. Let us add 1 to this to get x^d (smoothing). Number of vertices in the definition graph = V Thus the Convergence factor C and Intrinsic merit I become,

$$C = x^d - V \quad (3)$$

$$I = V * E * (x^d - V) \quad (4)$$

Intrinsic Merit score can also be further fine-tuned by taking into account the level of definition tree at which first convergence(gloss overlap) happens, defined as firstconvergencellevel. Greater the firstconvergencellevel, more irrelevant the document "looks" (but has a deeper cohesion). Depth to which definition tree has to be grown is decided by extent of grasp needed by the reader. Thus greater the depth of definition tree, greater is the understanding.

It is obvious to see that Depth has to be greater than firstconvergencelevel so that some pattern can be mined from the document. Heuristically, we can grow the definition tree till intersection of leaves of all sub-trees of the keywords in the document is non-empty. This is the point where we can safely assume that all keywords in the document have been somehow related to one another. So, Intrinsic merit score can be improved by incorporating firstconvergencelevel denoted by f. Thus improved score is

$$I = V \cdot E \cdot (x^d - V) / f \quad (5)$$

(since merit is inversely proportional to firstconvergencelevel) .Complexity of constructing definition tree is $O(x^d)$. Since non-unique vertices are coalesced(through gloss overlap), definition graph can be constructed in $O(V)$ time (subexponential). Since x is the average number of children keywords per keyword, $x = E/V$. Substituting,

$$I = E * V * (E^d - V^d) / (V^d * f) \quad (6)$$

As an alternative to convergence factor, gloss relatedness score similar to the one discussed by Banerjee-Ted, but considering only one relation, number of overlapping parents and length of overlap can be used to get the interrelatedness/cohesion of the document. Replacing the convergence factor with relatedness, Intrinsic merit becomes, $I = V \cdot E \cdot Rel/f$ where Rel is the sum of relatedness scores, computed over all overlapping glosses at each convergence level and f is the level at which first gloss overlap occurs

$$Rel = \sum_{i=1}^n (relatedness(Level(i), keyword1, keyword2, \dots, keywordn)) \quad (7)$$

This relatedness score has been generalized to overlap of more than two glosses with single relation R ($R(x,y) = y$ is in definition of x). Function relatedness() for n-overlapping keywords is defined as,

$$\begin{aligned} relatedness(Level(i), keyword1, keyword2, \dots, keywordn) = & OverlapLengthAtLevel(i) \\ & (LinearOverlap) \end{aligned} \quad (8)$$

(or)

$$\begin{aligned} relatedness(Level(i), keyword1, keyword2, \dots, keywordn) = & n \cdot (OverlapLengthAtLevel(i)^2) \\ & (QuadraticOverlap) \end{aligned} \quad (9)$$

The relatedness score reflects the convergence since it takes into account the overlapping keywords at each level and length of the overlap. Thus first version of relatedness() function, implies the convergence factor (difference in

number of vertices of definition tree and definition tree, signifying overlap) Intrinsic merit/Relatedness score can be used to rank the set of documents and display them to the user. Referring back to examples in 5.5, quadratic relatedness measure ((9) above) is a better choice than linear overlap since it is a function of both overlapping parents and the overlap length. The quadratic overlap gives greater weightage to length of overlap by squaring it while keeping the number of parents involved linear.

5.7 Intuition captured by above intrinsic merit score

The number of edges (representing relation between parent term and its definitions) increase as relationship among vertices of definition graph increases. The number of vertices(keywords) in the definition graph increases, as the knowledge represented by the document increases. The depth of the definition tree increases, as the understanding grows. Convergence factor increases as number of overlapping terms in definition graph increases. Similarly quadratic relatedness score increases with number of keywords involved in overlap and the length of overlap, thus pointing to stronger semantic relationship among the keywords. Intuitively, definition graph is WordNet(or any other ontology) projected onto the document.

5.8 Breadth/Depth first search of definition graph and why it is not a good choice for computing merit score

Since Breadth/Depth first search of graph can model human process of thinking, BFS/DFS algorithms can be applied to get the merit score. Since BFS/DFS algorithms run in $O(V + E)$ time merit score is proportional to $V + E$ - all vertices of the graph are visited in $O(V + E)$ time. But the drawback of this approach is that strength of underlying theme of the document and cohesion of keywords is not captured by this merit score. Since Intrinsic merit score obtained by Convergence reckons with depth and overlapping keywords, BFS/DFS merit score is discarded

5.9 Sentiment analysis applying Recursive gloss overlap

Recursive Gloss Overlap algorithm after few levels down the definition tree would spell out the sentiment of writer.

Example1: "That movie was fantastic; Graphics was awesome"
Keywords at level-1 of Definition graph construction:

1. movie - motion picture; positive
2. fantastic - good, excellent; positive
3. graphics - software technique; positive
4. awesome - good, great; positive

Overlapping terms are {good, positive} and large number of keywords(parents) contribute to this overlap. Thus the document is of extolling nature about some target entity. Prerequisite is a dictionary which annotates each word with the sentiment and sense of the word(Implementation uses SentiWordNet which gives positivity/negativity for each lemma).

Sentiment analysis with Recursive Gloss Overlap is applied to finding the polarity of an edge in Citation graph (See (1)). Recursive Gloss Overlap algorithm is applied to each Citation context and a definition graph is constructed. Keyword vertices with more than one indegree are then tested for positivity and negativity using SentiWordNet. If majority of these is positive then polarity for citation edge is positive, otherwise negative.

5.10 False negatives

Convergence algorithm never assigns lower merit score to a document which deserves a higher merit since a document with higher merit explains the concept with more depth/cohesion than document with lower merit. So false negatives do not exist

5.11 False positives

False positives exist since both a document and its arbitrarily jumbled version will get same merit score. This is prevented by assuming grammatically correct documents or by preprocessor which does parts of speech parsing to validate the grammatical structure of the document.

5.12 Normalization

Intrinsic merit can be compared only if the compared documents are of same class. Thus 2 documents explaining special relativity can be compared while a document on journalism can not be compared with a document on special relativity. Intrinsic Merit scores can be normalized by,

$$\text{NormalizedIntrinsicMeritScore} = \text{Score}/\text{MaximumScore} \quad (10)$$

5.13 Ordering and Relative Merit

Definition 4. *Document1 is more meritorious than document2 if*

1. *document1 has more keywords that need to be understood than those of document2,*
2. *cohesion/interrelation of the keywords in document1 is more than that of document2,*
3. *average number of keywords per definition is greater for document1 than document2,*

4. *firstconvergencelevel(level at which first gloss overlap occurs) of document1 is less than that of document2 and*
5. *depth of definition tree of document1 is greater than that of document2.*

If we want a weaker definition of the above, ranking may be a partial order(where some pairs of documents may not be comparable) than a total order. This appeals to intuition since document1 may be better in some aspects but worse in some other relative to document2

5.14 Semantic relatedness or Meaningfulness of a document

Definition 5. *A document is meaningful to a human reader if any pair of keywords in the document are within a threshold WordNet distance e.g Jiang-Conrath distance*

5.15 Formal proof of correctness of Convergence and Intrinsic Merit Score

Theorem 1. *If a document lacks merit, convergence(or gloss overlap) does not occur (Corollary: Document's merit is measured by extent of convergence)*

Proof. By "meritorious" document, we imply a document which is meaningful as per the definition of meaningfulness above(i.e. keywords in a document are separated within threshold WordNet distance metric like Jiang-Conrath distance). Let us denote R as a relation "is descendant of". If xRy then y is in (gloss)definition tree of keyword x(i.e y is descendant of x). If definition trees of keywords of the document are disjoint, then there is no y such that xRy and zRy for two keywords x and z. Let us define the relation S to be "two keywords are related". xSz iff xRy and zRy for some y. Thus we formalise cohesiveness/meaningfulness of a document in terms of definition graph. If a document is not meaningful then there exist no x and z such that xSz , which implies that for no y, xRy and zRy . Thus there exist no vertex y which is in definition tree of two key words. Thus convergence is a necessary condition for merit. The relation S implies that there exists a path between two keywords x and y in the document, through some intermediate nodes which are in the definition/gloss tree of x and y. There exists a threshold WordNet distance greater than length of this path since the length is finite and whether a document is meaningful depends on this threshold. Thus convergence(generalized gloss overlap) implies meaningfulness of a document as per the definition above. Moreover Intrinsic merit increases with number of edges and relatedness() - linear or quadratic. So with greater relatedness() and more number of vertices and edges, overlaps and number of nodes involved in overlap increase. This in turn implies that more number of paths are available amongst the keywords of the document since every overlap acts as a meeting point of two keyword definition trees. Probability that lengths of these paths

are less than threshold WordNet distance is inversely proportional to firstconvergencelevel(level of first gloss overlap). Thus intrinsic merit score discussed earlier captures this notion. \square

5.16 Parallelizability

Recursive gloss overlap is parallelizable by partitioning the tokens at each level and assigning each subset to different processors (Map) to get the tokens for next level. Individual results from processors are merged (Reduce) to get the final set of tokens for a level. This is repeated for all levels. MapReduce can be applied for parallelism.

6 Interview Algorithm (applying (1) and/or (2) for computing intrinsic merit)

6.1 Motivation for Interview algorithm

Here we map the real world scenario of an interview being conducted on a candidate where a panel asks questions and judges the candidate based on the quality of answers by candidate - candidate is a document and it is "interviewed" by a reference set of authorities. Each document x is interviewed/evaluated by set of reference documents which will decide on the merit of the document x. Reference set initially consists of n user chosen authorities on the subject. Interview is set of queries made by reference set on the document and evaluating the answer to the queries. If x passes the interview it is inducted into reference set. Next document will be interviewed by n+1 documents including last selected document and so on. Hierarchy of interviews can be built. For example Document x interviews documents y and z. Document y interviews w and document z interviews p. Thus we get a tree of interviews (it could be a directed acyclic graph too, if a candidate is interviewed by more than one reference, one of which itself was a candidate earlier). The interview scores can be weighted and summed bottom-up to get the merit of the root (Analogy: hierarchy in an organization).

6.2 Steps of the Interview algorithm

1. Relevance of the document to the reference set is measured by a classifier (NaiveBayesian or SVM or search engine results for a query)
2. Intrinsic merit score of the document is computed either by Recursive Gloss Overlap algorithm (measures the meaningfulness/sanity of the candidate) (or) by citation digraph
3. Reference set interviews the candidate and gets the score
4. Value addition of the candidate document is measured (what extra value candidate brings over and above reference set)

5. Candidate is inducted into reference set based on the above criteria if candidate is above a threshold.

6.3 Mathematical formulation of an interview

Interview is abstracted in terms of a set of tuples, where each tuple is of the form

$$t(i) = (question, answer, expectedanswer, score) \quad (11)$$

for question i.

$$Interview(I) = \{t(1), t(2), t(3), \dots, t(n)\} \quad (12)$$

$$t(i).score = PercentageOfMatch(t(i).answer, t(i).expectedanswer) \quad (13)$$

$$if \left(\sum_{i=1}^n (t(i).score) \right) > referencethreshold \quad (14)$$

then induct the document into reference set. In the Information Retrieval context, a question is a query and the answer is the context within the document that matches the query. The answer returned by the document is then compared with expectedanswer. Comparison is done by Jaccard coefficient of shingles (n-grams)

$$t(i).score = \frac{|shingle(answer) \cap shingle(expectedanswer)|}{|shingle(answer) \cup shingle(expectedanswer)|} \quad (15)$$

1. Supervised: In supervised setting, each reference document is pre-equipped with user-decided set of queries and answers it expects. Thing to note is that a document is made a live object - it both has content and questions it intends to ask (set of search queries). Alternative way to compute $t(i).score$ is to find out the definition graph of answer and expectedanswer and compute the difference between the two graphs (e.g. edit distance). Downside of this is the assumption of pre-existence of correct answers which makes this a supervised learning.
2. Unsupervised: In the absence of reference questions and answers, questions that a document "intends" to ask can be thought of as set of queries for which the document has better answers (results). These set of queries/results (questions and answers) can be automatically obtained from a document through an unsupervised way by computing set of more likely to be important n-grams (by computing key phrases with tfidf above threshold) and the context of the n-grams in the reference documents. These n-grams/contexts can later be used as "reference questions" (n-grams) and "reference answers" (contexts of the corresponding n-grams) to the candidate document. Thus we compensate for lack of reference Questions and Answers. Alternatively,

an interview can be simply considered as the percentage similarity of definition-graph(reference) and definition-graph(candidate) obtained by edit distance.

6.4 Searching for answer to a query within the document (as implemented)

If a document describing tourist places is given and the query is "What are the good places to visit in this city?", then query is parsed into key words like "good", "places", "visit" and "city" and matching contexts within the document are returned where context is the phrase of length $2n + 1$ (from $x - n$ to $x + n$ locations with location of keyword being x).

6.5 Value addition measure

Recursive gloss overlap algorithm gives the definition graph of the candidate document. To measure the value addition we can run the recursive gloss overlap algorithm on the reference set to get the definition graph of reference set and find out the difference between the two definition graphs - reference and candidate. Since value addition is defined as the value added which is not already present, extra vertices and edges present in candidate but absent in the reference set are a measure of value addition. Value addition can be measured by either edit distance(cost of transforming one graph to the other after adding/deleting vertices/edges), maximum common subgraph or difference of adjacency matrices. Implementation uses graph edit distance measure.

6.6 Update summarization through Interview algorithm (applying algorithm given in 6.2)

Given a news summary and a candidate news to be added to summary

1. Label the summary as reference set.
2. Run a classifier on summary and candidate to get the class to which both belong to (or get from search engine results on a news topic)
3. If $\text{class}(\text{summary}) == \text{class}(\text{candidate})$ proceed further
4. Calculate intrinsic merit score of the candidate news document through recursive gloss overlap algorithm described in (2) (or) from citation digraph described in (1)
5. Candidate news is interviewed by reference set(summary in this case)
6. Compute value addition of candidate to summary
7. Add the value added information from candidate into existing summary to get new summary (by getting cream of sentences with top sentence scores)

6.7 Application to Topic Detection, Link detection and Tracking

1. Interview algorithm and graph edit distance measure can be applied to news topic link detection(Answers the question - Does a pair of news stories discuss same topic?). Since same news item falls under multiple topics and is changing over time, topic of a news story is in a state of flux. Given a pair of news stories (n1, n2) execute interview of n2 with n1 as reference. This interview score decreases and edit distance grows as n2 becomes more irrelevant to n1. By defining a threshold for interview and edit distance scores to belong to same topic, link detection can be achieved. It is important to note that interview score and value addition score are inversely related.
2. At any point in time, compute edit distance for all possible pairs Nx, Ny in a topic (after getting their respective definition graphs) and choose Ny which has largest edit distance to others and hence an outlier and least likely to be in the topic. Thus topic detection is achieved(Answers the question - Does this story exist in correct topic?).
3. Topic tracking can be done by constructing definition graph and finding vertices with high number of indegrees. These keywords are voted high and point to the maximum likely topic of the news story (works as an unsupervised text classifier).This process has to be periodically done since topic of a story might change and thus the definition graph will change.

6.8 Implementation in Python

1. Get documents of same class/topic (from Reuters corpus or search engine query results) and their future references as input
2. Compute the intrinsic merit score for both documents:
 - applying citation digraph construction (or) applying definition graph convergence(generalized recursive gloss overlap)
 - Parse into keywords and get keywords above a threshold tf-idf
 - Perform WSD using Lesk's algorithm
 - Get glosses of matching sense through wordnet api
 - get overlaps at level i, update intrinsic merit score (either using linear or quadratic overlap)
 - repeat for sufficient number of levels defined by "depth"
3. execute interview if reference questions and answers are available (supervised) or through getting important n-grams/context described above (unsupervised) - at present restricted to 1-gram for keywords and bigrams for jaccard coefficient calculation

4. compute value addition through definition graph edit distance between reference and candidate, and get the score.
5. get percentage weighted sum of intrinsic merit, interview and value addition scores and get final score.
6. **APPLY (2), (3) and (4) ABOVE TO NEWS UPDATE SUMMARIZATION:** If final score is above threshold, update the news summary with candidate news and publish (sentence scoring is done by sum of tfidf scores of words in a sentence)
7. **APPLY (2) TO GET INTRINSIC MERIT RANKING:** Compare the scores of the two documents from (2) and rank. (Citation graph based maxflow ranking internally uses sentiment analysis using one of 1) Recursive gloss overlap 2) SentiWordNet 3) Entropy analysis of citation context)
8. **APPLY (3) and (4) ABOVE TO TOPIC DETECTION AND TRACKING**

7 Results

7.1 Results - Intrinsic Merit score with quadratic overlap for top 10 Google ranked documents for query 'data mining' sorted ascending

Pagerank - (Document,relatedness,vertices,edges,firstconvergencelevel) -
IMScore

- 6 - ('ThesisDemo-datamining-test6.txt', 477660, 372, 576, 1) - 102349163520.0
- 10 - ('ThesisDemo-datamining-test10.txt', 139114790, 1172, 2339, 1) - 3.81356486745e+14
- 8 - ('ThesisDemo-datamining-test8.txt', 310161784, 1456, 3034, 1) - 1.37014092147e+15
- 7 - ('ThesisDemo-datamining-test7.txt', 310161784, 1456, 3034, 1) - 1.37014092147e+15
- 5 - ('ThesisDemo-datamining-test5.txt', 51304180926L, 2938, 10643, 1) - 1.60423730814e+18
- 4 - ('ThesisDemo-datamining-test4.txt', 99651694978L, 3324, 12921, 1) - 4.27998090689e+18
- 9 - ('ThesisDemo-datamining-test9.txt', 133686525217L, 3186, 13468, 1) - 5.73636152749e+18

- 3 - ('ThesisDemo-datamining-test3.txt', 354003740698L, 3901, 18039, 1) - 2.49112924394e+19
- 2 - ('ThesisDemo-datamining-test2.txt', 594730534291L, 3935, 20502, 1) - 4.79801059042e+19
- 1 - ('ThesisDemo-datamining-test1.txt', 2753901168066L, 5832, 33386, 1) - 5.36204253324e+20

7.2 Results - Intrinsic Merit score with quadratic overlap for top 10 Google ranked documents for query 'philosophy' sorted ascending

Pagerank - (Document,relatedness,vertices,edges,firstconvergencelevel) - IMScore

- 3 - ('ThesisDemo-philosophy-test3.txt', 63840296, 1110, 2165, 1) - 1.53417807332e+14
- 7 - ('ThesisDemo-philosophy-test7.txt', 456552729, 1234, 3041, 1) - 1.71325703153e+15
- 5 - ('ThesisDemo-philosophy-test5.txt', 915190280, 1428, 3651, 1) - 4.77146166914e+15
- 6 - ('ThesisDemo-philosophy-test6.txt', 1128268242, 1891, 4577, 1) - 9.76528235921e+15
- 2 - ('ThesisDemo-philosophy-test2.txt', 2739304610L, 2033, 5316, 1) - 2.96048373426e+16
- 10 - ('ThesisDemo-philosophy-test10.txt', 6630859968L, 2289, 6471, 1) - 9.82170869184e+16
- 9 - ('ThesisDemo-philosophy-test9.txt', 7675201402L, 2105, 6477, 1) - 1.04644348307e+17
- 8 - ('ThesisDemo-philosophy-test8.txt', 9692242200L, 2165, 6733, 1) - 1.41283281476e+17
- 4 - ('ThesisDemo-philosophy-test4.txt', 14535833906L, 2553, 7920, 1) - 2.93911072979e+17
- 1 - ('ThesisDemo-philosophy-test1.txt', 9611266377319L, 7552, 49449, 1) - 3.58922024377e+21

7.3 Results - Intrinsic Merit score with quadratic overlap for human (2 judges) judged documents on topic 'democracy' - sorted ascending

Human ranking - (Document,relatedness,vertices,edges,firstconvergencelevel) - IMScore

- 5,5 - ('ThesisDemo-democracy-test2.txt', 15535, 270, 406, 1) - 1702946700.0
- 4,6 - ('ThesisDemo-democracy-test6.txt', 60534, 253, 373, 1) - 5712533046.0
- 6,1 - ('ThesisDemo-democracy-test1.txt', 136281, 249, 384, 1) - 13030644096.0
- 2,2 - ('ThesisDemo-democracy-test4.txt', 245448, 358, 568, 1) - 49910378112.0
- 1,3 - ('ThesisDemo-democracy-test3.txt', 1623723, 364, 671, 1) - 396584600412.0
- 3,6 - ('ThesisDemo-democracy-test5.txt', 1167039, 485, 847, 1) - 479413786005.0

7.4 Results - Intrinsic Merit score with quadratic overlap for human (1 judge) judged documents on topic 'soap' - sorted ascending

Human ranking - (Document,relatedness,vertices,edges,firstconvergencelevel) - IMScore

- 4 - ('ThesisDemo-soap-test4.txt', 52, 212, 346, 2) - 1907152.0
- 3 - ('ThesisDemo-soap-test2.txt', 735, 113, 146, 1) - 12126030.0
- 2 - ('ThesisDemo-soap-test3.txt', 1368, 109, 152, 1) - 22665024.0
- 1 - ('ThesisDemo-soap-test1.txt', 2912, 188, 251, 1) - 137411456.0
- 5 - ('ThesisDemo-soap-test5.txt', 25641, 230, 353, 1) - 2081792790.0

7.5 Results - Intrinsic Merit score with quadratic overlap for top 7 Google news stories for query 'haiti earthquake' - sorted ascending

Pagerank - (Document,relatedness,vertices,edges,firstconvergencelevel) - IMScore

- 4 - ('ThesisDemo-haiti-test4.txt', 11683630, 710, 1343, 1) -
1.11406917139e+13
- 2 - ('ThesisDemo-haiti-test2.txt', 65287245, 1002, 2008, 1) -
1.31358981536e+14
- 7 - ('ThesisDemo-haiti-test7.txt', 219493417, 1258, 2785, 1) -
7.69001771262e+14
- 6 - ('ThesisDemo-haiti-test6.txt', 491851745, 1321, 3223, 1) -
2.09409962803e+15
- 3 - ('ThesisDemo-haiti-test3.txt', 4268535180L, 1966, 5693, 1) -
4.7775315353e+16
- 5 - ('ThesisDemo-haiti-test5.txt', 7043167094L, 2120, 6412, 1) -
9.57408693023e+16
- 1 - ('ThesisDemo-haiti-test1.txt', 44329850203L, 3052, 10603, 1) -
1.434529734e+18

7.6 Results - Intrinsic Merit score with quadratic overlap for top 10 Google ranked documents for query 'literary' sorted ascending

Pagerank - (Document,relatedness,vertices,edges,firstconvergencelevel) -
IMScore

- 5 - ('ThesisDemo-literary-test5.txt', 38252283, 1032, 1944, 1) -
7.67420361729e+13
- 7 - ('ThesisDemo-literary-test7.txt', 815611695, 2020, 4386, 1) -
7.22609124643e+15
- 3 - ('ThesisDemo-literary-test3.txt', 5989035631L, 2039, 5938, 1) -
7.25127400033e+16
- 10 - ('ThesisDemo-literary-test10.txt', 6155411625L, 2467, 6713, 1) -
1.01939593415e+17
- 8 - ('ThesisDemo-literary-test8.txt', 296376674293L, 4598, 18333, 1) -
2.4983111474e+19
- 4 - ('ThesisDemo-literary-test4.txt', 529359994275L, 5074, 21758, 1) -
5.84413920691e+19
- 2 - ('ThesisDemo-literary-test2.txt', 643163471944L, 4920, 22433, 1) -
7.09861839373e+19
- 1 - ('ThesisDemo-literary-test1.txt', 1149789557857L, 5126, 25916, 1) -
1.52744272126e+20

- 9 - ('ThesisDemo-literary-test9.txt', 2149531315027L, 6056, 31756, 1) - 4.13385687561e+20
- 6 - ('ThesisDemo-literary-test6.txt', 3388627800057L, 6826, 36617, 1) - 8.4697952824e+20

7.7 Results Excerpt- Intrinsic Merit Ranking of Reuters corpus in 'earn' category

(Document,relatedness,vertices,edges,firstconvergencelevel) - IMScore

- ('test/15046', 34, 59, 93, 1) - 186558.0
- ('test/14911', 55, 164, 219, 1) - 1975380.0
- ('test/15213', 65, 169, 234, 1) - 2570490.0
- ('test/15063', 105, 226, 331, 2) - 3927315.0
- ('test/14899', 79, 199, 278, 1) - 4370438.0
- ('test/15070', 107, 199, 306, 1) - 6515658.0
- ('test/15185', 117, 210, 327, 1) - 8034390.0
- ('test/15074', 116, 219, 335, 1) - 8510340.0
- ('test/15103', 107, 259, 366, 1) - 10142958.0
- ('test/14965', 125, 232, 357, 1) - 10353000.0

7.8 Results - Spearman ranking coefficient and Pearson coefficient for the above rankings

- (1) . Spearman ranking coefficient for Google ranking for 'data mining' : 0.733333333333
- (2) . Pearson ranking coefficient for Google ranking for 'data mining' : 0.00888888888889
- (1) . Spearman ranking coefficient for Google ranking for 'literary' : 0.0909090909091
- (2) . Pearson ranking coefficient for Google ranking for 'literary' : 0.00110192837466
- (1) . Spearman ranking coefficient for Google ranking for 'philosophy' : 0.0424242424242
- (2) . Pearson ranking coefficient for Google ranking for 'philosophy' : 0.000514233241506

- (1) . Spearman ranking coefficient for Google ranking for 'haiti earthquake' : 0.25
- (2) . Pearson ranking coefficient for Google ranking for 'haiti earthquake' : 0.00892857142857
- (1) . Spearman ranking coefficient for Human ranking(2 judges) for 'democracy' : 0.385714285714
- (2) . Pearson ranking coefficient for Human ranking(2 judges) for 'democracy' : 0.0174334140436
- (1) . Spearman ranking coefficient for Human ranking(1 judge) for 'soap' : 0.9
- (2) . Pearson ranking coefficient for Human ranking(1 judge) for 'soap' : 0.09

As per Spearman coefficient, correlations between Google ranking and Recursive Gloss Overlap are 73%, 4%, 9% and 25% while with human ranking they are 38% and 90%

7.9 Results - Update Summarization applying Interview algorithm with Recursive Gloss Overlap - Example - Summary size is 12.5%

Candidate document:

1 Dead in Bangkok Protests, More Than 70 Wounded

VOA News22 April 2010 A Thai woman lies injured on the ground in Bangkok after several small explosions occurred near site of anti-government protests in Bangkok, 22 Apr 2010 Photo: AP

A Thai woman lies injured on the ground in Bangkok after several small explosions occurred near site of anti-government protests in Bangkok, 22 Apr 2010

Hospitals in Thailand's capital, Bangkok, say at least one person has been killed and many others wounded in a series of explosions near an encampment of anti-government protesters.

More than 70 people were reported wounded Thursday at the camp in the city's business district, which is packed with armed troops and diverse groups of protesters. Reports say at least five hand grenades exploded, prompting the closure of a nearby train station.

The protesters have besieged central Bangkok for weeks, trying the patience of citizens and business leaders. A coalition of groups gathered to drive the so-called Red Shirt protesters from the main retail and tourist district. Police separated the two sides.

The Red Shirt protesters have rallied for five weeks, demanding new elections and the resignation of Prime Minister Abhisit Vejjajiva.

A coalition opposed to the Red Shirts - called the Multi-Colored Shirts - has announced a mass rally for Friday.

Most of the protesters support former Prime Minister Thaksin Shinawatra, who was ousted in 2006. Mr. Thaksin lives in exile and faces a prison sentences on corruption charges in Thailand. He has a significant following among the country's rural and low-income population.

Mr. Abhisit came to power in December 2008, after months of massive anti-Thaksin protests by demonstrators known as the Yellow Shirts.

The military said soldiers will use tear gas, rubber bullets and live ammunition, if necessary, to remove the protesters. But Army Chief General Anupong Paochinda has been reluctant to use arms fearing renewed bloodshed.

An April 10 clash between the Red Shirts and Thai security forces left at least 25 people dead, and 850 others injured.

Reference document:

Bangkok blasts kill one, injure 75 - Thai media 11:16am EDT

BANGKOK, April 23 (Reuters) - A series of grenade blasts that rocked Bangkok's business district on Friday killed at least one person and wounded 75, hospitals and the Thai media said.

Five M-79 grenades hit an area packed with heavily armed troops and studded with banks, office towers and hotels. Four of the wounded had serious injuries, including two foreigners, according to witnesses, hospital officials and an army spokesman. (Additional reporting by Nopporn Wong-Anan; Writing by Jason Szep; Editing by Bill Tarrant)

Summary:

1 Dead in Bangkok Protests, More Than 70 Wounded

VOA News 22 April 2010 A Thai woman lies injured on the ground in Bangkok after several small explosions occurred near site of anti-government protests in Bangkok, 22 Apr 2010 Photo: AP

A Thai woman lies injured on the ground in Bangkok after several small explosions occurred near site of anti-government protests in Bangkok, 22 Apr 2010

Hospitals in Thailand's capital, Bangkok, say at least one person has been killed and many others wounded in a series of explosions

near an encampment of anti-government protesters . Bangkok blasts kill one, injure 75 - Thai media 11:16am EDT

BANGKOK, April 23 (Reuters) - A series of grenade blasts that rocked Bangkok's business district on Friday killed at least one person and wounded 75, hospitals and the Thai media said .

7.10 Results Excerpt - Topic Detection and Tracking applying Interview algorithm with Recursive Gloss Overlap

Example Reference News Story (ThesisDemo-ipad-test1) - Topic 'ipad':

Apple unveils iAd platform; iPad sales look strong Photo 6:29am EDT

By Gabriel Madway

CUPERTINO, California (Reuters) - Apple CEO Steve Jobs showed off a new smartphone operating system on Thursday that features an advertising platform to compete with Google's, and revealed stronger-than-expected sales of 450,000 units for the iPad.

The iPhone 4.0 software will be available on Apple's hugely popular smartphone this summer, complete with a number of upgrades, including a long-awaited multi-tasking capability that allows the use of several applications at once.

A version of the iPhone's operating system is also used on the iPad, and the latest generation of software will come to Apple's new tablet computer this fall.

The new advertising platform for the iPhone and iPad – dubbed iAd – marks Apple's first foray into a small but growing market, and is sure to please the thousands of application developers who make their living off those devices, providing them with a new revenue stream.

The iPad's early sales impressed analysts, many of whom expect 1 million units to be sold in the quarter ending June, and roughly 5 million in 2010, though estimates vary widely.

"We're making them as fast as we can. Our ramp is going well, but evidently we can't quite make enough of them yet so we're going to have to try harder," Jobs said, noting iPad sellouts at Best Buy stores.

The electronics giant has staked its reputation on the 9.7-inch touchscreen tablet, essentially a cross between a smartphone and a laptop. It is helping foster a market for tablet computers that is expected to grow to as many as 50 million units by 2014, according to analysts.

"I think it's pretty impressive, five days almost half a million units, and it shows there's still pretty good momentum behind the first day," said Gartner analyst Van Baker.

Despite critics who question whether a true need exists for such a gadget, analysts expect Hewlett-Packard, Dell and others to trot out their own competing devices this year.

Since the iPad went on sale on April 3, users have downloaded 600,000 digital books and 3.5 million applications for the device, Jobs said. There are already 3,500 apps available for the iPad.

"It was above my expectations, frankly," said Joe Clark, managing partner of Financial Enhancement Group, referring to iPad sales.

"The day the original Apps Store launched it was a game change for the iPhone and it will do the same eventually for the iPad."

At a media event at the company's Cupertino, California, headquarters, Jobs said Apple had so far sold more than 50 million iPhones, the smartphone that competes with Research in Motion's Blackberry and Motorola's Droid.

That implies that the company sold 7 million or more devices in the March quarter, which would be above many analysts' forecasts.

MOBILE AD WAR

Apple is expected to launch the fourth-generation model of its iPhone, which was introduced in 2007, later this year.

Pancreatic cancer survivor Jobs, looking thin but energetic, introduced the iAd mobile platform, which he said had the opportunity to make 1 billion ad impressions a day on tens of millions of Apple mobile device users.

iAds will allow applications developers to use advertisements in their apps, pocketing 60 percent of the revenue. Apple will sell and host the ads.

Jobs harshly criticized the current manner and look of mobile advertising, particularly search ads. He promised that iAds will foster more engaging advertising that will not pull users away from the content within apps.

NEW ARENA

Tim Bajarin, president of consulting company Creative Strategies, said it was a dramatic shift in thinking about the delivery of mobile ads, and an obvious move by Apple to set itself apart from Google Inc, which made its name on search ads.

"It's very clear that Jobs believes that ads in the context of apps makes more sense than generic mobile search," he said.

Apple's entry into the mobile ad arena had been widely expected. This year, it paid \$270 million for Quattro Wireless, an advertising

network that spans both mobile websites and smartphone applications.

Google, which already sells advertising on smartphones, agreed to buy mobile ad firm AdMob late in 2009. U.S. regulators are examining the deal's antitrust implications.

Jobs said Apple was also in the hunt to buy AdMob before Google "snatched them from us because they didn't want us to have them." The comments were just the latest hint at the rift that has emerged between Apple and Google, which were once allies but now compete in a number of arenas.

Research group Gartner expects the mobile advertising market to expand by 78 percent to \$1.6 billion in 2010.

Jobs also said the new operating system will include support for multi-tasking – addressing a perennial consumer complaint – allowing users to switch between several programs running simultaneously.

Shares of Apple turned positive briefly after Jobs' announcement, before quickly dipping back into negative territory. They closed 0.3 percent lower at \$239.94 on the Nasdaq. (Writing by Edwin Chan; Editing by Steve Orlofsky, Leslie Gevirtz and Matthew Lewis)

Example Candidate News Story (ThesisDemo-dantewada-test1) - Topic 'Dantewada':

Chidambaram offers to quit, PM says no CNN-IBN

New Delhi: Union Home Minister P Chidambaram has reportedly offered to resign taking responsibility for the Dantewada massacre in which the Maoists butchered 76 security personnel. However, Prime Minister Manmohan Singh has rejected Chidambaram's offers to resign.

Chidambaram reportedly met Prime Minister Manmohan Singh taking "full responsibility" for the Dantewada attack on a CRPF patrol party and offered to step down.

Sources say that a section within the Congress party has been unhappy with the way Chidambaram has handled the Maoist issue so far including his tough talk on the rebels and his friction with West Bengal Chief Minister Buddhadeb Bhattacharjee.

When the Home Minister came back from his Dantewada tour he reportedly met the Prime Minister and told him that if he (Prime Minister) was dissatisfied with his performance, he was ready to step down.

Earlier, on Friday while attending the Valour Day function of the CRPF, Chidambaram said, "I salute the CRPF. I promise that

government will always stand by you. Where does the buck stop after Dantewada? The buck stops at my desk. I accept full responsibility of what happened in Dantewada. I told this to the Prime Minister as well."

In the past whenever there was talk of all-out action to control Maoists, it was usually tempered by the Congress party that the issue it was a socio-economic one and had to be dealt with in such a manner.

Many leaders had been maintaining that Maoists were our own people and so they should not be dealt with in the same firm way as one would deal with terrorists.

But now that stand seems to have been diluted a bit within the Congress following the massacre of the CRPF team on Tuesday.

The party, too, has begun to realise that to go soft in the weeding out Maoists will not really go down very well.

So Chidambaram's offer to step down accepting complete responsibility is seen as a political masterstroke by him and also a plus point in his favour.

A large group of Maoists had ambushed the CRPF team belonging to the 62 Battalion between 6 AM and 7 AM on Tuesday between Chintalnar and Tademetla villages in Sukma block of Dantewada district of Chhattigarh when the security personnel were on the way to Tademetla in a vehicle.

The Maoists, believed to be between 200-1000, first triggered a land mine destroying the vehicle carrying the security personnel. In the ensuing gunbattle 75 CRPF men and one local police constable were killed.

Topic Link Detection(to check if above two news stories discuss same topic):

- Interview score:3.09090909091
- Interview score(in percentage correctness): 10.303030303
- Edit Distance (as percentage value addition from reference):79.4014084507
- Topic Link Detection - ThesisDemo-ipad-test1.txt and ThesisDemo-dantewada-test1.txt do not discuss same topic

Topic Detection(to check if a news story is under correct topic) - Topics are 'ipad'(1 story), 'sukhna'(2 stories), 'lufthansa'(1 story), 'dantewada'(1 story):

- Topic Detection - News story ThesisDemo-ipad-test1.txt has largest pairwise editdistance from ThesisDemo-sukhna-test1.txt and least likely to be in this topic

- Topic Detection - News story ThesisDemo-ipad-test1.txt has largest pairwise editdistance from ThesisDemo-lufthansa-test1.txt and least likely to be in this topic
- Topic Detection - News story ThesisDemo-ipad-test1.txt has largest pairwise editdistance from ThesisDemo-sukhna-test2.txt and least likely to be in this topic
- Topic Detection - News story ThesisDemo-ipad-test1.txt has largest pairwise editdistance from ThesisDemo-dantewada-test1.txt and least likely to be in this topic
- Topic Detection - News story ThesisDemo-dantewada-test1.txt has largest pairwise editdistance from ThesisDemo-ipad-test1.txt and least likely to be in this topic

7.11 Results Excerpt - Applying Sentiment Analysis with Recursive Gloss Overlap for finding polarity of an edge in Citation Graph Maxflow (1)

- Nodes with more than 1 parent (and hence the most likely classes of document) are: set(['good', 'used'])
- getPositivity: good
- getNegativity: good
- getPositivity: used
- getNegativity: used
- negative words: []
- positive words: ['good', 'used']

7.12 Results Excerpt - Citation graph maxflow with a simple link graph example - polarity determined by Recursive Gloss Overlap

- Following is the adjacency matrix for hyperlink graph amongst 7 html documents (file1.html, file2.html, file3.html, file4.html, file5.html, file6.html, file7.html)

$$\begin{aligned}
 \text{Adjacency Matrix of Link Graph} = & \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 \\ -1 & 1 & 1 & 0 & -1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}
 \end{aligned}$$

- Average concept maxflow out of each page:
 - 'file2.html': 3.7142857142857144
 - 'file4.html': 3.2857142857142856
 - 'file5.html': 2.0
 - 'file3.html': 3.4285714285714284
 - 'file7.html': 0.0
 - 'file1.html': 3.4285714285714284
 - 'file6.html': 0.0
- Number of nodes within radius 2 of the source file2.html = 7
- Number of nodes within radius 2 of the source file3.html = 7
- Number of nodes within radius 2 of the source file4.html = 7
- Number of nodes within radius 2 of the source file5.html = 6
- Number of nodes within radius 2 of the source file1.html = 7
- Number of nodes within radius 2 of the source file6.html = 0
- Number of nodes within radius 2 of the source file7.html = 0

Average maxflow values above show that file2, file1, file3, file4, file5, file6, file7 are ranked in descending order of their Maxflow merit. Thus file2 is most influential in this community. The radius measure with length 2 gives the ranking: file2, file3, file4, file1 all tied in first and file5 ranked next.

7.13 Conclusion

Motivation for this exercise, as evident from results above, is to explore the possibility of finding an algorithm/framework to assess the merit of a document with and without link graph structure in place with greater emphasis on the latter. Citation graph maxflow measures the penetration of a concept (represented in a document), in a link graph while the Recursive gloss overlap objectively judges the document without getting inputs from any incoming links. Interview algorithm uses either of these two algorithms and abstracts some real world applications. As seen above Google rankings differ (with some exceptions) from Recursive gloss overlap intrinsic merit rankings which is on the expected lines - content-and-complexity based merit scoring is not necessarily same as popularity based ranking. Moreover the intrinsic ranking scheme given above need not be the only possible way of computing merit. Once we have definition graph for a document (whether multipartite or not), multitude of more ranking schemes can be invented - for example based on 1) k-connectedness of the definition graph 2) completeness or robustness of the definition graph 3) (multipartite) cliques of (multipartite) definition graph

(if multipartite) etc., Since definition graph construction is computationally intensive, there is a scope of improvement in improving the recursive gloss overlap algorithm by applying some parallel processing framework like MapReduce. Applying Evocation WordNet, implementing a MapReduce(e.g Hadoop) cluster and considering more than one relation are future directions to think about. Theoretical foundation for the recursive gloss overlap comes from WordNet itself which visualises the relatedness of words - Definition graph is just an induced subgraph of WordNet for a document. Since merit quantification of a document can not be done without analyzing relatedness of keywords in a document, definition graph, which is a subgraph of WordNet for a document, is a plausible representation. Accuracy of Recursive gloss overlap depends on the accuracy of WordNet, depth to which definition trees are grown and Word Sense Disambiguation.

References

- [1] Graph Similarity, Master's thesis by Laura Zager and George Verghese
EECS MIT 2005
- [2] Edit distance and its computation, presentation by Joszef Balogh and
Ryan Martin
- [3] Extended Gloss Overlaps as a measure of semantic relatedness, Satanjeev
Banerjee and Ted Pedersen
- [4] Semantic Language Models for TDT, Ramesh Nallapati, University of
Amherst
- [5] WordNet Evocation Project-
<http://wordnet.cs.princeton.edu/downloads/evocation/release-0.4/README.TXT>
- [6] SentiWordNet - <http://sentiwordnet.isti.cnr.it/>
- [7] WordNet - <http://wordnet.princeton.edu>
- [8] Navigli, R. 2009. Word sense disambiguation: A survey. ACM Comput.
Surv. 41, 2, Article 10 (February 2009)
- [9] Temporal information in Topic Detection and Tracking - Juha Makkonen,
University of Helsinki
- [10] Overview NIST Topic Detection and Tracking by G. Doddington ,
<http://www.itl.nist.gov/iaui/894.01/tests/tdt/tdt99/presentations/index.htm>
- [11] Topic Detection and Tracking Pilot Study - James Allan , Jaime
Carbonell , George Doddington , Jonathan Yamron , and Yiming Yang
UMass Amherst, CMU, DARPA, Dragon Systems, and CMU

- [12] The cognitive revolution: a historical perspective , George A. Miller
Department of Psychology, Princeton University, TRENDS in Cognitive Sciences Vol.7 No.3 March 2003
- [13] On Bipartite and Multipartite Clique Problems, Milind Dawande, Pinar Keskinocakc, Jayashankar M. Swaminathan and Sridhar Tayur,Journal of Algorithms 41, 388â€¢403 (2001)
- [14] Python Natural Language Toolkit - <http://nltk.sourceforge.net>
- [15] Partitioning CiteSeer's Citation Graph - Revised Version , Gregory Mermoud, Marc A. Schaub, and Gregory Theoduloz, School of Computer and Communication Sciences, Ecole Polytechnique Federale de Lausanne (EPFL), 1015 Lausanne, Switzerland
- [16] Semantic distance in WordNet: An experimental, application-oriented evaluation of five measures, Alexander Budanitsky and Graeme Hirst, Department of Computer Science, University of Toronto
- [17] The Official Python Tutorial - <http://docs.python.org/tut/tut.html>
- [18] MapReduce: Simplified Data Processing On Large Clusters, Jeffrey Dean and Sanjay Ghemawat, Google Inc.,
- [19] Opinion Mining and Summarization - Sentiment Analysis , Bing Liu
Department of Computer Science , University of Illinois at Chicago , Tutorial given at WWW-2008, April 21, 2008 in Beijing WWW 2008
- [20] Web Data Mining, Bing Liu, Department of Computer Science , University of Illinois at Chicago
- [21] Introduction to Algorithms - Second Edition, Thomas H.Cormen, Charles E.Liecerson, Ronald L.Rivest, Clifford Stein

(44) DECEMBER [21/11/2013] Discrete Hyperbolic Implement Factorization NC 2013a in

2013 In this sorted merged file DECEMBER 6

k -tile merge for above 3 tiles:

$$\text{tile 1} = (x, y, x + \Delta x, y)$$

$$\text{tile 2} = (x + \Delta x, y - 1, x + 2\Delta x, y - 1)$$

$$\text{tile 3} = (x + 2\Delta x, y - 2, x + 3\Delta x, y - 2)$$

After merger position of elements get shuffled. So binary search on the merged sorted tile needs to get the coordinate information with some additional ~~key factors~~ header.

for example: if $x = 3$ $y = 5$ and Δx

5 Thursday $x = 3$ fixed at 6, above tiles are

$y = 4$	15	20	25	30	35	40
$y = 5$	40	44	48	52	56	
$y = 6$	45	48	51	54	57	60
$y = 7$						

If a single rectangular block is considered, it is $N \times N$ and total number of tiles in the hyperbola is $\frac{N}{\Delta x} + \frac{N}{\Delta y} + \dots + \frac{N}{\Delta z}$ $\leq N^2 + \frac{N}{2^2} + \frac{N}{3^2}$

Merge needs $\frac{\log N \times (\log(N \times 1.33))}{N \text{ tiles}} \leq \frac{\log N \times (\log(N \times 1.33))}{6}$ Saturday 7

$= O((\log N)^2)$ with binary search, this is

$O((\log N)^2 + \log N)$

needs

If a merge algorithm has $\frac{\log N}{N}$ processors in CRCW model (in the complete distributed hyperbola)

1.33 N memory locations need to be populated with products and tags. tile id(s).

Due to CRCW with products and tags. tile id(s).

Sorted merged tile of above 3 is

$15, 20, 25, 30, 35, 40, 44, 45, 48, 48, 51, 52$

$45, 48, 51, 54, 57, 60$

$4, 4, 3, 1, 1, 1$

$3, 4, 3, 3$

NOV	M	T	W	T	F	S
2013	11	12	13	14	15	16
	12	13	14	15	16	17
	13	14	15	16	17	18
	14	15	16	17	18	19
	15	16	17	18	19	20
	16	17	18	19	20	21
	17	18	19	20	21	22
	18	19	20	21	22	23
	19	20	21	22	23	24

NOV	M	T	W	T	F	S
2013	11	12	13	14	15	16
	12	13	14	15	16	17
	13	14	15	16	17	18
	14	15	16	17	18	19
	15	16	17	18	19	20
	16	17	18	19	20	21
	17	18	19	20	21	22
	18	19	20	21	22	23
	19	20	21	22	23	24

NOV	M	T	W	T	F	S
2014	1	2	3	4	5	6
	2	3	4	5	6	7
	3	4	5	6	7	8
	4	5	6	7	8	9
	5	6	7	8	9	10
	6	7	8	9	10	11
	7	8	9	10	11	12
	8	9	10	11	12	13
	9	10	11	12	13	14
	10	11	12	13	14	15
	11	12	13	14	15	16
	12	13	14	15	16	17
	13	14	15	16	17	18
	14	15	16	17	18	19
	15	16	17	18	19	20
	16	17	18	19	20	21
	17	18	19	20	21	22
	18	19	20	21	22	23
	19	20	21	22	23	24

NOV	M	T	W	T	F	S
2014	20	21	22	23	24	25
	21	22	23	24	25	26
	22	23	24	25	26	27
	23	24	25	26	27	28
	24	25	26	27	28	29
	25	26	27	28	29	30
	26	27	28	29	30	31

NOV	M	T	W	T	F	S
2014	20	21	22	23	24	25
	21	22	23	24	25	26
	22	23	24	25	26	27
	23	24	25	26	27	28
	24	25	26	27	28	29
	25	26	27	28	29	30
	26	27	28	29	30	31

9 Monday Implications graphs and convex Hulls

To prove:

No polynomial sized subgraph of the universal implication graph, that contains the theorem premise and conclusion has a path that connects the premise and conclusion within it.



10 Tuesday

Set connectivity is NL

NL = coNL

Set non-connectivity is coNL

Immerman-Szelepcsenyi theorem

Open

Set ~~non~~ connectivity = } there is no path between S to t in graph

is in NL coNP

Algorithm needed to find minimum

Subgraph containing a path

Something similar to convex hull

Mon	Tue	Wed	Thu	Fri	Sat	Sun
11	12	13	14	15	16	17
12	13	14	15	16	17	18
13	14	15	16	17	18	19
14	15	16	17	18	19	20
15	16	17	18	19	20	21
16	17	18	19	20	21	22
17	18	19	20	21	22	23
18	19	20	21	22	23	24
19	20	21	22	23	24	25
20	21	22	23	24	25	26
21	22	23	24	25	26	27
22	23	24	25	26	27	28
23	24	25	26	27	28	29
24	25	26	27	28	29	30
25	26	27	28	29	30	31

Mon	Tue	Wed	Thu	Fri	Sat	Sun
11	12	13	14	15	16	17
12	13	14	15	16	17	18
13	14	15	16	17	18	19
14	15	16	17	18	19	20
15	16	17	18	19	20	21
16	17	18	19	20	21	22
17	18	19	20	21	22	23
18	19	20	21	22	23	24
19	20	21	22	23	24	25
20	21	22	23	24	25	26
21	22	23	24	25	26	27
22	23	24	25	26	27	28
23	24	25	26	27	28	29
24	25	26	27	28	29	30
25	26	27	28	29	30	31

Convexity in graphs Wednesday 11
where every proof path is a geodesic.

It suffices to prove that no polynomial-sized convex set containing a proof geodesic can be formed (or) all convex sets having proof geodesic are superpolynomial in size.

Convexity number of a digraph is the maximum cardinality of proper convex sets of D . Maximum convex set of a digraph D is a convex set with cardinality equal to convex number.

Convex Hull

Thursday 12

For a nonempty subset A of VD , the convex hull in the view of graph D , the convex hull is the minimal convex set containing A .

18/12/2013

Convex hull of proof geodesic (which is a subset of vertices) is the minimum learning neighborhood. Size of the convex hull decides the complexity class of a problem. If it is polynomial size then the problem is NP and if superpolynomial, in NP.

Mon	Tue	Wed	Thu	Fri	Sat	Sun
1	2	3	4	5	6	7
2	3	4	5	6	7	8
3	4	5	6	7	8	9
4	5	6	7	8	9	10
5	6	7	8	9	10	11
6	7	8	9	10	11	12
7	8	9	10	11	12	13
8	9	10	11	12	13	14
9	10	11	12	13	14	15
10	11	12	13	14	15	16
11	12	13	14	15	16	17
12	13	14	15	16	17	18
13	14	15	16	17	18	19
14	15	16	17	18	19	20
15	16	17	18	19	20	21
16	17	18	19	20	21	22
17	18	19	20	21	22	23
18	19	20	21	22	23	24
19	20	21	22	23	24	25
20	21	22	23	24	25	26
21	22	23	24	25	26	27
22	23	24	25	26	27	28
23	24	25	26	27	28	29
24	25	26	27	28	29	30
25	26	27	28	29	30	31

Mon	Tue	Wed	Thu	Fri	Sat	Sun
1	2	3	4	5	6	7
2	3	4	5	6	7	8
3	4	5	6	7	8	9
4	5	6	7	8	9	10
5	6	7	8	9	10	11
6	7	8	9	10	11	12
7	8	9	10	11	12	13
8	9	10	11	12	13	14
9	10	11	12	13	14	15
10	11	12	13	14	15	16
11	12	13	14	15	16	17
12	13	14	15	16	17	18
13	14	15	16	17	18	19
14	15	16	17	18	19	20
15	16	17	18	19	20	21
16	17	18	19	20	21	22
17	18	19	20	21	22	23
18	19	20	21	22	23	24
19	20	21	22	23	24	25
20	21	22	23	24	25	26
21	22	23	24	25	26	27
22	23	24	25	26	27	28
23	24	25	26	27	28	29
24	25	26	27	28	29	30
25	26	27	28	29	30	31

DECEMBER To prove PC NP it (48) 2013

13 Friday Suffixes to prove that all convex hulls are not of polynomial size

(superpolynomial size) asking implication is always decidable.

Discrete Hyperbolic Factorization - Segmented

Shifting Upward

$$[\text{10/12/2013}] \log_{1.2} N + \log_{2.3} N + \dots + \log_{\sqrt{N}} N$$

$$\frac{1}{2} \log_N (f_{N-1})$$

$$\frac{1}{2} \log_N (f_{N-1} f_{N-2})$$

$$\frac{1}{2} \log_N (f_{N-1} f_{N-2} f_{N-3})$$

$$= (f_{N-1}) \log N - \left[\log(f_{N-1}) + \log(f_{N-1}) \right]$$

$$14 \text{ Saturday} = (f_{N-1}) \log N - \left[(f_{N-1} \log(f_{N-1}) + (f_{N-1}) \log(f_{N-1})) \right]$$

$$= (f_{N-1}) \left[\log N - \log(f_{N-1}) \right] - \sqrt{N} \log(f_{N-1})$$

$$= (f_{N-1}) \left[\log N - \log(f_{N-1}) \right] - \log(f_{N-1})$$

$$15 \text{ Sunday} = (f_{N-1}) \left[\log N - \log(f_{N-1}) - \log(f_{N-1}) \right] + \log(f_{N-1})$$

$$= (f_{N-1}) \left[\log N - \log(f_{N-1})^2 \right] + \log(f_{N-1})$$

$$= \log \left[\frac{N}{(f_{N-1})^2} \right] + \log(f_{N-1})$$

NOV	M	T	W	F	S	S	M	T	F	S
2013	11	12	13	14	15	16	17	18	19	20

2013 = log

149 DECEMBER

19/12/2013

Monday 16

Trellis for CRF Viterbi path

Ops state \times state \times state



20/12/2013

A \cap B $P(A|B) = \frac{P(A \cap B)}{P(B)}$ Tuesday 17

B \cap A $P(B|A) = \frac{P(A \cap B)}{P(A)}$

$$P(A|B)P(B) = P(B|A)P(A)$$

$$\Rightarrow P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

$$[\text{23/12/2013}] (f_{N-1}) \left(\log(f_{N-1})^2 - \log(f_{N-1})^2 \right)$$

$$\log(f_{N-1})^2 - 2(f_{N-1}) \log(f_{N-1}) + \log(f_{N-1})^2$$

JAN	M	T	W	F	S	S	M	T	F	S
2014	6	7	8	9	10	11	12	13	14	15

DIARY 2013

②

Krishna iResearch

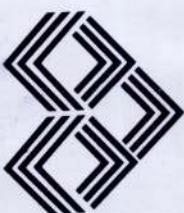
Open Source (AsFar, USB mod, VIRGO, King Cobra)
and Closed Source (SIMHA)

Design Notes and Implementation

Notes (and)

Academic Publications Notes

CONFIDENTIAL



इंडियन ऑवरसीज़ बङ्क
Indian Overseas Bank

Central Office: 763, Anna Salai, Chennai - 600 002. India

Ka. S. Srinivasan
29/12/2013

Wed. Anniversaries

FEB '13						
S	M	T	W	T	F	S
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28		

29/12/2013

Implication Graphs and
Convex Hulls

January 1
2013
Tuesday

If the implication graph is grown as a Random Growth Network (instead of a Dynamic Graph) which allows adding of new vertices and edges, at some point in time, the Proof path or Proof Geodesic between premise and conclusion is attained.

Convex Hulls of all possible Random growth graphs that contains the proof geodesic need to be found and minimum of all these Convex hulls containing proof geodesic should be of polynomial size in the length of proof geodesic.

K. Srinivasan
29/12/2013

PhD thesis proposal (Ka.Shrinivaasan)

Algorithms and Datastructures for Program Comprehension, Analysis and Optimization (with lowerbounds)

I would like to research on Algorithms/Datastructures and their lower bounds for some Program Comprehension, Analysis and Optimization problems. To start with I want to research on the following. There are many papers and theses already written on these which I have been reading. Hereby I have listed few points on proposed direction of research though they are in preliminary stages:

(*). Let us consider a program written in some high-level language like C/C++/Java. Many program slicers are available which would output the slices of the program. Thus a slice dependency graph can be built from the slices with the dependence relation as the edge. Each slice is a node in the slice dependence graph. Let us assign each a boolean variable $s(i)$ to each vertex i of this Slice dependence graph. The variable $s(i)$ takes value 1 if slice is executed , otherwise 0. Any path in Slice dependence graph corresponds to an execution path(trace). From the possible execution paths, a DNF formula can be obtained which is a disjunction of conjunction of boolean variables. This DNF corresponds to all possible slice paths a program can take. For example, $DNF(f) = (s1 . s3 . s5) + (s2 . s3 . s7) + \dots (s3 . s6 . s2)$ implies the disjunctions of all possible execution paths, $s1-s3-s5$, $s2-s3-s7$, $s3-s6-s2$ $Dnf(f)$ is satisfied if program takes one of the execution paths. Similarly a CNF(f) can also be written, which is satisfied if program does not take an execution path. Counting the number of execution paths in Slice dependency graph is #P-Complete (Valiant's result).

(*) Various properties of the Slice dependency graph can be tested. For example Clique of slice dependency graph corresponds to closely related portions of the code. Vertex cover of the slice dependency graph (set of vertices with atleast one edge incident) are the slices with high impact (a summary) on the program execution. Also Feedback vertex set of the graph yields set of slices whose removal leaves the slice dependency graph without cycles. Thus Feedback vertex set of Slice dependency graph is a good measure of subset of code which has high impact.

(*) Two programs can be compared for their similarity an oft-required functionality for anti-plagiarism tools. Graph isomorphism(a renumbering of vertices) can also be applied to two Slice dependency graphs of two different programs. Comparing two codebases for similarity reduces to Graph Isomorphism problem. This involves comparing each vertex(slice) on both the graphs and to see if renumbering of vertices yields the other graph.

(*) Two programs can be compared for similarity with the use of BK-Trees which use metric spaces to return closest matching set of strings to a given query string. The metric for the metric space can be the edit distance of two Slice dependency graphs.

(*) SATURN - Does Boolean encoding of program constructs and uses SAT solver. [This project made me to think on the following - What does it mean to say that a boolean encoding of a program is satisfied. SAT is NP-Complete and Program termination is Undecidable. A program terminates only if it has a satisfying assignment and finding a satisfying assignment to Boolean encoding of a program is NP-complete, which seems to be conflicting.]

(*) Program Comprehension - using boolean encoding as in SATURN to get the summary of a program which is human readable description of what a program does. This might also add a code search engine which takes a natural language query "Does the program do X?" comprehends the program and outputs the answer for this query in natural language.

(*) Program analysis with Binary Decision Diagram - We can construct a BDD from program slices with each slice being a boolean variable in the BDD. BDD compactly represents a boolean function on slice variables $s(i)$. $s(i)$ is 1 if slice i is executed and 0 if it is not. This BDD compactly represents a program.

(*) Program timeout algorithm 1 - In a multiprocessing system, there often arises a need to timeout processes which consume too much of time and possibly hang due to programming bug. So there is a need for an application level scheduler or a Timeout Manager. I have attached a document on a Timeout manager for transactions which I wrote in 2002 and did an invention disclosure in Sun Microsystems which can be modified for process timeout (Patent application has not been filed). This algorithm maintains an index which is equal to number of timeout thread executions a transaction survives and a datastructure which has both hashmap and linkedlist capabilities (it maps survival index to list of transactions with that survival index). The timeout thread decrements the survival index on each invocation through hashCode() and equals() adjustments

(*) Program timeout algorithm 2 - I also have an alternative algorithm for Timeout as follows which has complexity of $O(\text{timeout}/\text{timeout_thread_interval} + \text{max_length_of_process_list})$ per invocation of timeout thread. This does not use hashmaps but a simple vector of timeout entries as defined hereunder:

Timeout thread

```
-----
struct timeoutEntry
{
    int timeout;
    list<process*> processes;
}
```

```

vector<timeoutEntry> timeoutList;

while true
{
    wait for timeout interval
    cached_max_timeout_entry = NULL
    for i in timeoutList
    {
        i.timeout = i.timeout - timeoutInterval
        if i.timeout <= 0
            timeout all processes in i.processes
    }
}

```

When new process is created

```

-----
if(cached_max_timeout_entry == NULL)
{
    create a new timeout entry t
    t.timeout = timeout value from config
    push the process p to t.processes
    append t to timeoutList
    cached_max_timeout_entry = t
}
else
    push process p to cached_max_timeout_entry.processes

```

References

1. SATURN project - <http://saturn.stanford.edu>
2. Static detection of software errors – PhD thesis by Yichen Xie (SATURN)
3. Program Analysis with BDDDBDB – PhD thesis by John Whaley
4. Various sites on Program Comprehension
5. New Slicing Algorithms for Parallelizing Single-Threaded Programs – Intel Labs
6. Computation Slicing: Techniques and Theory – Neeraj Mittal and Vijay Garg
7. Various sites on Program Slicing and Compilers

24/03/2014

p = probability of good choice.

$(1-p)$ = complement above.

16-01-2014

January Thursday

population size 2^n

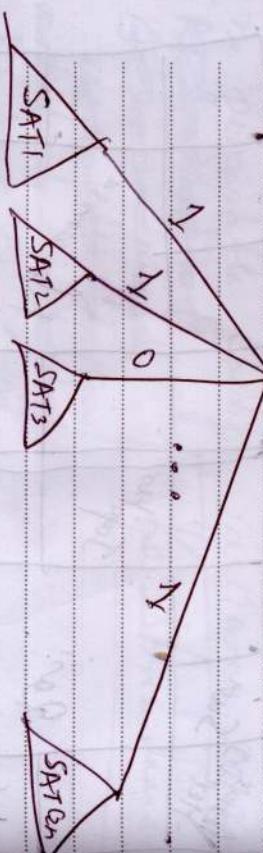
January

17 Friday

$$\begin{aligned} \text{Probability of atleast } n+1 \text{ people in voters} \\ \text{making good decision.} \\ P(\text{good } \geq n+1) = P(h+1) + P(h+2) + \dots + P(2^n) \\ 2^n \left[p^{n+1} (1-p)^{n-1} \right] + 2^n \left[p^{n+2} (1-p)^{n-2} \right] + \dots \\ = n+1 \left[p (1-p) \right] + 2^n \left[p^{n+1} (1-p)^{n-1} \right] + \dots \\ + \dots + 2^n \left[p^{2^n} (1-p)^{2^n} \right] \end{aligned}$$

When $p = 1$ above is $2^n \binom{n}{2^n} = \frac{2^n!}{2^n!} = 1$

NP Complete Democracy Circuit $\rightarrow 2^n / 0 / 1$
Corresponding MAJORITY with SAT circuit



Each voter has a SAT circuit which needs to be assigned by a candidate. Candidate is chosen if atleast

$(n+1)$ SATs are satisfied
 \rightarrow if all voters are non-motiv

above is an error-free democracy circuit which is NP-comp like.

Previous is for RHS of $P(\text{good})$ when all voters are perfect.

17-01-2014

January Friday 17

LHS of $P(\text{good})$ is pseudorandom choice which is in P and probability of good choice = no. of good voters / no. of total voters "one of the voters". If all voters are error-free pseudorandom choice is in P .

Example: 3 voters $n+1 = 2$

$\begin{matrix} 1 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{matrix}$ 8 possible
 good or bad voter patterns
 for 3 voters = $P(2) + P(3)$

$$\begin{matrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{matrix} \left(\binom{3}{2} + \binom{3}{3} \right) \times \frac{1}{2^3} \quad \text{if good & bad have equal prob.} \\ = \frac{3!}{2^3} + \frac{3!}{2^3} = 3 + 1 = \frac{4}{2^3} = \frac{1}{2}$$

~~non-uniformly~~
 If good and bad decisions are (uniquely) distributed some bit patterns in above should never occur.

Elections Campaign is the process of "Satisfying" the Voter circuit.

S	M	T	W	T	F	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

February 2014

S	M	T	W	T	F	S
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

SAT + Majority circuit is that a candidate can vote for himself (a real world election process). Thus with perfect zero error voting and pseudorandom choice there exists a algorithm for NP. Is there a counter example in complexity?

Philosophical implications of the Shor's reason

26/3/2014
thus is :

With more perfection "hard problems" become "easy". Thus becoming "zero-error" itself is by learning from mistakes and applying that.

19 Sunday

19-01-2014
Talent - Expression. Quick reminiscence of Expression: Natural ability & experience

Either problem definition is wrong or something missing in above. This would imply that irreversible functions are the only possibility (hard to invert) with zero-error and one-way functions cannot exist if everything is perfect in judgement.

December 2013

S	M	T	W	T	F	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

February 2014

S	M	T	W	T	F	S
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	

TAC 2010 - Update Summarization applying Interview Algorithm

Ka.Shrinivaasan
Chennai Mathematical Institute
shrinivas@cmi.ac.in



Motivation

- Is prestige based ranking perfect?
- Are there alternatives?
- Two judging traditions – majority voting and interactive – which is right? Subjective or objective?
- Can a document be analyzed independently to get its quality?

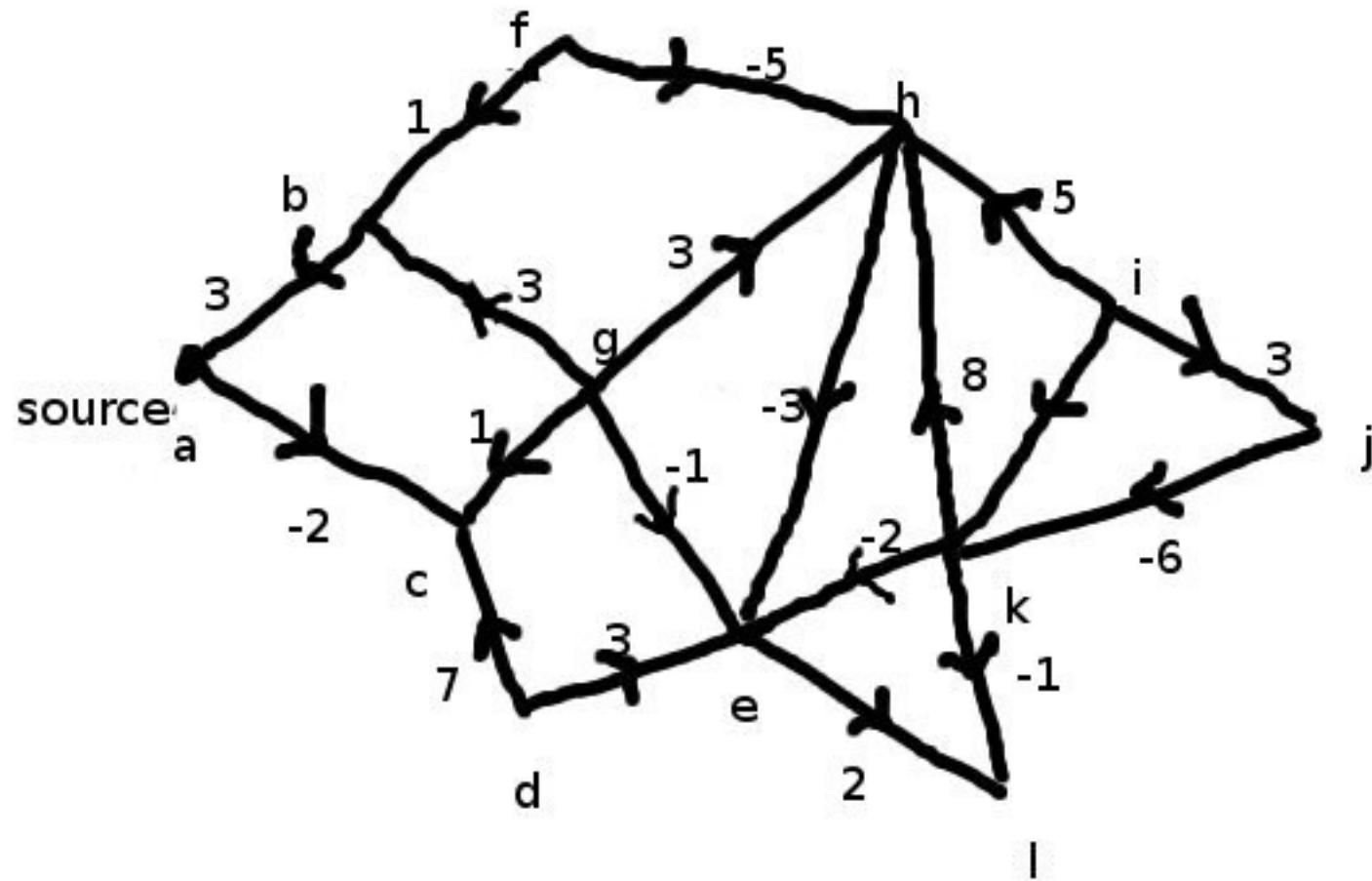
Three Algorithms ...

- **Citation Graph Maxflow and Path Lengths**
(uses Recursive gloss overlap for sentiment analysis - finding polarity)
- **Definition Graph Convergence (or)**
Generalized Recursive Gloss Overlap
- **Interview algorithm – applies either of above**
- **Application to Update summarization**

Part I - Directed Graph of Citations

- Merit = influence on future documents = citations
- Construct a directed graph of citations
- Weight of an edge (u,v) = No. Of citations of u by v (is this only way to weight?)
- Polarity of (u,v) = Sentiment Analysis of Citation Context – Positive or Negative
- Number of nodes in all paths of fixed length from source s is a measure of merit (might mislead)

Citation digraph - How it looks



Mincut/Maxflow of Citation DiGraph

- Get Maxflow from Ford-Fulkerson algorithm with each distinct vertex pair as (source, sink) – Capacity of the edge = weight of the edge
- Mincut of citation graph carries Maximum Flow of the concept from source document s - “most influenced by the source document s”
- Average Maxflow out of a source s, is thus a measure of merit of s ($= (\sum \text{mx}(s,t)) / (|V|-1)$)

Part II – Definition Graphs

- “Fruit”
- Evocative - What do we get reminded of after reading the above? (*plant, tree, sweet, taste, food, juice, result ...?*)
- Evocation WordNet

Human thought process and Definition Graphs

- Humans scan through the natural language text
- Relate the keywords – motivation behind WordNet
- **What distinguishes the merit of 2 documents X and Y? Grammatical correctness? No. Both X and Y written equally grammatically. Content and Complexity? Yes. How to measure?**

Recursive Understanding - An Example

- *Document: “Car race ends with flag waving”*
- What is “Car”? Car is an automobile
 - What is “automobile”? Fuel driven Machine
 - What is “Fuel”? Petroleum ...
- What is “race”? Race is ethnic group; contest
 - What is “contest”? Game
 - What is “Game” ? Play ...
- What is “end”? ...
- What is “flag”? ...
- What is “waving”? ...

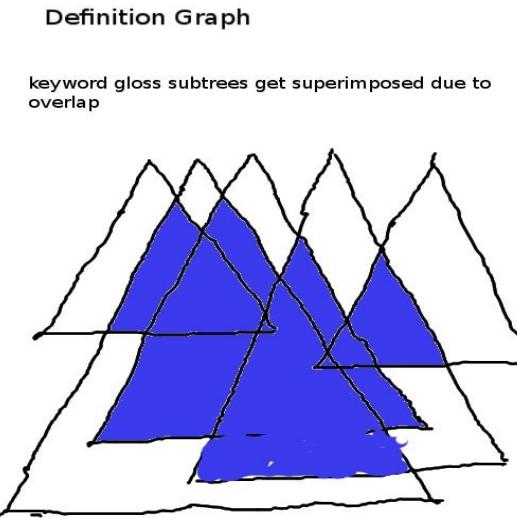
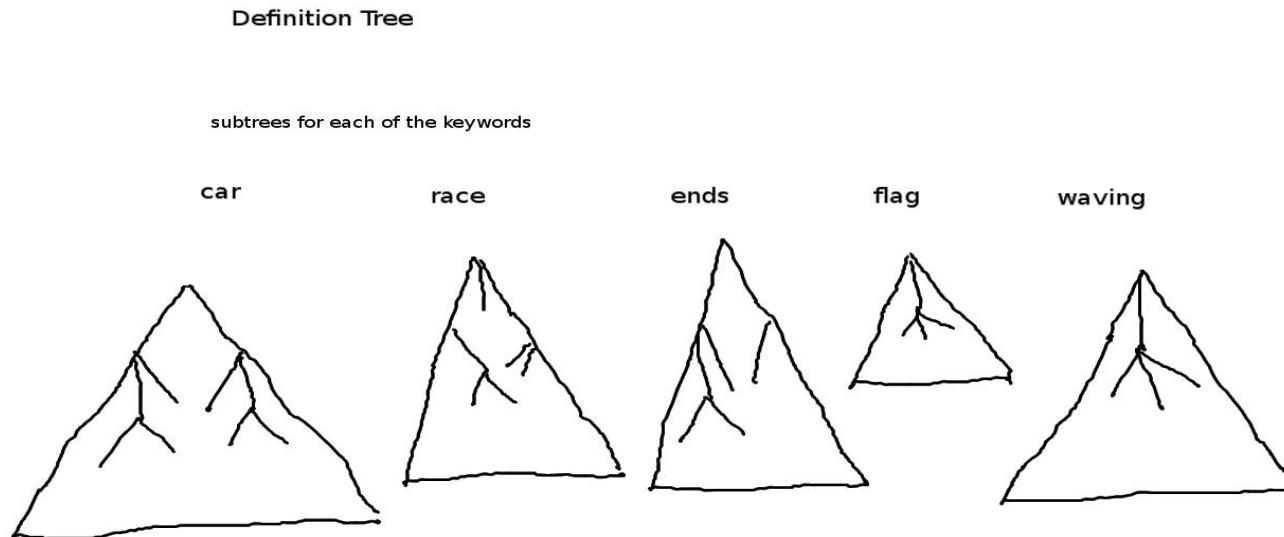
Definition Graph Convergence (or) Generalized Recursive Gloss Overlap

- Meaningfulness: “Meaningful” text has its keywords' Synsets within threshold WordNet distance (e.g Jiang-Conrath)
- WordNet relates words by relations - “is-a”, “has-a” etc., - SYNonymous SETs
- *Map a document to a **subgraph** of WordNet (Definition Trees/Graphs): $F(\text{Document}) = G(V, E)$*

Definition tree and Definition graph

- *DefinitionTree(keyword) = DefinitionTree(gkeyword1) DefinitionTree(gkeyword2) DefinitionTree(gkeyword3) ... DefinitionTree(gkeywordn)* where gkeyword1 through gkeywordn are in the gloss(keyword)
- N subtrees obtained above overlap to form a graph

Definition Tree and Graph - example



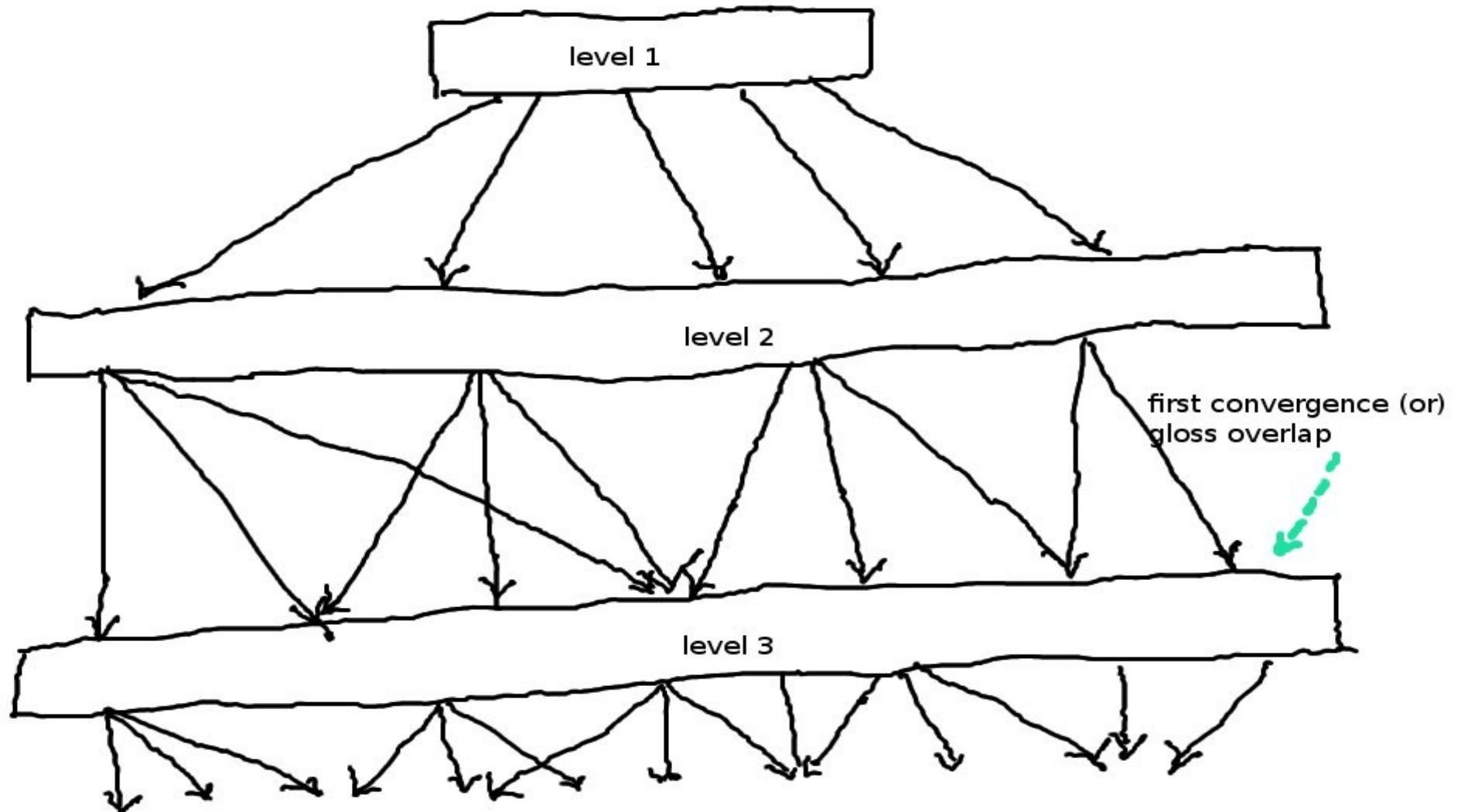
Properties of definition graph

- Definition graph is **multipartite**
- Difference in number of vertices in definition tree and definition graph = convergence factor
- Convergence factor is due to gloss overlap - indicator of relatedness
- Relatedness differentiates 2 documents
- We do not consider grammatical structure

Properties of Definition Graph(contd...)

- Multipartiteness - vertices are partitioned into sets; edges only amongst the sets – useful for preserving recursion level and multipartite-cliques
- Degrees of vertices can be thought of as “votes” for a “theme” keyword – **unsupervised text classifier**
- Context-sensitiveness still present - Word Sense Disambiguation is done during graph construction

Definition Multipartite Graph Visualised



Recursive Gloss Overlap algorithm

- 1) Get the document as input
- 2) `keywordsatthislevel = {keywords from the document through tf-idf filter (implementation uses 0.02)}`
- 3) `While (current_level < depth_required) {`
 - For each keyword from `keywordsatthislevel` lookup the **best matching definition(WSD) for the keyword** and add to a set of tokens in next level

Recursive Gloss Overlap algorithm(contd...)

- Remove common tokens (isomorphic nodes) with previous levels - an optimization
- Update the number of vertices(*unique tokens*), edges($(x,y) = 'y \text{ is in definition of } x'$) and relatedness (*linear overlap or quadratic overlap*)
- Update keywords at this level

Recursive Gloss Overlap algorithm(contd...)

} //end while

5) Output the *Intrinsic merit score* = $|\text{vertices}| * |\text{edges}| * |\text{relatedness}| / \text{first_convergence_level}$

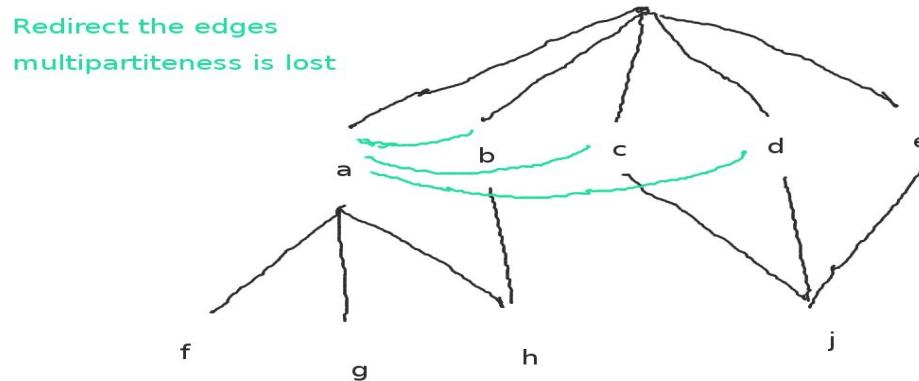
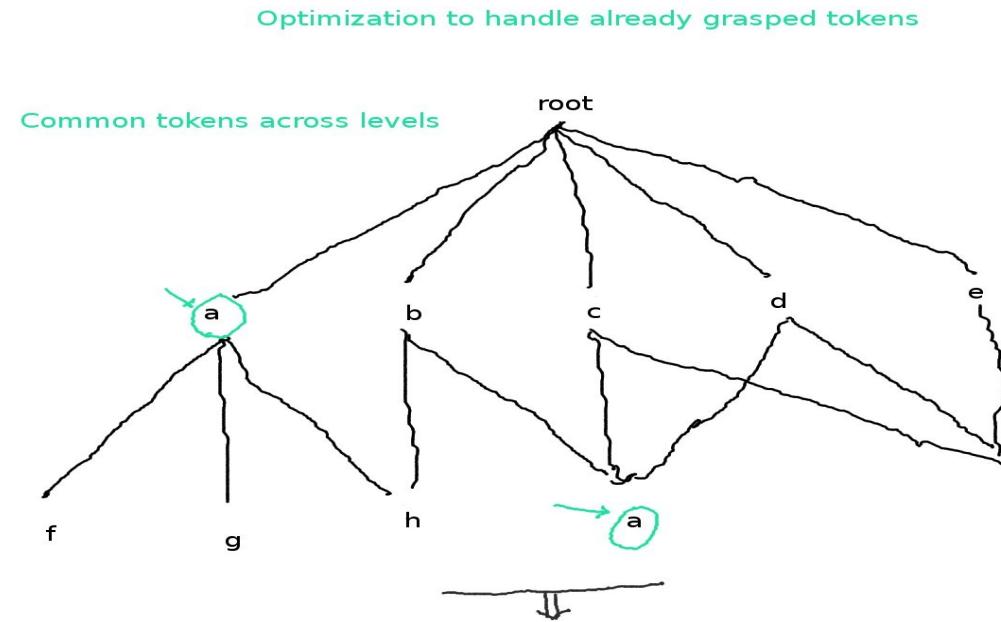
Where

a) Relatedness = *number of overlaps* (linear, also called as convergence factor) (or)

Relatedness = *number of overlapping parents* * *number of overlaps*^{**2} (quadratic)

b) First_convergence_level = level of first gloss overlap

Snapshot of Definition graph

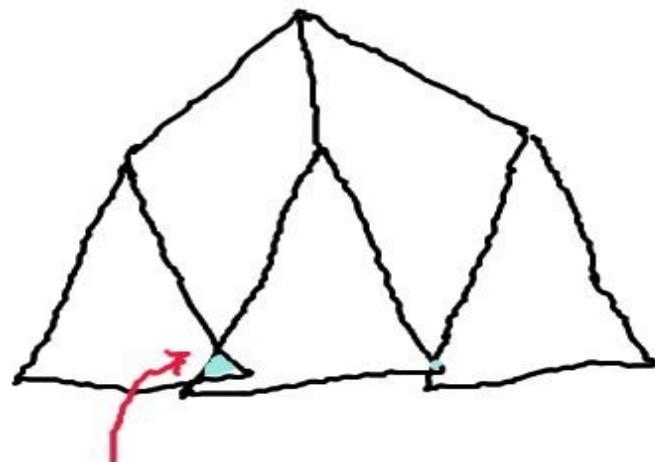


Intuition behind the intrinsic merit score

- vertices ~ knowledge represented by document
- edges ~ relationship among keywords (relation: 'x is in definition of y')
- relatedness ~ complexity quantified by overlap
- first_convergence_level ~ Mingling of definition subtrees
- Above suffice to quantify “meaningfulness” defined earlier (*proportional to V^*E^*R/f*)

Comparing two documents for merit

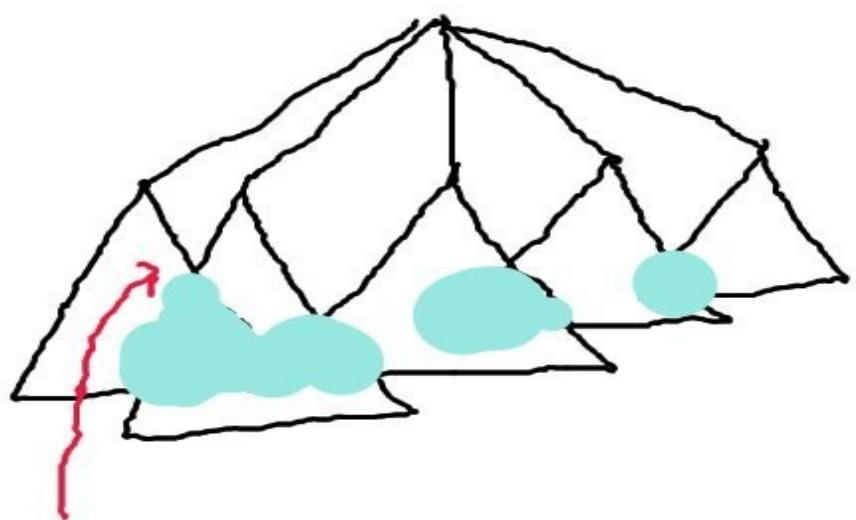
Document1 has less overlap



first convergence level = 5

Example: Car plies on sky

Document2 has more overlap



first convergence level = 2

Example: Cars and buses ply on road

BFS/DFS of definition graph

- Visiting all nodes of definition graph – $O(V+E)$
- But this does not take into account the relatedness
- Worst case size of definition graph is $O(x \square d)$
(where x is the average size of a keyword definition and d is the depth)
- For a meaningful document, overlaps bring down this to great extent – no exponential blowup- $O(V)$

Pros and Cons

- No False negatives, False positives exist
- Other ranking schemes can be derived from definition graph – based on graph connectedness, completeness etc.,
- Definition graph relies on relatedness of words instead of hyperlink graph – one more layer of abstraction – **all documents now become projected subgraphs of larger universal semantic graph (e.g WordNet)**

Running time of Recursive Gloss Overlap Algorithm

Worst case running time of Recursive gloss overlap is $O(E^*(V^2))$ where V is the number of vertices and E the edges of graph

This is sum total of the running times for gloss lookups, merging duplicate vertices and isomorphic nodes removal.

Application of Recursive gloss overlap to sentiment analysis

- Needed - SentiWordNet - gloss with quantified positivity/negativity score for a keyword
- Example: “*That movie was fantastic. Graphics was awesome*”
- Def Graph level 1: {movie:*motion picture*;+0.1, fantastic:*great*;+0.7, graphics:*software technique*;+0.05, awesome:*great*;+0.7}
- Polarity of Overlap – {great} with positivity score +0.7

Parallelizability of Recursive Gloss Overlap

- Def Graph construction parallelizable - set of tokens of each level broken into subsets
- Assign each subset to a processor (Map)
- Get the results of gloss lookup for subsets and merge them (Reduce)
- To do – Apply MapReduce framework to Recursive Gloss Overlap – E.g Needs a Hadoop cluster

Part III – Interview Algorithm

- Reference “interviews” the candidate – both are documents
- Candidate is inducted into reference if the interview score is above threshold
- Interview is less invasive compared to definition graph construction
- Tree/Graph of interviews can be built (transitive)
 - e.g x interviews (y,z), y interviews w, z interviews p

Interview Algorithm (contd...)

- Intrinsic merit of candidate measured by either a) Citation Digraph or b) Recursive gloss overlap algorithms
- Interview - a) **supervised** (reference Q&A available) or b) **unsupervised** (reference Q&A are computed from reference – 'Q's are keywords / 'A's are contexts)
- Interview is the set of tuples = $\{t(1), t(2), \dots, t(n)\}$
 $t(i) = (question, answer, expected_answer, score)$

Interview Algorithm (contd...)

- Total interview score = $\sum(t(i).score)$ (where
 $t(i).score = |\text{shingles}(\text{answer}) \cap \text{shingles}(\text{expected_answer})| / |\text{shingles}(\text{answer}) \cup \text{shingles}(\text{expected_answer})|$)
- Value addition = edit distance of
DefGraph(Reference) and DefGraph(Candidate)
(where $\text{EditDistance}(G, H) = |\text{edges added}| + |\text{edges removed}|$ to transform G into H)
- Final score = $w1 * \text{intrinsic_merit} + w2 * \text{interview_Q\&A_score} + w3 * \text{value_addition}$, where
 $w1, w2$ and $w3$ are weights

Application of Interview Algorithm to TAC 2010 Update summarization

- Split each dataset into candidate and reference
- Go through the Interview algorithm and get scores for candidates (for intrinsic merit, recursive gloss overlap was used)
- Choose the best candidate and update the summary after sentence scoring (sentence score = sum of tf-idf values of constituent terms)
[NOTE: Only 25 (out of 92) datasets were tried due to hardware issue]

25 February
Tuesday

28/8/2014 ①

25-02-2014

Theta Zeta Function

$$\text{Cycle of length } p: \frac{1}{(1 - \frac{1}{q^p})} \quad q \text{-regular graph.}$$

$$\frac{1}{s - \frac{1}{(1 - \frac{1}{q^p})}} = RZF \left(1 + \frac{1}{2^s}\right) \left(1 - \frac{1}{2^s}\right)$$

$$Z(s) \left(\det \left(I - A_2^{-s} + 2^{1-2s} I \right) \right) = \left(1 - \frac{1}{2}\right)^{1|V|-|E|}$$

$$Z_2(s) \left(\det \left(I - A_2^{-s} + 3^{1-2s} I \right) \right) = \left(1 - \frac{1}{3}\right)^{1|V|-|E|}$$

$$Z_3(s) \left(\det \left(I - A_3^{-s} + 5^{1-2s} I \right) \right) = \left(1 - \frac{1}{5}\right)^{1|V|-|E|}$$

$$Z_4(s) \left(\det \left(I - A_7^{-s} + 7^{1-2s} I \right) \right) = \left(1 - \frac{1}{7}\right)^{1|V|-|E|}$$

$$Z_5(s) \left(\det \left(I - A_{11}^{-s} + 11^{1-2s} I \right) \right) = \left(1 - \frac{1}{11}\right)^{1|V|-|E|}$$

$$\vdots$$

$$Z_{27}(s) \left(\det \left(I - A_{27}^{-s} + 27^{1-2s} I \right) \right) = \left(1 - \frac{1}{27}\right)^{1|V|-|E|}$$

$$Z_{27}(s) \left(\det \left(I - A_{27}^{-s} + 3^{1-2s} I \right) \right) = \left(1 - \frac{1}{3}\right)^{1|V|-|E|}$$

$$Z_{27}(s) \left(\det \left(I - A_{27}^{-s} + 5^{1-2s} I \right) \right) = \left(1 - \frac{1}{5}\right)^{1|V|-|E|}$$

$$Z_{27}(s) \left(\det \left(I - A_{27}^{-s} + 7^{1-2s} I \right) \right) = \left(1 - \frac{1}{7}\right)^{1|V|-|E|}$$

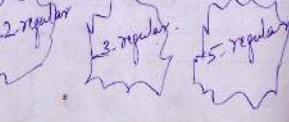
$$Z_{27}(s) \left(\det \left(I - A_{27}^{-s} + 11^{1-2s} I \right) \right) = \left(1 - \frac{1}{11}\right)^{1|V|-|E|}$$

$$\vdots$$

$$Z_{27}(s) \left(\det \left(I - A_{27}^{-s} + 27^{1-2s} I \right) \right) = \left(1 - \frac{1}{27}\right)^{1|V|-|E|}$$

January 2014

S	M	T	W	T	F	S
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	

27 February
Thursday

of Adjacency matrix of

27-02-2014

prime regularity.

$$\left(q^s + q^{1-s} \right) = \lambda$$

$$q^{a+ib} + q^{1-a-ib}$$

$$q^{a+ib} + q^{1-a-ib} = \frac{q^{a-b} + q^{b-a}}{q^{a+ib}}$$

$$\frac{q^{2s} + q^{-2s}}{q^s} = \lambda$$

Adjacency matrix of n vertices and q -regular graph

$$n \times n \begin{bmatrix} 0 & 1 & 1 & \dots & 1 & 0 & q & 1 & \dots \\ 1 & 0 & 1 & \dots & \dots & 0 & q & 1 & \dots \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots \\ q & 1 & 1 & \dots & 1 & 0 & q & 1 & \dots \\ 1 & 0 & 1 & \dots & \dots & 0 & q & 1 & \dots \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots \\ q & 1 & 1 & \dots & 1 & 0 & q & 1 & \dots \end{bmatrix}$$

January 2014

S	M	T	W	T	F	S
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	

$$q^s, q^{1-s}$$

28/8/2014 ②

26 February
Wednesday

$$Z(s) \left(\det \left(I - A_2^{-s} + 2^{1-2s} I \right) \right) = \left(1 - \frac{1}{2}\right)^{1|V|-|E|}$$

$$Z_2(s) \left(\det \left(I - A_2^{-s} + 3^{1-2s} I \right) \right) = \left(1 - \frac{1}{3}\right)^{1|V|-|E|}$$

$$Z_3(s) \left(\det \left(I - A_3^{-s} + 5^{1-2s} I \right) \right) = \left(1 - \frac{1}{5}\right)^{1|V|-|E|}$$

$$Z_4(s) \left(\det \left(I - A_7^{-s} + 7^{1-2s} I \right) \right) = \left(1 - \frac{1}{7}\right)^{1|V|-|E|}$$

$$Z_5(s) \left(\det \left(I - A_{11}^{-s} + 11^{1-2s} I \right) \right) = \left(1 - \frac{1}{11}\right)^{1|V|-|E|}$$

$$\vdots$$

$$Z_{27}(s) \left(\det \left(I - A_{27}^{-s} + 27^{1-2s} I \right) \right) = \left(1 - \frac{1}{27}\right)^{1|V|-|E|}$$

$$Z_{27}(s) \left(\det \left(I - A_{27}^{-s} + 3^{1-2s} I \right) \right) = \left(1 - \frac{1}{3}\right)^{1|V|-|E|}$$

$$Z_{27}(s) \left(\det \left(I - A_{27}^{-s} + 5^{1-2s} I \right) \right) = \left(1 - \frac{1}{5}\right)^{1|V|-|E|}$$

$$Z_{27}(s) \left(\det \left(I - A_{27}^{-s} + 7^{1-2s} I \right) \right) = \left(1 - \frac{1}{7}\right)^{1|V|-|E|}$$

$$Z_{27}(s) \left(\det \left(I - A_{27}^{-s} + 11^{1-2s} I \right) \right) = \left(1 - \frac{1}{11}\right)^{1|V|-|E|}$$

$$\vdots$$

$$Z_{27}(s) \left(\det \left(I - A_{27}^{-s} + 27^{1-2s} I \right) \right) = \left(1 - \frac{1}{27}\right)^{1|V|-|E|}$$

March 2014

S	M	T	W	T	F	S
30	31					1
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

28 February
Friday

28

28-02-2014

Amavasya

$$\begin{bmatrix} 2 & b & c \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix} - \lambda \begin{bmatrix} \lambda & 0 & 0 \\ 0 & \lambda & 0 \\ 0 & 0 & \lambda \end{bmatrix}$$

$$= \begin{bmatrix} -\lambda & 1 & 1 \\ 1 & -\lambda & 1 \\ 1 & 1 & -\lambda \end{bmatrix} = -\lambda \left(\lambda^2 - 1 \right) - \left(-\lambda - 1 \right)$$

$$+ \left(1 + \lambda \right).$$

$$-\lambda^3 + \lambda + \lambda + 1 + \lambda + 1 = 0.$$

$$-\lambda^3 + 3\lambda + 2 = 0 \quad s \leftarrow 1$$

$$\lambda^3 - 3\lambda - 2 = 0 \quad 2 = 2 + 2$$

$$\lambda^3 - \lambda - 2\lambda - 2 = 0$$

$$\lambda^2(\lambda - 1) - 2(\lambda + 1) = 0.$$

$$\lambda = 2 \quad 8 - 6 - 2 = 0$$

$$8 - 8 = 0$$

$$(\lambda - 2)(\lambda^2 -$$

March 2014

S	M	T	W	T	F	S
30	31					1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29

Then

6 March Thursday $q^s + q^{1-s} = q$ (X) 06-03-2014

$$q^{s-1} + q^{-s} = 1$$

$$q^{2s-1} + 1 = q^s$$

$$q^{2s-1} - q^s + 1 = 0.$$

$$s = a + ib$$

$$q^{2a-1+2ib} - q^{a+ib} + 1 = 0.$$

Cont. (1) if $a \neq \frac{1}{2}$

$$q^s = y$$

$$(y) - y + 1 = 0$$

$$y^2 - 2y + 1 = 0$$

February 2014

S	M	T	W	T	F	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28

7 March Friday $y = \frac{q \pm \sqrt{q^2 - 4q}}{2}$ For $q = 2$
2 regular graph $2 \pm \sqrt{4-8}$
 $2 = q$
 $2^s = 1 \pm i$
 $2^{a+ib} = 1 \pm i$
 $2^{ib} = \frac{1}{\sqrt{2}} \pm \frac{i}{\sqrt{2}}$
 $e^{ib \ln 2} = \frac{1}{\sqrt{2}} + i \frac{\ln 2}{\sqrt{2}}$
 $e^{ib \ln 2} = \frac{1}{\sqrt{2}} + i \frac{1}{\sqrt{2}}$

8 March Saturday $q^a q^{ib} = \frac{q \pm \sqrt{q^2 - 4q}}{2}$ (X) 08-03-2014

Cont. (2) If $a = \frac{1}{2}$

$$q^a q^{ib} = \sqrt{q} \cdot \sqrt{q} = \sqrt{q^2 - 4} \quad \text{if } a = \frac{1}{2}$$

$$q^{ib} = \frac{\sqrt{q} \pm \sqrt{q^2 - 4}}{2}$$

10 March Sunday $e^{ib \log q} = \frac{\sqrt{q} \pm \sqrt{q^2 - 4}}{2}$ 09-03-2014

$$e^{ib \log q} = \frac{\sqrt{q} \pm \sqrt{q^2 - 4}}{2}$$

$$\cos b \log q + i \sin b \log q = \frac{\sqrt{q} \pm \sqrt{q^2 - 4}}{2}$$

$$b \log q = 0. \text{ but } q \neq 0 \quad \text{Im}(q) = 0$$

$$\Rightarrow a \neq \frac{1}{2} \quad b \neq 0$$

Is RH false?

February 2014

S	M	T	W	T	F	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28

10 March Monday $y = \frac{q \pm \sqrt{q^2 - 4q}}{2}$ For $q = 2$
2 regular graph $2 \pm \sqrt{4-8}$
 $2 = q$
 $2^s = 1 \pm i$
 $2^{a+ib} = 1 \pm i$
 $2^{ib} = \frac{1}{\sqrt{2}} \pm \frac{i}{\sqrt{2}}$
 $e^{ib \ln 2} = \frac{1}{\sqrt{2}} + i \frac{\ln 2}{\sqrt{2}}$
 $e^{ib \ln 2} = \frac{1}{\sqrt{2}} + i \frac{1}{\sqrt{2}}$

April 2014

S	M	T	W	T	F	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30					

steps in previous proof (12) if it proves

10-03-2014

- Construct infinitely many prime-regular graphs (X)
- Using Ihara Zeta functions for all these graphs equal to Riemann Zeta function (page 122)
- Zeros of RZF in RHS (page 2) should either be 3.1) zeros of $Z_p(s)$ - the Ihara Zeta functions of the prime regular graphs (or) 3.2) ~~not~~ the eigen values of the product of Ihara identities for all graphs
- 3.1) implies graph is Ramanujan (3.2) is for non-Ramanujan graphs and maximum eigen value of p -regular graph is p . Thus $q^s + q^{1-s} = q$ (eigenvalue degree) ~~extending~~
- Assuming $\text{Re}(s) = \frac{1}{2} = a$, there seems to be a contradiction as in page 12 in deriving $\text{Im}(s) = b$. Probably hints at RH being false for some s , $a \neq \frac{1}{2}$

April 2014

S	M	T	W	T	F	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30					

Ka Shrinivasan

To verify:

11 March
Tuesday

11

13

11-03-2014

1) Can a p -regular graph have eigenvalues other than p ?

2) Is it right to equate q^{ib} to

$$\frac{\sqrt{q^2 - 4}}{2}$$

3) Can q^{ib} be real

if the eigenvalue is less than q
for q regular graphs

$$q^s = \frac{q^2 - qg + 1}{2g} + \frac{1}{2}g$$

$$\text{If } s = \frac{1}{2} + ib$$

$$q^s = \sqrt{q^2 - q^2 b^2} = q \pm \sqrt{q^2 - 4b^2}$$

$$q^s = q \pm \sqrt{q^2 - 4b^2}$$

February 2014

S	M	T	W	T	F	S
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29

$$e^{ib\log q} = \cos b\log q + i\sin b\log q = q^{ib} \quad (14)$$

12-03-2014

$$= q \pm \sqrt{q^2 - 4b^2}$$

March
Wednesday

12

$2\sqrt{q}$

$$\cos b\log q = q \pm \sqrt{q^2 - 4b^2}$$

$2\sqrt{q}$

$$\sin b\log q = 0$$

$\Rightarrow b = 0 \Rightarrow \log q = 0$ a contradiction
thus $\operatorname{Re}(s)$ cannot be $\frac{1}{2}$ for all zeroes
of RZF

K. Srinivasan

30/8/2014

April 2014						
S	M	T	W	T	F	S
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30		

K.Srinivasan

(also known as : SrinivasanKannan, Ka.Shrinivaasan, ShrinivasKannan)

About Myself

Worked for various IT majors and startups for 19 years from 1999 and did Doctoral research in theoretical computer science till 2011. Presently working on a non-funded and not-for-profit opensource initiative and pursuing independent academic research.

Open Source Initiative - Krishna iResearch – 2003-present

Presently working individually on research and development of non-commercial, non-funded open source copyleft dual-licensed initiative (no team or sponsor involved) - cloud, bigdata analytics and machine learning augmented new Linux Kernel fork-off:

NeuronRain Research - http://sourceforge.net/users/ka_shrinivaasan

NeuronRain Enterprise - <https://github.com/shrinivaasanka/>

NeuronRain Documentation, FAQ and Licensing - <http://neuronrain-documentation.readthedocs.io/en/latest/>

Previous repositories include an open learning free courseware (https://github.com/shrinivaasanka/Grafit/tree/master/course_material) and implementations of publications and drafts in <https://sites.google.com/site/kuja27/>

Research Interests-Theory and Engineering

Computational Number Theory Algorithms, Computational Geometry, Computational Linguistics and Natural Language Processing, Computational Economics, Algorithms for Massive Datasets and Machine Learning, Intrinsic Fitness/Merit, Computational Complexity of Majority Voting, Satisfiability and related, Pseudorandomness, Program Analysis

Research Publications

- Decidability of Complementation - 2011 - <http://arxiv.org/abs/1106.4102>
- Algorithms for Intrinsic Merit - 2010 - <http://arxiv.org/abs/1006.4458>
- NIST TAC 2010 version of Algorithms for Intrinsic Merit -
http://www.nist.gov/tac/publications/2010/participant.papers/CMI_IIT.proceedings.pdf

Independently researched draft publications – from 2012 - <https://sites.google.com/site/kuja27/>

Advisors/Co-Authors:

- Google Scholar - <https://scholar.google.co.in/citations?user=eLZY7CIAAAJ&hl=en>
- DBLP - <http://dblp.dagstuhl.de/pers/hd/s/Shrinivaasan:Ka>
- CMI - <https://www.cmi.ac.in/people/alumni-profile.php?id=shrinivas>

Publication Drafts - Unguided and Unreviewed – 2012-present

Independent academic research publication drafts expanded on previous publications in

<https://sites.google.com/site/kuja27/> delve into Complexity Theoretic Analysis of Non-majority and Majority Social Choice, Pseudorandomness, Goodness of Voting and Condorcet Jury Theorem, Complement Function, Ramsey coloring of sequences, Diophantine Analysis, Riemann Zeta Function, Hypergeometric Functions, Graph theoretic/Computational linguistic/Interview Intrinsic Merit/Fitness and Experiential Learning in the context of WWW (mostly text analysis) and Social networks, Flow Market Equilibrium and Merit Equilibrium, Neural Networks and Deep Learning, Quantum mechanics and Intrinsic Fitness, Hash Functions, Integer Partitions, Space filling/Tiling/Packing, Satisfiability (CNFSAT and QBF-SAT), Linear and Convex-Concave Programming, Computational Geometric Integer Factoring and Connections amongst them etc., . These conceptual relations are described in NeuronRain FAQ: <http://neuronrain-documentation.readthedocs.io/en/latest/>

Statement of Research

Ka.Shrinivaasan, M.Sc(CS), Chennai Mathematical Institute (CMI)
(shrinivas@cmi.ac.in)

May 29, 2010

1 Research interests

Computational number theory, Graph and Combinatorial Algorithms (problems in <http://www2.research.att.com/~dsj/nsflist.html>), Program analysis and Verification, Distributed Algorithms and Complexity

2 Factorization

2.1 Description

This problem talks about factoring a composite integer z into two integers x and y (both possibly primes).

2.2 Relevance and Current status

All known algorithms for factorization take exponential time in the length of input integer. This problem is a good example of search vs decision paradox. Deciding whether a number is prime is in P by AKS algorithm (polynomial in length of input bits) but searching for the factors is still elusive and no polynomial algorithm has been found yet except for Shor's algorithm for Quantum computers (in Bounded Quantum Error Polynomial - BQP). Decision version of Factorization lies in intersection of P and coNP. Best known algorithm for factorization is the General number field sieve algorithm which takes time

$$O(\exp[(64/9 * \log n)^a * (\log \log n)^b]) \quad (1)$$

where $a = 0.33$ and $b = 0.66$

2.3 Methodologies that might be required to find a poly-time algorithm

Randomization might help in finding a polynomial time algorithm though not better - Since factorization is in BQP, it will be reasonable to ask if it is in BPP (Bounded error probabilistic polynomial). Victor Shoup's book discusses

one such probabilistic polytime algorithm. Since it is widely believed that $BPP = P$ (i.e randomization does not achieve extensive speed up), if factoring is in BPP then it should be in P .

3 Edge partitioning a planar graph into two outerplanar graphs

3.1 Description

This problem talks about edge partitioning a planar graph into two outerplanar graphs

3.2 Relevance and Current status

Partitioning a graph has many applications in electronics. This problem was discussed as part of Graph Theory course in M.Sc and an algorithmic solution for the dual of the above problem was tried out in endsem exam (sketch given below). Best known result is discussed in paper "Edge Partition of Planar Graphs into Two Outerplanar Graphs" by Daniel Gonçalves (LaBRI, U.M.R. 5800, Université Bordeaux 1, 351, cours de la Libération 33405 Talence Cedex, France)

3.3 Methodologies that might be required

Given the dual (3-regular face graph of triangulated planar graph)

- Given the edge set E of the graph G , partition E into two sets E_1 and E_2 of size n and n if $|E|$ is even or two sets of size $n - 1$ and $n + 1$ if $|E|$ is odd such that each vertex is listed in both sets.
- Merge two paths in E_i if one of the endpoints of paths match and no cycle is formed.
- Repeat previous step till no new path is formed by the merger

Writeup on research interests and some results

Srinivasan Kannan (alias) Ka.Shrinivaasan (alias) Shrinivas Kannan
Ph: 9789346927, 9003082186, 9791165980
Krishna iResearch Open Source Products Profiles:
http://sourceforge.net/users/ka_shrinivaasan,
Personal website(research): <https://sites.google.com/site/kuja27/>
ZODIAC DATASOFT: <https://github.com/shrinivaasanka/ZodiacDatasoft>
emails: ka.shrinivaasan@gmail.com, shrinivas.kannan@gmail.com,
kashrinivaasan@live.com

My research interests are:

- Hash Functions and Datastructures (bloom filters and other hashing algorithms)
- Satisfiability and Majority Voting (majority circuits and SAT)
- Graph algorithms (Expander graphs, Random graphs, Dynamic graphs, Graph spectra)
- Parallel RAM algorithms for sorting and merging sorted lists
- Circuit lowerbounds
- Probabilistic, Approximate, Correct Learning and Learning boolean functions from a dataset
- Computational Number Theory and Geometry
- Application of Graph algorithms to machine learning problems viz., Document Comparison and Ranking without Link graphs, Summarization, etc.,
- Non-statistical Graphical inference models
- Program (Software) Analysis, application of graph reachability and machine learning to Program analysis and debugging

During past 2 years and more I am privately researching on some complexity related problems (atleast as directions for research) though not officially as a PhD student. I have uploaded my research drafts in my website: <https://sites.google.com/site/kuja27/>. This mail is about few probable results in these drafts. These are my ongoing private research only and might have errors:

1. A Computational Geometric Algorithm for Factorization – Discrete Hyperbolic Factorization

Computational geometric factorization algorithm that uses hyperbola to factorize in parallel time using Parallel RAM NC circuits:

Link to the above parallel factorization draft PDF (with lot of handwritten illustrations also) is at:
http://sourceforge.net/projects/acadpdrafts/files/DiscreteHyperbolicPolylogarithmicSieveForIntegerFactorization_PRAM_TileMergeAndSearch_And_Stirling_Upperbound.pdf/download

and Parallel RAM implementation handwritten notes with illustrations are at (requires a tile_id due to shuffling after parallel merge and sort of the tiles):

<http://sourceforge.net/p/asfer/code/HEAD/tree/ImplementationDesignNotesForDiscreteHyperbolicFactorizationInPRAM.jpg>.

TeX version of the draft is at:

http://sourceforge.net/projects/acadpdrafts/files/DiscreteHyperbolicPolylogarithmicSieveForIntegerFactorization_PRAM_TileMergeAndSearch_And_Stirling_Upperbound.tex/download

A recent version with Parallel RAM to NC reduction and Input size references is at:

http://sourceforge.net/projects/acadpdrafts/files/DiscreteHyperbolicPolylogarithmicSieveForIntegerFactorization_PRAM_TileMergeAndSearch_And_Stirling_Upperbound_updateddraft.pdf/download

http://sourceforge.net/projects/acadpdrafts/files/DiscreteHyperbolicPolylogarithmicSieveForIntegerFactorization_PRAM_TileMergeAndSearch_And_Stirling_Upperbound_updateddraft.tex/download

Above first discretizes or tessellates a hyperbolic curve ($xy = <\text{some integer}>$) and uses Binary Search, Interpolation Search and All Nearest Smaller Values parallel RAM algorithm (http://en.wikipedia.org/wiki/All_nearest_smaller_values#CITEREFBerkmanSchieberVishkin1993) on a sorted list of integers which are points on merged discretized hyperbolic tiles to find the factors in $O(\log N * \log N)$ polylog time using polynomial processors in PRAM model and thus in NC.

The Sequential version of the above factorization algorithm has been implemented already and is working

(at: <http://sourceforge.net/p/asfer/code/HEAD/tree/cpp-src/miscellaneous/DiscreteHyperbolicFactorizationUpperbound.cpp>).

Sample result logs for sequential implementation are at: http://sourceforge.net/p/asfer/code/HEAD/tree/cpp-src/miscellaneous/DiscreteHyperbolicFactorization_FactorsOf_4189998_Screenshot_201311-08_103903.png.

(Older PDF and TeX versions of the above are in the same url -
<https://sites.google.com/site/kuja27>)

2. Majority Voting and Pseudorandom Choice

This set of drafts is about even more significant problem of containment of P in NP. Infact this has origins to a probability series I was working in 2006 about relative merits of Majority Voting and Pseudorandom choice (i.e Randomly chosen or majority voted which is better). I did my Master's thesis partly using it in 2010 followed by publication in TAC 2010. They are available at:

2.1 <http://arxiv.org/abs/1006.4458> (Initial sections mention about this series) (2010)

2.2 http://www.nist.gov/tac/publications/2010/participant.papers/CMI_IIT.proceedings.pdf (Initial sections mention about this series) (2010-11)

2.3

https://sites.google.com/site/kuja27/DocumentSummarization_using_SpectralGraphTheory_RGOGraph_2014.pdf?attredirects=0&d=1 (2014)

This series and probability of good choice defined in:

https://sites.google.com/site/kuja27/CircuitForComputingErrorProbabilityOfMajorityVoting_2014.pdf?attredirects=0&d=1

The computation of series is at: http://sourceforge.net/p/asfer/code/HEAD/tree/cpp-src/miscellaneous/pgood_batch_sum.cpp and requires hypergeometric functions in general case. But it converges in special cases of $p=1$ and $p=0.5$ (i.e Probability of good choice for each individual voter is uniformly 1 or 0.5) to 1 and 0.5 both sides pseudorandom choice and majority voting.

From 2012 onwards, I thought of extending the above graduate thesis into a PhD level dissertation problem and wrote a set of drafts during past 2 years which again surprisingly gave me a result $P=NP$ if there is a perfect judgement i.e With zeroerror $P=NP$ (and) $P \neq NP$ is undecidable if existence of Perfect Voter with zeroerror is undecidable by constructing a circuit for the above Majority Voting and Pseudorandom Choice series. Because of the enormity of this, I am still verifying it. The series convergence is also illustrated in
<http://sourceforge.net/p/asfer/code/HEAD/tree/cpp-src/miscellaneous/MajorityVotingErrorProbabilityConvergence.JPG>

I also felt that it has some connections to Arrow's Theorem - A Proof of Generalization of Arrow's theorem to infinite number of candidates (if decidable) implies $P \neq NP$ due to the above series. Point 10 in <http://sourceforge.net/p/asfer/code/HEAD/tree/AstroInferDesign.txt> describes this.

Following are the set of drafts and handwritten notes on the above (these are all connected in someway to the above series):

2.4 - Equating Majority Voting and Pseudorandom Choice
<https://sites.google.com/site/kuja27/LowerBoundsForMajorityVotingPseudorandomChoice.pdf?attredirects=0>

2.5 - Above Series Derivation
https://sites.google.com/site/kuja27/CircuitForComputingErrorProbabilityOfMajorityVoting_2014.pdf?attredirects=0&d=1

2.6 - Axiom of Choice and Majority Voting
https://sites.google.com/site/kuja27/IndepthAnalysisOfVariantOfMajorityVotingwithZFAOC_2014.pdf?attredirects=0

2.7 - A PRG for Pseudorandom Choice using Chaos Theory Attractors
<https://sites.google.com/site/kuja27/ChaoticPRG.pdf?attredirects=0>

2.8 - Interview Algorithm and Interactive Proof
<https://sites.google.com/site/kuja27/InterviewAlgorithmInPSPACE.pdf?attredirects=0>

2.9 - Equivalence of Integer Partitions and Hash Functions
https://sites.google.com/site/kuja27/IntegerPartitionAndHashFunctions_2014.pdf?attredirects=0&d=1. (This also gives an alternative proof of NP completeness of multipartisan democracy using reduction from restricted partition problem Money Changing Problem and Schur theorem for partitions)

2.10 - A gadget for randomized space filling to simulate some natural phenomena and LP for it -
<https://sites.google.com/site/kuja27/Analysis%20of%20a%20Randomized%20Space%20Filling%20Algorithm%20and%20its%20Linear%20Program%20Formulation.pdf?attredirects=0>

2.11 – Arrow's Theorem , Circuit for Democracy and P versus NP
https://sites.google.com/site/kuja27/CircuitsForDemocracyAndPseudorandomChoice_and_PVsNP.pdf?attredirects=0&d=1

Handwritten notes:

2.12 - Philosophical analysis of Democracy circuit and Pseudorandom choice
https://sites.google.com/site/kuja27/PhilosophicalAnalysisOfDemocracyCircuitAndPRGChoice_20140326.pdf?attredirects=0

2.13 – Handwritten notes on Schur Theorem, Restricted Partitions and Hash Functions, NP-Completeness of MultiPartisan Majority Voting
https://sites.google.com/site/kuja27/SchurTheoremMCPAndDistinctPartitions_20140417.pdf?attredirects=0

2.14 - An experimental theory of Convex Hull of the logical implication graph and Perfect Voter problem -
https://sites.google.com/site/kuja27/ImplicationRandomGraphConvexHullsAndPerfectVoterProblem_20140111.pdf?attredirects=0&d=1

2.15 - Experimental Notes on Logical Implication Graphs:
An experimental logical implication graph model for P and NP http://sourceforge.net/projects/acadpdrafts/files/ImplicationGraphsPGoodEquationAndPNotEqualToNPQuestion_

excerpts.pdf/download

Basic idea is to construct a Majority with SAT circuit as inputs and each SAT input is a voter who needs to be satisfied. Thus Majority voting or Democracy is NPComplete, but the Pseudorandom generator is in P or NC. In the above series if both LHS(Pseudorandom choice) and RHS(Majority voting) are 100% (i.e zero error) as shown in above convergence, then there is a polytime algorithm (pseudorandom choice) to the NP problem of Majority voting which is a counterexample. One-wayfunctions exist when the series does not converge and do not exist when series converges to 100%.

Thus if $P \neq NP$ then there cannot be a perfection. Moreover, finding even a single perfect voter who never makes an error itself looks to be intractable. Infact this circuit requires all voter SATs to be perfect for the series to converge to 100% thereby placing a tighter restriction than mere intractability. Thus above is not against what is widely believed (i.e P is properly contained in NP). Rather above gives 3 possibilities – when P can equal NP , when it cannot and whether question itself is decidable for infinite case.

(Older PDF and TeX versions of the above are in the same url -
<https://sites.google.com/site/kuja27>)

3.Circuit for Complement Functions – special case for Riemann Zeta Function

A non-uniform boolean circuit and arithmetic circuit for complement function (special case for Riemann Zeta Function for prime distribution) using Fourier Analysis and Euler product and some conjectures based on the property of this non-uniform circuit graph family. The draft writeup is at point 24 of <http://sourceforge.net/p/asfer/code/HEAD/tree/AstroInferDesign.txt>.

This has some importance due to the connection that can be conjectured between the zeros of Riemann Zeta Function (if Riemann Hypothesis is true) and the patterns in the circuit graphs for this non-uniform family. Also it has connections to Ihara Zeta Function and Ramanujan Graphs. Riemann Zeta Function can be written in terms of Ihara identity of Ihara Zeta Function by assuming infinite set of prime-regular graphs. Solving the product of Characteristic polynomials of the prime regular graphs for the zeroes by equating them to eigenvalue (\leq degree) gives some information about the Riemann Zeta Function zeros. Illustrations for this are at :

<https://sites.google.com/site/kuja27/RamanujanGraphsRiemannZetaFunctionAndIharaZetaFunction.pdf?attredirects=0&d=1>

This is an extension of a miniproject course I was doing with Meena Mahajan (IMSc) in 2011 (<http://arxiv.org/abs/1106.4102>)

My detailed CV is at: https://sites.google.com/site/kuja27/CV_of_SrinivasanKannan_alias_KaShrinivaasan_alias_ShrinivasKannan.pdf?attredirects=0&d=1. Open Source Product codebases above are my own.

<p>① <u>RZF & IZF</u> - Rederived 29 March Saturday 29-03-2014 24/10/2014</p>	<p>$s = a + ib$ is a zero of RZF in RHS. (2) 31-03-2014 Telugu New Year's Day March Monday 31</p>
$\begin{aligned} & \bar{Z}_1(s) \det \left(I - A q_1^{-s} + q_1^{1-2s} I \right) = (1 - q_1^{-s})^M \\ \Rightarrow & \frac{(1 + q_1^{-s})^{M-1}}{(1 + q_1^{-s})^{M-1}} = (1 - q_1^{-s}) \end{aligned}$	$\begin{aligned} & \text{for some } s \text{ and some prime } q_1 \\ & (1 + q_1^{-s}) = 0 \\ & -1 = q_1^{-s} \\ & \pi = -\log q_1 \\ & n = \frac{-\log q_1}{\pi} \end{aligned}$
$\begin{aligned} & \bar{Z}_1(s) \det \left(I - A q_1^{-s} + q_1^{1-2s} I \right) = (1 - q_1^{-s}) \\ & \frac{(1 + q_1^{-s})^{M-1}}{(1 + q_1^{-s})^{M-1}} = (1 - q_1^{-s}) \\ & (1 + q_1^{-s}) = \frac{1}{(1 - q_1^{-s})} \end{aligned}$	$\begin{aligned} & q_1^{a+ib} = -1 \\ & q_1^a q_1^{ib} = -1 \\ & q_1^{ib} = \frac{-1}{q_1^a} \\ & e^{ib \log q_1} = \frac{-1}{q_1^a} \end{aligned}$
$\begin{aligned} & \text{For all prime regular products in LHS} \quad \text{Amavasya, 30-03-2014} \\ & \text{Graph} \\ & \text{30 Sunday} \end{aligned}$	$\begin{aligned} & \cos(b \log q_1) = \frac{-1}{q_1^a} \\ & \cos(b \log q_1) + i \sin(b \log q_1) = \frac{-1}{q_1^a} \\ & \text{Im}(\frac{-1}{q_1^a}) = \frac{\pi}{2} \\ & b \log q_1 + \frac{\pi}{2} = 0 \\ & \cos(b \log q_1) = \frac{-1}{q_1^a} \\ & \frac{b \log q_1}{\pi/2} = \infty \end{aligned}$
$\begin{aligned} & \text{If RHS} = 0 \\ & \text{February 2014} \\ & \begin{array}{ccccccc} \text{S} & \text{M} & \text{T} & \text{W} & \text{T} & \text{F} & \text>S \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 8 & 9 & 10 & 11 & 12 & 13 & 14 \\ 15 & 16 & 17 & 18 & 19 & 20 & 21 \\ 22 & 23 & 24 & 25 & 26 & 27 & 28 \end{array} \end{aligned}$	$\begin{aligned} & \text{April 2014} \\ & \begin{array}{ccccccc} \text{S} & \text{M} & \text{T} & \text{W} & \text{T} & \text{F} & \text>S \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 8 & 9 & 10 & 11 & 12 & 13 & 14 \\ 15 & 16 & 17 & 18 & 19 & 20 & 21 \\ 22 & 23 & 24 & 25 & 26 & 27 & 28 \end{array} \end{aligned}$

<p>Continued from 31/3/2014</p> <p>3 April Thursday</p> $\frac{b \log q}{\pi} = -n$ $\cos(b \log q + (2n+1)\frac{\pi}{2}) = \frac{-1}{q^a}$ $\cos(b \log q + \left(\frac{2b \log q + 1}{\pi}\right)\frac{\pi}{2}) = \frac{-1}{q^a}$ $\cos(b \log q + b \log q + \frac{\pi}{2}) = \frac{-1}{q^a}$ $\cos(\frac{\pi}{2}) = \frac{-1}{q^a}$ $0 = \frac{1}{q^a} \quad \boxed{q^a = \infty}$ $\therefore a = \infty$ <p>If denominator is infinity</p> $\left[z_1 z_2 \dots \det() \det() \dots \right] = \infty$ $\text{or } \left[z_1 z_2 \dots \det() \det() \dots \right]^{1/(q-1)} = \infty$	<p>3 03-04-2014 ★ Kritikal</p> <p>4 April Friday</p> $z_1 z_2 \dots \det() \det() \dots$ <p><u>25/4/2014</u> If R.H is true:</p> $q^{16} \leq \frac{\sqrt{q} \pm \sqrt{q-4}}{2} \quad \text{for some } q\text{-regular graph (q prime)}$ $q^{16} \leq \frac{\sqrt{q} \pm \sqrt{q-4}}{2}$ $\cos(b \log q + i \sin b \log q) \leq \frac{\sqrt{q} \pm \sqrt{q-4}}{2}$ $\sin(b \log q + n\pi) = 0$ $b \log q = -n\pi$ $n = -\frac{b \log q}{\pi}$ $\cos(b \log q + (2n+1)\frac{\pi}{2}) \leq \frac{\sqrt{q} \pm \sqrt{q-4}}{2}$ $\cos(b \log q + (-b \log q + n\pi) + \frac{\pi}{2}) \leq \frac{\sqrt{q} \pm \sqrt{q-4}}{2}$ $\cos(\frac{\pi}{2}) \leq \frac{\sqrt{q} \pm \sqrt{q-4}}{2}$ $0 \leq \frac{\sqrt{q} \pm \sqrt{q-4}}{2}$
---	--

S	M	T	W	T	F	S
10	11					1
2	3	4	5	6	7	8
8	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29

For $q < 4$ (2,3)
 above has imaginary part

For $g \geq 5$ 7 08-04-2014

8 April Tuesday

$$a - \frac{1}{2} + ib \leq \log \left(\frac{\sqrt{g} \pm \sqrt{g+4}}{2} \right)$$

$$a - \frac{1}{2} \leq \log \left[\frac{\sqrt{2} \pm \sqrt{q-4}}{2} \right] \log q$$

$$a \leq \frac{\log \left[\frac{\sqrt{2} \pm \sqrt{2-4}}{2} \right]}{\log 2} + \frac{1}{2}$$

$$q^s + q^{1-s} \leq q$$

$$q^s + \frac{q}{q^s} \leq q$$

$$q^{25} + q \leq q^{5+1}$$

$$\begin{array}{cccccc} \text{March 2014} & & & & & \\ \text{M} & \text{T} & \text{W} & \text{T} & \text{F} & \text{S} \\ \hline n & & & & & \end{array} \quad q^{2s} + q - q^{s+1} \leq 0$$

March 2014

Q8 Re-Re derived differently $a+ib$ 25/10/2014 12:45pm
09-04-2014 April 9

$$e^{(2a+2ib)\log z} + z - e^{(a+1+ib)\log z} \leq 0$$

$$e^{2a\log g} \times e^{2b\log g} + g - e^{(a+b)\log g} \times e^{b\log g} \leq 0$$

$$e^{2a \log_2 q} \left(\cos(2b \log_2 q) + i \sin(2b \log_2 q) \right) + e^{(a+r) \log_2 q} \left(\cos b \log_2 q + i \sin b \log_2 q \right) \leq 0$$

$$\text{Re}(z) \geq \log q, \quad e^{2\pi i \log q} \cos(2\pi \log q) + q + e^{(\alpha+1)\log q} \cos \pi \log q \leq 0$$

$$e^{2a\log^q g} \sin(2b\log g) + e^{(a+1)\log^q g} \sin b\log g \leq 0$$

Reminiscences

$$R(x) = 9 \cos(2.6 \log x) + 9 + 9 \cos(b \log x)$$

$$q^{2a} \sin(2b \log q) + q^{(au)} \sin b \log q \leq 0$$

May 2014

Dividing Real and Imaginary: ⑨ 10-04-2014

10 April Thursday

$$q^{2a} \sin(2b\log q) = -q^{(a+1)} \sin b\log q$$

$$q^{2a} \cos(2b\log q) + q = -q^{(a+1)} \cos b\log q$$

$$\frac{q^{2a} \sin(2b\log q)}{q^{2a} \cos(2b\log q) + q} = \tan b\log q \quad ③$$

If RH is true $a = \frac{1}{2}$ for all $s = a+ib$

From ③ $q^{2a} = q$

$$\frac{q \sin(2b\log q)}{q \cos(2b\log q) + 1} = \tan b\log q \quad -④$$

$$\frac{\sin(2b\log q)}{\cos(2b\log q) + 1} = \tan b\log q \quad -⑤$$

March 2014

S	M	T	W	T	F	S
30	31	1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31	1	2

11-04-2014 April Friday 11

$$e^{ix} = \cos x + i \sin x \quad \text{⑩}$$

$$e^{ix} \cdot e^{ix} = (\cos x + i \sin x)^2$$

$$e^{2ix} = \cos^2 x + 2i \sin x \cos x + \sin^2 x$$

$$\cos 2x + i \sin 2x = \cos^2 x - \sin^2 x + i(2 \sin x \cos x)$$

$$\sin 2x = 2 \sin x \cos x$$

$$\sin(2b\log q) = 2 \sin(b\log q) \cos(b\log q)$$

$$\cos(2b\log q) = \cos^2 b\log q - \sin^2 b\log q$$

From ⑤

$$\frac{2 \sin(b\log q) \cos(b\log q)}{\cos^2 b\log q - \sin^2 b\log q + 1} = \tan b\log q$$

$$\frac{2 \cos(b\log q)}{\cos^2 b\log q - \sin^2 b\log q + 1} = \frac{1}{\cos b\log q} \quad -⑥$$

May 2014

S	M	T	W	T	F	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31	1	2	3	4

12 April Saturday

$$\frac{2 \cos^2(b\log q)}{2 \cos^2(b\log q)} = 1 \quad 12-04-2014$$

$\Rightarrow a = \frac{1}{2}$ for all $s = a+ib$

from $\det(I - A q^{-s} + q^{1-s} I) = 0$

and using $q^s + q^{1-s} \leq q$

$$(or) q^{25} + q^{-25} - q^{50} \leq 0$$

Page 8, 9, 10, 11 9/4/2014 to 12/4/2014

Assumption of $a = \frac{1}{2}$ for all $s = a+ib$ (non-Ramanujan) doesn't give any contradiction (graphs)

Ramanujan graphs have RZF as zeros

for $a \neq \frac{1}{2} \cdot \frac{q^{2a} \sin(2b\log q)}{q^{2a} \cos(2b\log q) + q}$

would not be equal to RHS as q would not be common in num & denom

March 2014

S	M	T	W	T	F	S
30	31	1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31	1	2

14-04-2014 April Monday 14

For numerator product = 0. ⑫ (page 29/3/2014)

Tamil New Year's Day, Dr. Ambedkar Jayanti, Pournami

$1 + q^{-s} = 0$ for some $s = a+ib$
when RZF in RHS is zero.

$$q^{-s} = -1$$

$$q^s = -1$$

$$q^{a+ib} = -1$$

$$(a+ib)\log q = -1$$

$$a\log q + ib\log q = -1$$

$$e^{a\log q} \cdot e^{ib\log q} = -1$$

$$e^{a\log q} (\cos b\log q + i \sin b\log q) = -1$$

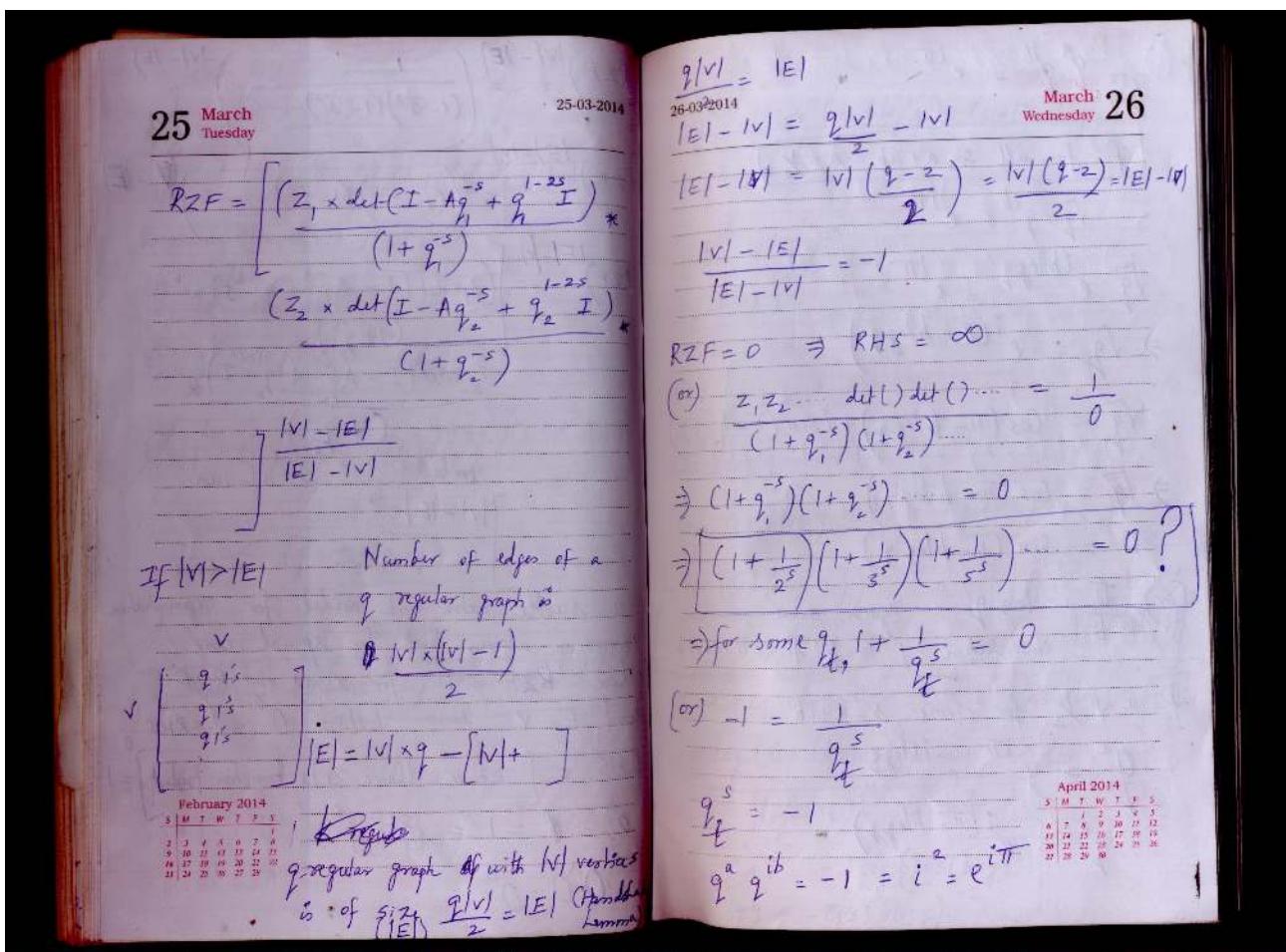
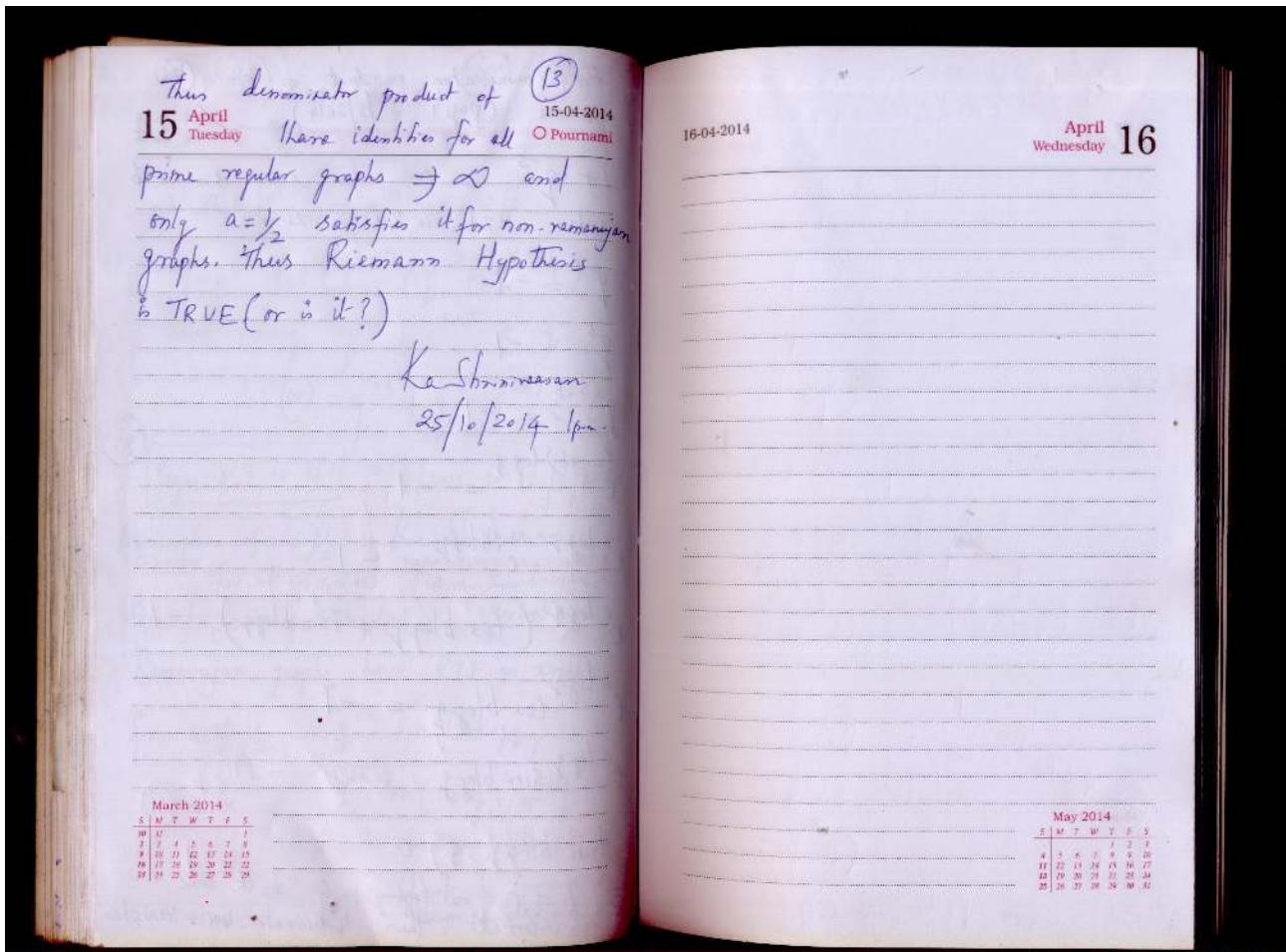
$$e^{a\log q} \cos b\log q = -1$$

$$e^{a\log q} \sin b\log q = 0$$

May 2014

S	M	T	W	T	F	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31	1	2	3	4

$\tan b\log q = 0$
 $\Rightarrow b = 0$ or $\log q = 0$ be the case
incorrect. Thus numerator never vanishes.

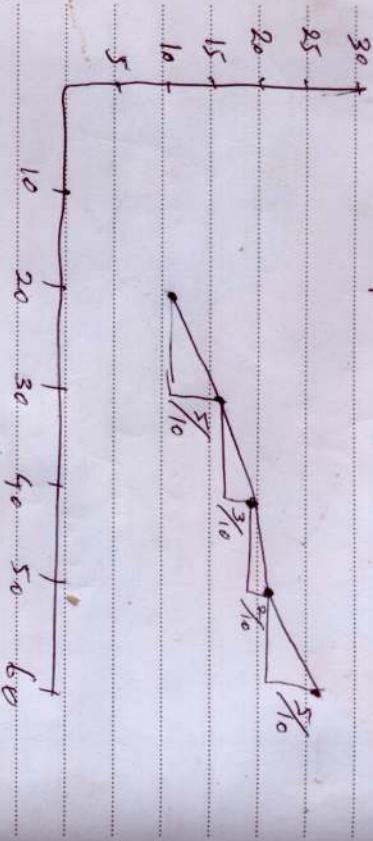


21

January 21-01-2014

blue total white
blue 16/30 after 30 coin tosses
blue 19/40 after 40 coin tosses
blue 21/50 after 50 coin tosses

26/60 after 60 coin tosses



16/30
19/40
21/50

Distinct partitions (restricted partitions with distinct parts):
Grainger - Euler's sum - no of partitions with distinct parts = no of partitions with odd parts.

December 2013

10
 $\prod (1+x^i)$ is generating function

$= \sum q(n) x^n$ number of distinct partitions

January 22

Wednesday

22

Number of valid multiparty voting patterns = number of distinct partitions of n ($q(n)$) \equiv number of k -sets with distinct collision chain size (without permutation) partition is NP-hard finding one such

$$a_1 + a_2 2 + a_3 3 + a_4 4 + \dots + a_n n = n$$

$$(a_i = 0 \text{ or } a_i = 1 \text{ for some } a_i)$$

Out of n possible bit patterns only $q(n)$ patterns are valid (distinct partitions).
Above is a variant of money changing problem with denominations 1, 2, 3, 4, ..., n and finding the numerations (16 or 1).
Thus it is a direct reduction from

NP - hard money changing problem thus finding one valid multiparty voting pattern (distinct parts) is NP-complete. Above always has a biggest part (majority)

(subset sum problem is also similar to the above)

\Rightarrow Finding such a hash function in NP-complete

Demonstrating with Heuristic in NP-complete

Affirmative proof in that $\sum_{i=1}^n$

Circuits for voters

	S	M	T	W	T	F	S
1	2	3	4	5	6	7	1
2	9	10	11	12	13	14	8
3	15	16	17	18	19	20	13
4	22	23	24	25	26	27	22
5	29	30	31				23

February 2014

	S	M	T	W	T	F	S
1	2	3	4	5	6	7	8
2	9	10	11	12	13	14	13
3	16	17	18	19	20	21	22
4	24	25	26	27	28		23

$$6 \times 5 = 30$$

23 17/4/2014 Schur's Theorem

Thursday

23-01-2014

finding a multi-partitioning with no two 24-01-2014 candidates getting equal votes. Friday

Number of denumerants for diophantine equation $a_1 \cdot 1 + a_2 \cdot 2 + a_3 \cdot 3 + \dots + a_n \cdot n$ is given by Schur's theorem and is $= n$ thus a lower bound for number of hash functions (since order is ignored).

$$x = c_1 a_1 + \dots + c_n a_n \text{ if } x = n$$

for $n = 1 \cdot a_1 + 2 \cdot a_2 + 3 \cdot a_3 + \dots + n \cdot a_n$ number of denumerant tuples is $\binom{n}{a_1, a_2, \dots, a_n}$

$$\binom{n}{a_1, a_2, \dots, a_n} \approx \frac{n^{(n-1)}}{a_1 a_2 \dots a_n}$$

$$\approx \frac{1}{\sqrt{2\pi n}} \frac{n^n}{e^n} a_1 a_2 \dots a_n$$

$$\approx \frac{1}{\sqrt{2\pi n}} \frac{n^n}{e^n} \approx \frac{1}{\sqrt{2\pi (a_1 a_2 \dots a_n) x^n}}$$

- 1) find one such denumerant (NP-hard due to induction from MCP)
- 2) $\text{permute } \leq (n!)$

Ka-Shrivastava

December 2013						
S	M	T	W	T	F	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

17/4/2014

Recurrence relation for ①

$$P(N, M, n) = P(N, M-1, n) +$$

$$n \text{ is partitioned into } P(N-1, M, n-M)$$

1) parts and each part of size at most N

COLLISION PRONE

Maximizing hash collisions is necessary for clustering similar data into a single collision chain (opposite of collision resistance (DNA Similarities and Voicemails Similarities need to minimize collisions))

COLLISION FREE

Collision Resistant Hash Function is a one-way function such that any randomized polynomial time algorithm finds Collision with negl probability ($f(x) = f(y)$).

Restricted partitions with can be classified as following.

- 1) Set of partitions in which no part exceeds k . (more COLLISION FREE)
- 2) Set of partitions in which no part is less than k . (more COLLISION PRONE)

- 1) is more Collision - Free (when mapped to hash table)
- 2) is more Collision - Maximizing (when mapped to a hash table).

February 2014						
S	M	T	W	T	F	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

25

January
Saturday

27-01-2014

January
Monday 27

$$\begin{array}{c}
 \text{Expt No} \quad N \quad \text{atmos} \quad N \\
 \hline
 M & \boxed{1} & \boxed{2} & \boxed{3} & \boxed{4} & \boxed{5} & \boxed{6} \\
 \vdots & \boxed{1} & \boxed{2} & \boxed{3} & \boxed{4} & \boxed{5} & \boxed{6} \\
 M & \boxed{1} & \boxed{2} & \boxed{3} & \boxed{4} & \boxed{5} & \boxed{6} \\
 \hline
 & & & & & & \boxed{25-01-2014}
 \end{array}$$

Subtracting the above
 we get
 recurrence for $\textcircled{1}$
 from $P(n)$ gives $\textcircled{2}$ (set of partitions
 in which no part is less than a constant)

for Spring season
17/01/2014

26 Sunday

Republic Day, 26-01-2014

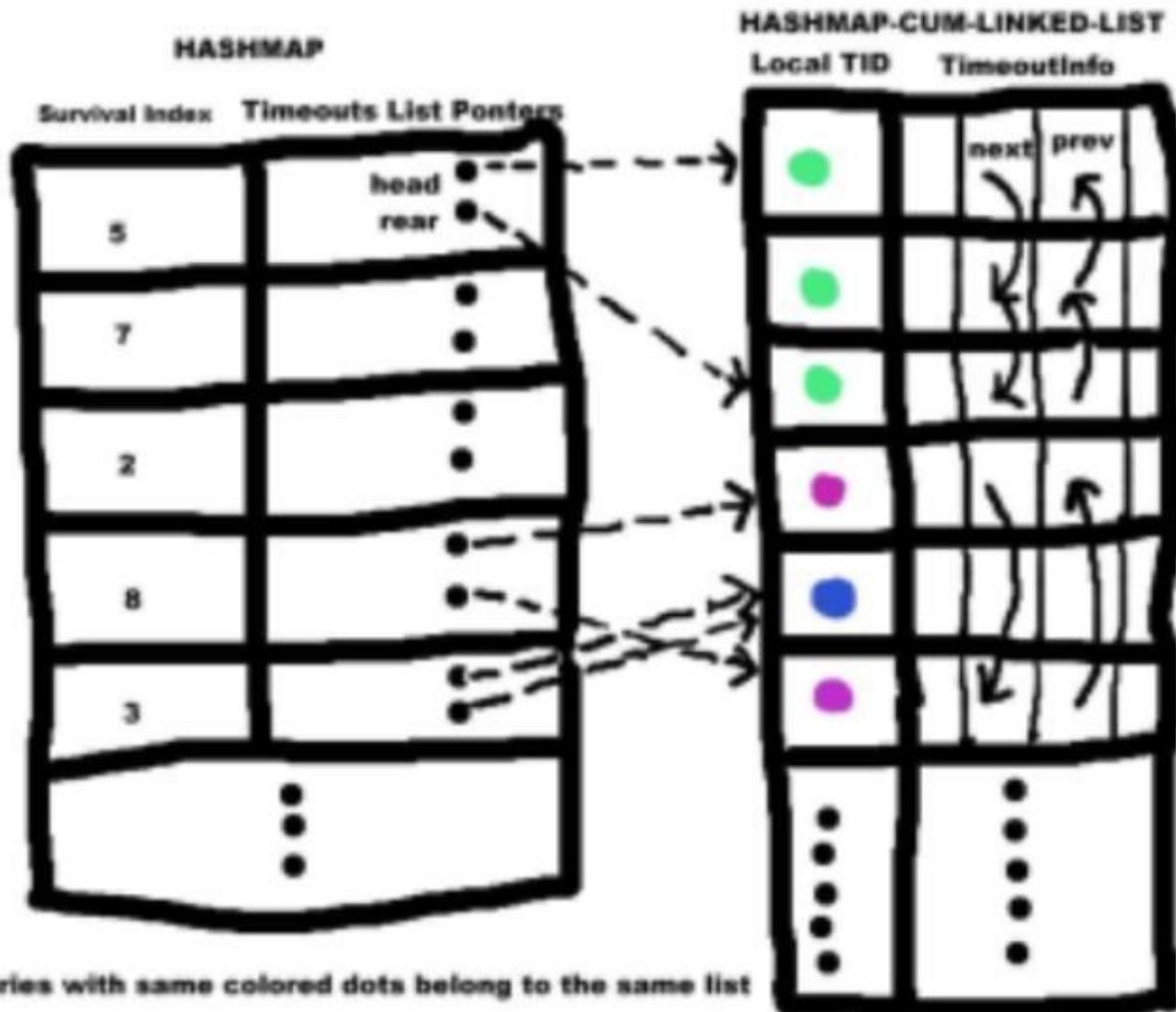
December 2013

S	M	T	W	T	F	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

February 2014

S	M	T	W	T	F	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

TIMEOUT MANAGER REDESIGN PROPOSAL



setTimeout()

2a. Maximum number of invocations of timeout thread that a transaction will survive is calculated as below:

Survival Index

(or)

maximum number of invocations of TimeoutThread this txn may survive =

CEIL{(timeout_for_this_txn - time_remaining_for_the_next_invocation_of_timeout_thread) / TIMEOUT_INTERVAL} + 1

An illustration:

TO interval

|<----->|<----->|<----->|
---><---|

left right

remainder remainder

|*****|

max transaction duration(or) timeout

|-----|

(time remaining for the next invocation of TO thread)

|-----|

(timeout for this txn - time remaining for the next invocation of timeout thread)

|| (i.e only 2 integers are in the range)

CEIL{(timeout for this txn - time remaining for the next invocation of timeout thread)/TIMEOUT_INTERVAL}

||| (i.e add one more to compensate for left remainder)

CEIL{(timeout for this txn - time remaining for the next invocation of timeout thread)/TIMEOUT_INTERVAL} + 1

-----addTimeoutInfo()-----

2b.synchronized(pendingTimeouts)

{

If value obtained in 2(a) exists within HashMap of SurvivalIndices

{

the transaction info will be put into localTID->TimeoutInfo Map and appended to already existing linked list for that value

update head and rear pointers for this list

}

else

{

put a new survival index entry in SurvivalIndex HashMap; put timeout info into localTID->TimeoutInfo Map and add the same as the first element of the timeout info list

initialise head and rear pointers to this list

}

}

-----addTimeoutInfo()-----

TimeoutManager.run()

while(true)

{

Sleep for TIMEOUT_INTERVAL

For all survival Indices in pendingTimeouts

{

increment timeoutThreadInvocationCount for this entry by 1

if(survivalIndex == timeoutThreadInvocationCount) /* there is a TIMEOUT */

{

```
synchronized(TimeoutListPointer for this list) // as it is the wrapping object of this list
```

```
{
```

All the transactions in the TimeoutInfo linked list corresponding to this survival index have to be rolled back/recovered accordingly, if still alive. The head and rear pointers to the list are maintained by TimeoutsListPtr object which is part of the survival index entry.

Remove the survival index entry and accompanying timeout info list

```
}
```

```
}
```

```
}
```

```
}
```

SURVIVAL INDEX equals() method

a. Consider the following scenario of SurvivalIndex Map contents. Figure in parentheses is timeout thread invocation count. Letters in RHS of arrow signify transactions corresponding to this survival index:

1) Before invocation of Timeout thread

5(0) => a b c

4(0) => d e

6(0) => f g h i

2) After invocation of timeout thread

5(1) => a b c

4(1) => d e

6(1) => f g h i

Now say a transaction j with SurvivalIndex 4 has to be added to this map. But to which list? It is decided by the equals() method of SurvivalIndex object. Two survival indices A and B are equal iff,

A.survivalIndex - A.timeoutThreadInvocationCount == B.survivalIndex -

B.timeoutThreadInvocationCount

Using above criterion, transaction j with SurvivalIndex 4 (and timeoutThreadInvocationCount 0, as it is nascent) is equal to SurvivalIndex 5 (since $5 - 1 = 4$ (survivalIndex - timeoutThreadInvocationCount))
So our map becomes,

5(1) => a b c j

4(1) => d e

6(1) => f g h i

SURVIVAL INDEX hashCode() method

A new static variable globalTimeoutThreadInvocationCount is introduced in Timeout Manager for the purpose described below. It will be incremented on each invocation of Timeout Thread. It is publicly accessible.

Survival Index should also implement hashCode() method since survivalIndex varies after each Timeout Thread invocation. As a result, hashCode() must be returned in such a way as to get the correct entry. Consider the following :

Logical view of survivalIndex -> TimeoutsListPtr map

SurvivalIndex ----- TimeoutsListPtr

1

11

21

4

14
24
7
17
27
37

Internal representation of the above logical view (Before Timeout Thread Invocation)

Let array_length be 10.

globalTimeoutThreadInvocationCount == 0

array_index(hashCode % array_length) all survival indices with same value of (hashCode % array_length)
1 ----- 1 -> 11 -> 21
4 ----- 4 -> 14 -> 24
7 ----- 7 -> 17 -> 27 -> 37

Internal representation of the above logical view (After Timeout Thread Invocation)

globalTimeoutThreadInvocationCount == 1

array_index(hashCode % array_length) all survival indices with same value of (hashCode % array_length)
1 ----- 0 -> 10 -> 20
4 ----- 3 -> 13 -> 23
7 ----- 6 -> 16 -> 26 -> 36

Now, suppose a transaction with survival index 13 has to be added to localTID->TimeoutInfo Map, we have to get a hashCode in such a way that we end up in index 4 (instead of 3). So hashCode must return the value of the expression:

**this.survivalIndex - this.timeoutThreadInvocationCount +
TimeoutManager.globalTimeoutThreadInvocationCount**

For our example, above expression becomes $13 - 0$ (because it has just been created) $+ 1 = 14$

So, $(hashCode() \% array_length)$ becomes, $14 \% 10 = 4$ which is the required index.

Then the equals method starts a search down the list for this index, and decides that there is an entry 13 already using logic described in previous section on equals() method.

BENEFITS

- a. timeout is calculated **a priori** instead of checking at every timeout thread invocation.
- b. overhead to check each and every element of the pendingTimeouts for timeout is minimized to large extent by iterating only through survival indices on each timeout thread invocation. Since in a typical stress scenario, large number of transactions will be created in each timeout interval, they all would have same survival index (provided same timeout is set for all txns which will mostly be the case) and thus would be part of a single timeout info list. Thus **length of survival index list is directly proportional to survival index instead of total number of transactions.**

EXPERIMENT

A performance comparison of existing timeout manager and the implementation of above design :

Number of requests = 30 users * 30 iterations

Average Time for Existing Timeout manager - 27 seconds

Average Time for new implementation - 24 seconds

NOTE: The above values are the best among many runs for the corresponding implementation

TAC 2010 Summarization Track - Update Summarization with Interview Algorithm

Ka.Shrinivaasan, Chennai Mathematical Institute (CMI) (shrinivas@cmi.ac.in)

Abstract

Existing models for ranking documents(mostly in world wide web) are prestige based. In this article, alternative graph-theoretic schemes to objectively judge the merit of a document independent of any external factors (like link graph) and without probabilistic inference are proposed and application of these to TAC 2010 Update summary component is presented.

1 TAC 2010 dataset preprocessing and algorithms used

TAC 2010 dataset was split into candidate and reference sets. 25 out of 92 folders in the datasets were evaluated. In each folder, the datasets were split arbitrarily into reference and candidate texts. Both reference and candidate texts were concatenated to get two big documents - reference and candidate. These preprocessed texts were then applied to Interview algorithm described in detail below. Description of the algorithm is essential to understand how the dataset was evaluated to get intrinsic merit score and application of a threshold to this score to create summary. No guided summarization aspects were used in the TAC 2010 runs and focus was on update summarization component alone.

2 Motivation

Motivation for objective, independent judgement of a document is founded on the following example:

Judge X decides about the merit of an entity Z purely by what other entities opine about Z without interacting with Z; Judge Y decides about the merit of Z by interacting only with Z. Question now is who is better judge - X or Y.

Probability of judgmental error of judge X is equal to probability of collective error of entities opining about Z while probability of judgemental error of judge Y is 0.5 as the following elementary arithmetic shows. Let us assume there are $2n$ voters and they need to decide/vote on whether a candidate is good or bad.

A candidate getting majority ($n + 1$ good votes) will be winner.

Question: What is the probability that people have made a good decision?

Answer: Probability of each voter making a good decision is p and bad decision is $1 - p$ ($0 <= p <= 1$). Let $p = 0.5$ for an unbiased voter.

So for a candidate to be judged 'good', atleast $n + 1$ people should have made a good decision. Probability of a good choice for these $2n$ voters, skipping the calculations, is :

$$\begin{aligned} P(\text{good}) = & ((2n)!/4^n) * \\ & ((1/((n+1)!(n-1)!)) + \\ & 1/((n+2)!(n-2)!)) \\ & + \\ & + 1/((n+n)!(n-n!))) \quad (1) \end{aligned}$$

If there is an objective judgement without voting, probability of good decision is 0.5. It is interesting to see that above series tends to 0.5 as n grows infinitely. Thus, the judgement-through-majority-vote error probability is equal to the error probability of judge X who uses only the inputs from witnesses to judge Z while judgement-through-interaction(without election) error probability is equal to the error probability of judge Y (i.e. 0.5) who does not use witnesses. Thus, both judges X and Y are equally fallible but the cost incurred in a real world scenario for simulating X far outweighs that of Y. Thus it is worth delving into schemes for objective judgement like Y.

3 Three algorithms presented hereunder

1. Maxflow and Path lengths of Citation graphs - objective judgement (differs from Pagerank since it is Maxflow based and not prestige based)
2. Generalized Recursive Gloss Overlap - objective judgement (simulates judge Y with a 'white-box', invasive, intrinsic merit scoring) - covers majority of this report
3. Interview algorithm - objective judgement (simulates judge Y; Uses questions and answers to judge a candidate - 'black-box' and less-invasive - and also incorporates intrinsic merit score obtained from either MaxFlow of Citation graph or Generalized Recursive Gloss Overlap)

4 Directed Graph of Citations

4.1 Average Maxflow and Path lengths of Directed Graph of Citations

Given a corpus, algorithm constructs directed graph of incoming links to a document x from those documents chronologically later than x . Thus corpus is partitioned into set of digraphs. Indegree of a vertex in this digraph reflects the importance of a document represented by a vertex. This digraph can be thought of as a flow network where concept flows from a document to others which cite. Each edge has a weight. Capacity/weight for an (u,v) edge is defined as number of references v makes while citing u though there could be other ways to weight an edge. Assigning polarity to this capacity/weight is discussed in 4.2. Mincut of the digraph is the set of documents which are "potentially most influenced by the source document" (because maximum flow of concept from source occurs through this set to outside world/sink). Thus size of maxflow/mincut, averaged over all vertex-pairwise maxflow values, is a measure of influence of a source document in a community and thus points to its merit. (E.g., Chronology for web documents can be found by 'Last-modified' HTTP header which every dynamic document server is mandated to send to client). Alternative way to get the merit is to count the number of vertices in a predefined radius from source (i.e set of paths of some fixed length from source) which can be less accurate and sometimes misleading. Thus documents can be ranked using average Maxflow values. Advantage of this scheme is that it quantifies the ex-

tent of percolation of a concept within a community through Maxflow, without giving importance to the prestige measure of the vertices(documents) involved. So, this is one way of objectively assessing the merit of a vertex(document). Implementation applies Ford-Fulkerson algorithm to each s, t distinct pair and finds the average maxflow out of each vertex.

4.2 Polarity of citation edge

Parse the document/sentence containing the citation/link into tokens and find polarity. Whether a word is positive or negative can be decided by:

1. looking up a sentiment annotated ontology (e.g positivity/negativity of a lemma in SentiWordNet) or
2. entropy analysis - using $\sum_{i=0}^1 (-P(i)\log P(i))$ where $P(0)$ = percentage of positive words and $P(1)$ = percentage of negative words. Closer the entropy to zero, clearer the sentence/document on its viewpoint (very good or very bad) or
3. recursive gloss overlap algorithm to the citing document to get the polarity/sentiment of context citing the document.

Implementation tries all the three above. If the polarity/sentiment is negative, the weight for edge (u,v) is made negative in citation digraph, indicating a negative flow of concept to vertex v from the cited vertex u .

5 Definition Graph Convergence(or)Generalized recursive gloss overlap

5.1 Motivation for computing Intrinsic Merit of a document

Intrinsic merit is defined as the amount of intellectual effort put forth by the reader of a document and we try to quantize this effort. It is important to note that this quantized effort is independent of any observer/link-graph. Any document goes through some human understanding and we try to model it through what can be called Iceberg/Convergence/Generalized recursive gloss overlap algorithm (named so because a web document contains only a tip of the knowledge a document represents and understanding the document requires deeper recursive understanding of the facts or definitions the document is home to.). For example, going

through a research paper requires the understanding of the concepts which draw a logical graph in our mind. Thus time spent on grasping the concepts and hence the intrinsic merit is proportional to the size and complexity of this graph and points to its merit (which is equal to the intellectual effort of the human reader). Since WordNet is the existing model for semantic relationship, we will try to establish that a text document can be mapped to a graph which is a subgraph of WordNet and merit can be derived applying some metrics on this graph. This is the intuition behind the algorithms that follow.

5.2 Definition tree of a document

Given a document its definition tree is recursively defined as

Definition 1. *definitiontree(all keywords of document) = definitiontree(term1) definitiontree(term2) ... definitiontree(termn) where term1, term2,...termn occur in the definition of keywords of a document.*

For example, let us consider the following document which talks about Kuratowski theorem

Document1 = Every K5,K3,3-free graph is planar

This document contains key terms like "K5,K3,3-free", "graph" and "planar". Now we recursively construct the definition tree for these terms. Key terms are decided after filtering out stopwords and by computing TF-IDF and only terms above a threshold tfidf are chosen for constructing the definition graph.

definitions at level 1:

1. K5 = Complete graph of 5 vertices (key terms: graph, vertices)
2. K3,3 = graph of two sets of 3 vertices each interconnected (key terms: graph, two sets, vertices, interconnected)
3. graph = set of vertices and edges among them (key words: vertices, edges, set)
4. planar = graph embedded on a plane (key words: graph, embedded, plane)

Thus the definition tree goes deeper as each keyword/concept is dissected and understood. Given above is level-1 grasping of the document. Important thing to note is that intersection of the sets of keywords in the definition of K5, K3,3, graph and planar is not an empty set (glosses for two or more keywords overlap). For example, intersection of definitions of

K5 and K3,3 is the set {graph, vertices}. Thus the overlap of the terms "graph" and "vertices" in two definitions of K5 and K3,3 is an indication of deeper cohesion/interrelatedness of the terms in the document. Thus the replicated terms (represented by vertices) in the definition tree can be merged to get convergence (gloss overlap generalized to more than two glosses). Thus the definition tree is transformed into definition graph (since a vertex can have more than one parent) by merging replicated keyword vertices into 1 vertex. Synset definitions in WordNet gloss are used for getting keyword definitions in the implementation. But WordNet Gloss does not work for terms specialized for a domain (e.g. gloss for "graph" does not have a synset for graph theory as part of its senses set). This requires ontologies for the class the document belongs to. Thus recursive gloss overlap algorithm is limited by WordNet in present implementation. At each level, word sense disambiguation is done by following Lesk's algorithm adapted to Generalized Recursive Gloss overlap to choose the synset definition fitting the context. It is important to note that 1) only one relation ("is in definition of") is used and 2) only keywords within the document are considered 3) gloss overlap is computed recursively at each level of understanding till required depth is reached.

5.3 Definition graph convergence and steps of Recursive Gloss Overlap algorithm

Convergence of a document is defined as the decrease in the number of unique vertices of the set of definition trees of its keywords from level k to level k+1. For example definition tree of the above document converges to {edges, vertices} after expanding the definition tree further down. Thus the above document has "edges" and "vertices" as its undercurrent. Thus the Convergence algorithm takes no labelled examples for inference. Only requirement is to have a dictionary/gloss/ontology of terms and their corresponding definitions. If a document's definition tree does not converge within a threshold called "depth" number of levels then the document is most likely less meaningful or of low merit. Thus the Convergence algorithms strikingly adapts an iceberg which has seemingly unconnected set of "tips" at the top but as we go deeper get unified. Level where this unification happens is a differentiator of merit. If while recursively expanding the definition tree, a vertex results in a child vertex which is same as some sibling of the parent then we compute and remove the intersection of keywords at present and previous level - since these common vertices have already been grasped. Accord-

ingly, number of edges, vertices and relatedness are updated for each level. Number of vertices are adjusted for removal of common tokens, but number of edges remain same since they just point to a different vertex at that level. This process continues top-down till required depth is reached.

Steps:

1. Get the document as input
2. currentlevel = 1
3. keywordsatthislevel = {keywords from the document through tfidf filter (e.g > 0.02)}
4. While (currentlevel < depthrequired) {
 - For each keyword from keywordsatthislevel lookup the best matching definition for the keyword and add to a set of tokens in next level - requires WordSenseDisambiguation - implementation uses Lesk's algorithm
 - Remove common tokens with previous levels since they have been grasped in previous level (this is an optimization)
 - Update the number of vertices, edges and relatedness (vertices correspond to unique tokens, edges correspond to the single relation 'y is in definition of x' and relatedness is linear overlap or quadratic overlap) and Update tokensofthislevel
 - currentlevel = currentlevel + 1
}
5. Output the Intrinsic merit score =

$$|vertices| * |edges| * |relatedness| / firstconvergencelevel \quad (2)$$

Where

- Relatedness = $Number\ of\ Overlaps$ (linear, also called as convergence factor) (or)
- Relatedness = $Number\ of\ Overlapping\ Parents * Number\ of\ Overlaps^2$ (quadratic) (3)

- firstconvergencelevel = level of first gloss overlap

At the end of recursive gloss overlap, nodes with high number of indegrees(parents) are indicators of the class of the document since greater the indegree, greater is the number of keywords overlapping (voting for an underlying theme). From graph theoretic view, Definition Graph constructed above is a multipartite graph since vertices can be partitioned into sets with no edges within a set and edges only across sets (without removal of common tokens between levels - which is only an optimization since by removing common tokens we redirect edges to vertices within the same set and multipartiteness is lost). Preserving multipartiteness is useful since it groups the tokens at each level of recursion into single set with edges across these sets - multipartite cliques of this multipartite graph can be analyzed to get the robustness. Moreover, this algorithm ignores grammatical structure. Reason is that principal differentiator in analyzing relative merit of two documents is the quality of content and complexity of content and both documents are equally grammatical. Quality of content is proportional to the vertices of the definition graph and complexity of the content is proportional to the relatedness and edges of definition graph. In spite of ignoring grammatical structure, the graph constructed above is context-sensitive since word sense disambiguation is done while choosing the synset matching a keyword. This way, the definition graph is a graph representation of the knowledge in the document sans the grammatical connectives.

5.4 Definition of shrink

Definition 2. Let us define "shrink" to be the amount of decrease in the number of unique vertices between levels k and $k + 1$ during convergence (gloss overlap)

5.5 Comparison of two documents for relative merit - two examples

Document1 : Car plies on sky

Constructing definition graph for level-1 we get,

1. Car - automobile used for surface transport
2. plies - is flexible; goes on a surface; moves
3. sky - atmosphere; not on earth;

As can be readily seen there is overlap of 2 key terms at level 1 of the tree and thus there is less gloss overlap. Thus at level-1 document looks less meaningful.

Document2 : Cars and buses ply on road

Constructing definition graph for level-1 we get,

1. Car - automobile used for surface transport
2. Buses - automobile used for surface transport
3. ply - flexible; go on a surface; move
4. road - asphalted surface used for transport

All 4 keywords overlap giving surface as common token in their respective glosses. Overlap is better than Document1, since more keywords contribute to overlap. Both examples are grammatically correct but one of them is less related semantically.

5.6 Intrinsic merit score, Convergence factor and Relatedness

Definition 3. Let us define Intrinsic merit I to be the product of number of vertices(V), number of edges(E) and Convergence factor(C) of the definition graph of the document.

$$I = V * E * C \quad (4)$$

Convergence factor (C) is the difference between number of vertices in definition tree and number of vertices in definition graph (V). Number of vertices in definition tree includes overlapping vertices without coalescing them (since after coalescence we get the definition graph). Number of vertices in the definition tree = $x^d - 1$ where x is the average number of keywords per term definition and d is the depth of the definition tree of the document. Let us add 1 to this to get x^d (smoothing). Number of vertices in the definition graph = V Thus the Convergence factor C and Intrinsic merit I become,

$$C = x^d - V \quad (5)$$

$$I = V * E * (x^d - V) \quad (6)$$

Intrinsic Merit score can also be further fine-tuned by taking into account the level of definition tree at which first convergence(gloss overlap) happens, defined as firstconvergencelevel. Greater the firstconvergencelevel, more irrelevant the document "looks" (but has a deeper cohesion). Depth to which definition tree has to be grown is decided by extent of grasp needed by the reader. Thus greater the depth of definition tree, greater is the understanding. It is obvious to see that Depth has to be greater than firstconvergencelevel so that some

pattern can be mined from the document.

Heuristically, we can grow the definition tree till intersection of leaves of all sub-trees of the keywords in the document is non-empty. This is the point where we can safely assume that all keywords in the document have been somehow related to one another. So, Intrinsic merit score can be improved by incorporating firstconvergencelevel denoted by f . Thus improved score is

$$I = V * E * (x^d - V) / f \quad (7)$$

(since merit is inversely proportional to firstconvergencelevel). Complexity of constructing definition tree is $O(x^d)$. Since non-unique vertices are coalesced (through gloss overlap), definition graph can be constructed in $O(V)$ time (subexponential). Since x is the average number of children keywords per keyword, $x = E/V$. Substituting,

$$I = E * V * (E^d - V^d) / (V^d * f) \quad (8)$$

As an alternative to convergence factor, gloss relatedness score similar to the one discussed by Banerjee-Ted, but considering only one relation, number of overlapping parents and length of overlap can be used to get the interrelatedness/cohesion of the document. Replacing the convergence factor with relatedness, Intrinsic merit becomes, $I = V * E * Rel / f$ where Rel is the sum of relatedness scores, computed over all overlapping glosses at each convergence level and f is the level at which first gloss overlap occurs

$$Rel = \sum_{i=1}^n (relatedness(Level(i), keyword1, keyword2, \dots, keywordn)) \quad (9)$$

This relatedness score has been generalized to overlap of more than two glosses with single relation R ($R(x,y) = y$ is in definition of x). Function relatedness() for n -overlapping keywords is defined as,

$$\begin{aligned} & relatedness(Level(i), \\ & \quad keyword1, keyword2, \dots \\ & \quad , keywordn) = \\ & \quad OverlapLengthAtLevel(i) \\ & \quad (LinearOverlap) \quad (10) \end{aligned}$$

(or)

$$\begin{aligned} \text{relatedness}(\text{Level}(i), \\ \text{keyword1}, \text{keyword2}, \dots \\ , \text{keywordn}) = n \cdot (\text{OverlapLengthAtLevel}(i)^2) \\ (\text{QuadraticOverlap}) \quad (11) \end{aligned}$$

The relatedness score reflects the convergence since it takes into account the overlapping keywords at each level and length of the overlap. Thus first version of relatedness() function, implies the convergence factor (difference in number of vertices of definition tree and definition tree, signifying overlap) Intrinsic merit/Relatedness score can be used to rank the set of documents and display them to the user. Referring back to examples in 5.5, quadratic relatedness measure (9) above) is a better choice than linear overlap since it is a function of both overlapping parents and the overlap length. The quadratic overlap gives greater weightage to length of overlap by squaring it while keeping the number of parents involved linear.

5.7 Intuition captured by above intrinsic merit score

The number of edges (representing relation between parent term and its definitions) increase as relationship among vertices of definition graph increases. The number of vertices(keywords) in the definition graph increases, as the knowledge represented by the document increases. The depth of the definition tree increases, as the understanding grows. Convergence factor increases as number of overlapping terms in definition graph increases. Similarly quadratic relatedness score increases with number of keywords involved in overlap and the length of overlap, thus pointing to stronger semantic relationship among the keywords. Intuitively, definition graph is WordNet(or any other ontology) projected onto the document.

5.8 Breadth/Depth first search of definition graph and why it is not a good choice for computing merit score

Since Breadth/Depth first search of graph can model human process of thinking, BFS/DFS algorithms can be applied to get the merit score.

Since BFS/DFS algorithms run in $O(V + E)$ time merit score is proportional to $V + E$ - all vertices of the graph are visited in $O(V + E)$ time. But the

drawback of this approach is that strength of underlying theme of the document and cohesion of keywords is not captured by this merit score. Since Intrinsic merit score obtained by Convergence reckons with depth and overlapping keywords, BFS/DFS merit score is discarded

5.9 Sentiment analysis applying Recursive gloss overlap

Recursive Gloss Overlap algorithm after few levels down the definition tree would spell out the sentiment of writer.

Example1: "That movie was fantastic; Graphics was awesome" Keywords at level-1 of Definition graph construction:

1. movie - motion picture; positive
2. fantastic - good, excellent; positive
3. graphics - software technique; positive
4. awesome - good, great; positive

Overlapping terms are {good, positive} and large number of keywords(parents) contribute to this overlap. Thus the document is of extolling nature about some target entity. Prerequisite is a dictionary which annotates each word with the sentiment and sense of the word(Implementation uses SentiWordNet which gives positivity/negativity for each lemma). Sentiment analysis with Recursive Gloss Overlap is applied to finding the polarity of an edge in Citation graph (See (1)). Recursive Gloss Overlap algorithm is applied to each Citation context and a definition graph is constructed. Keyword vertices with more than one indegree are then tested for positivity and negativity using SentiWordNet. If majority of these is positive then polarity for citation edge is positive, otherwise negative.

5.10 False negatives

Convergence algorithm never assigns lower merit score to a document which deserves a higher merit since a document with higher merit explains the concept with more depth/cohesion than document with lower merit. So false negatives do not exist

5.11 False positives

False positives exist since both a document and its arbitrarily jumbled version will get same merit score. This is prevented by assuming grammatically

correct documents or by preprocessor which does parts of speech parsing to validate the grammatical structure of the document.

5.12 Definition graph and Hyperlink graph

Prestige measures obtained from hyperlink graph for a given document are dependent on prestiges of linking documents whereas the Definition graphs are results of human judgements in different viewpoint (e.g WordNet is a result of some experiments done on human judgements). Moreover the hyperlink graph is coarse-grained interconnection of documents and the Definition graph is fine-grained interconnection of words within the same document. Definition graphs are projections of a larger, absolute, universal graph (e.g WordNet). Thus definition graphs depend only on the accuracy of this absolute ontology of which they are subgraphs and definition graphs place one more level of abstraction on the way "judgement" is perceived. We can imagine this to be a two phase process - 1) electing a system which in turn judges documents objectively (e.g WordNet is the elected system) 2) judgement of a document by the elected system (e.g application of WordNet to judge a document as in definition graph construction).

5.13 Normalization

Intrinsic merit can be compared only if the compared documents are of same class. Thus 2 documents explaining special relativity can be compared while a document on journalism can not be compared with a document on special relativity. Intrinsic Merit scores can be normalized by,

$$\text{NormalizedIntrinsicMeritScore} = \frac{\text{Score}}{\text{MaximumScore}} \quad (12)$$

5.14 Ordering and Relative Merit

Definition 4. *Document1 is more meritorious than document2 if*

1. *document1 has more keywords that need to be understood than those of document2,*
2. *cohesion/interrelation of the keywords in document1 is more than that of document2,*
3. *average number of keywords per definition is greater for document1 than document2,*

4. *firstconvergencelevel/level at which first gloss overlap occurs) of document1 is less than that of document2 and*

5. *depth of definition tree of document1 is greater than that of document2.*

If we want a weaker definition of the above, ranking may be a partial order (where some pairs of documents may not be comparable) than a total order. This appeals to intuition since document1 may be better in some aspects but worse in some other relative to document2

5.15 Semantic relatedness or Meaningfulness of a document

Definition 5. *A document is meaningful to a human reader if any pair of keywords in the document are within a threshold WordNet distance e.g Jiang-Conrath distance*

5.16 Formal proof of correctness of Convergence and Intrinsic Merit Score

Theorem 1. *If a document lacks merit, convergence(or gloss overlap) does not occur (Corollary: Document's merit is measured by extent of convergence)*

Proof. By "meritorious" document, we imply a document which is meaningful as per the definition of meaningfulness above (i.e. keywords in a document are separated within threshold WordNet distance metric like Jiang-Conrath distance). Let us denote R as a relation "is descendant of". If xRy then y is in (gloss)definition tree of keyword x (i.e y is descendant of x). If definition trees of keywords of the document are disjoint, then there is no y such that xRy and zRy for two keywords x and z. Let us define the relation S to be "two keywords are related". xSz iff xRy and zRy for some y. Thus we formalise cohesiveness/meaningfulness of a document in terms of definition graph. If a document is not meaningful then there exist no x and z such that xSz , which implies that for no y, xRy and zRy . Thus there exist no vertex y which is in definition tree of two key words. Thus convergence is a necessary condition for merit. The relation S implies that there exists a path between two keywords x and y in the document, through some intermediate nodes which are in the definition/gloss tree of x and y. There exists a threshold WordNet distance greater than length of

this path since the length is finite and whether a document is meaningful depends on this threshold. Thus convergence(generalized gloss overlap) implies meaningfulness of a document as per the definition above. Moreover Intrinsic merit increases with number of edges and relatedness() - linear or quadratic. So with greater relatedness() and more number of vertices and edges, overlaps and number of nodes involved in overlap increase. This in turn implies that more number of paths are available amongst the keywords of the document since every overlap acts as a meeting point of two keyword definition trees. Probability that lengths of these paths are less than threshold WordNet distance is inversely proportional to firstconvergencelevel(level of first gloss overlap) as follows. Probability that a path exists from x to y in the definition graph(P1) =

$$\frac{NoOf(Overlaps(DefTree(x), DefTree(y)))}{TotalNoOf(paths)}. \quad (13)$$

Probability that such an x-y path is less than the threshold WordNet distance (P2) =

$$\frac{NoOf(x - y \text{ paths} < \text{ThresholdLength})}{NoOf(x - y \text{ paths})} \quad (14)$$

Probability $P3 = P1 * P2$ (by conditional probability that there is a path between x-y and such a path is less than threshold length) is proportional to meaningfulness by definition above. With greater the first level in which gloss overlap occurs, the length of x-y path increases for all of the x-y paths penalising meaningfulness, since any x-y path has to pass through such an overlapping vertex due to multipartiteness. Thus intrinsic merit score discussed earlier captures this notion. \square

5.17 Extending the above theorem for general graphs

Above theorem can be extended to general graphs by constraining the longest shortest path (diameter) of any pair of vertices (s,t) of the definition graph to be less than the threshold wordnet distance. But ranking scheme has to be re-invented since above ranking is specific to multipartite definition graphs.

5.18 Worst case running time analysis of Recursive Gloss Overlap algorithm

Let overlap at level i = $OL(i)$ and branching degree = x (=average number of tokens per keyword gloss)

Number of vertices in definition graph

$$V = x + x^2 + \dots + x^z - \sum_{i=1}^z OL(i) \quad (where \quad z = (d - 1)) \quad (15)$$

Running time for:

1. finding overlaps at level i and merge them to single vertex =

$$O(x^k) \quad (where \quad k = 2 * i + 1) \quad (16)$$

2. get tokens =

$$O(x^i - OL(i)) \quad (17)$$

3. remove isomorphic nodes across levels =

$$O(x^k) \quad (where \quad k = 2 * i + 1) \quad (18)$$

Steps 1) ,2) and 3) together have running time $O(x^p)$ where $p = 2d + 1$. But $V = O(x^d)$. Thus running time of recursive gloss overlap = $O(E * V^2)$ since x is upperbounded by V, where V is the number of nodes in Definition Graph and E is the number of edges in Definition graph.

5.19 Parallelizability

Recursive gloss overlap is parallelizable by partitioning the tokens at each level and assigning each subset to different processors (Map) to get the tokens for next level. Individual results from processors are merged (Reduce) to get the final set of tokens for a level. This is repeated for all levels. MapReduce can be applied for parallelism.

6 Interview Algorithm (applying (1) and/or (2) for computing intrinsic merit)

6.1 Motivation for Interview algorithm

Here we map the real world scenario of an interview being conducted on a candidate where a panel asks questions and judges the candidate based on the quality of answers by candidate - candidate is a document and it is "interviewed" by a reference set of authorities. Each document x is interviewed/evaluated by set of reference documents

which will decide on the merit of the document x. Reference set initially consists of n user chosen authorities on the subject. Interview is set of queries made by reference set on the document and evaluating the answer to the queries. If x passes the interview it is inducted into reference set. Next document will be interviewed by n+1 documents including last selected document and so on. Hierarchy of interviews can be built. For example Document x interviews documents y and z. Document y interviews w and document z interviews p. Thus we get a tree of interviews (it could be a directed acyclic graph too, if a candidate is interviewed by more than one reference, one of which itself was a candidate earlier). The interview scores can be weighted and summed bottom-up to get the merit of the root (Analogy: hierarchy in an organization).

6.2 Steps of the Interview algorithm

1. Relevance of the document to the reference set is measured by a classifier (NaiveBayesian or SVM or search engine results for a query)
2. Intrinsic merit score of the document is computed either by Recursive Gloss Overlap algorithm (measures the meaningfulness/sanity of the candidate) (or) by citation digraph
3. Reference set interviews the candidate and gets the score
4. Value addition of the candidate document is measured (what extra value candidate brings over and above reference set)
5. Candidate is inducted into reference set based on the above criteria if candidate is above a threshold.

6.3 Mathematical formulation of an interview

Interview is abstracted in terms of a set of tuples, where each tuple is of the form

$$t(i) = (question, answer, expectedanswer, score) \quad (19)$$

for question i.

$$Interview(I) = \{t(1), t(2), t(3), \dots, t(n)\} \quad (20)$$

$$t(i).score = \text{PercentageOfMatch}(t(i).answer, t(i).expectedanswer) \quad (21)$$

$$\text{if} \left(\sum_{i=1}^n (t(i).score) \right) > \text{referencethreshold} \quad (22)$$

then induct the document into reference set. In the Information Retrieval context, a question is a query and the answer is the context within the document that matches the query. The answer returned by the document is then compared with expectedanswer. Comparison is done by Jaccard coefficient of shingles (n-grams)

$$\begin{aligned} t(i).score = & \\ |\text{shingle}(answer) \cap \text{shingle}(expectedanswer)| / & \\ |\text{shingle}(answer) \cup \text{shingle}(expectedanswer)| & \end{aligned} \quad (23)$$

1. Supervised: In supervised setting, each reference document is pre-equipped with user-decided set of queries and answers it expects. Thing to note is that a document is made a live object - it both has content and questions it intends to ask(set of search queries). Alternative way to compute t(i).score is to find out the definition graph of answer and expectedanswer and compute the difference between the two graphs(e.g edit distance). Downside of this is the assumption of pre-existence of correct answers which makes this a supervised learning.
2. Unsupervised: In the absence of reference questions and answers, questions that a document "intends" to ask can be thought of as set of queries for which the document has better answers(results). These set of queries/results (questions and answers) can be automatically obtained from a document through an unsupervised way by computing set of more likely to be important n-grams(by computing key phrases with tfidf above threshold) and the context of the n-grams in the reference documents. These n-grams/contexts can later be used as "references questions" (n-grams) and "reference answers"(contexts of the corresponding n-grams) to the candidate document. Thus we compensate for lack of reference Questions and Answers. Alternatively, an interview can be simply considered as the percentage similarity of definition-graph(reference) and definition-graph(candidate) obtained by edit distance.

6.4 Searching for answer to a query within the document (as implemented)

If a document describing tourist places is given and the query is "What are the good places to visit in this city?", then query is parsed into key words like "good", "places", "visit" and "city" and matching contexts within the document are returned where context is the phrase of length $2n + 1$ (from $x - n$ to $x + n$ locations with location of keyword being x).

6.5 Value addition measure

Recursive gloss overlap algorithm gives the definition graph of the candidate document. To measure the value addition we can run the recursive gloss overlap algorithm on the reference set to get the definition graph of reference set and find out the difference between the two definition graphs - reference and candidate. Since value addition is defined as the value added which is not already present, extra vertices and edges present in candidate but absent in the reference set are a measure of value addition. Value addition can be measured by either edit distance(cost of transforming one graph to the other after adding/deleting vertices/edges), maximum common subgraph or difference of adjacency matrices.

Implementation uses graph edit distance measure.

6.6 Update summarization through Interview algorithm (applying algorithm given in 6.2)

Given a news summary and a candidate news to be added to summary

1. Label the summary as reference set.
2. Run a classifier on summary and candidate to get the class to which both belong to (or get from search engine results on a news topic)
3. If $\text{class}(\text{summary}) == \text{class}(\text{candidate})$ proceed further
4. Calculate intrinsic merit score of the candidate news document through recursive gloss overlap algorithm described in (2) (or) from citation digraph described in (1)
5. Candidate news is interviewed by reference set (summary in this case)
6. Compute value addition of candidate to summary

7. Add the value added information from candidate into existing summary to get new summary (by getting cream of sentences with top sentence scores)

6.7 Application to Topic Detection, Link detection and Tracking

Interview algorithm can be applied to TAC 2010 topic detection tasks though no runs were done specifically for this purpose.

1. Interview algorithm and graph edit distance measure can be applied to news topic link detection (Answers the question - Does a pair of news stories discuss same topic?). Since same news item falls under multiple topics and is changing over time, topic of a news story is in a state of flux. Given a pair of news stories (n_1, n_2) execute interview of n_2 with n_1 as reference. This interview score decreases and edit distance grows as n_2 becomes more irrelevant to n_1 . By defining a threshold for interview and edit distance scores to belong to same topic, link detection can be achieved. It is important to note that interview score and value addition score are inversely related.
2. At any point in time, compute edit distance for all possible pairs N_x, N_y in a topic (after getting their respective definition graphs) and choose N_y which has largest edit distance to others and hence an outlier and least likely to be in the topic. Thus topic detection is achieved (Answers the question - Does this story exist in correct topic?).
3. Topic tracking can be done by constructing definition graph and finding vertices with high number of indegrees. These keywords are voted high and point to the maximum likely topic of the news story (works as an unsupervised text classifier). This process has to be periodically done since topic of a story might change and thus the definition graph will change.

6.8 TAC 2010 Dataset Evaluation Methodology

1. Split each dataset into two : Reference and Candidate (as described in preprocessing section)
2. Compute the intrinsic merit score for Candidate:

- applying citation digraph construction (or) recursive gloss overlap - recursive gloss overlap was applied since it was difficult to get a citation graph for dataset
 - Parse into keywords and get keywords above a threshold tf-idf
 - Perform WSD using Lesk's algorithm
 - Get glosses of matching sense through wordnet api
 - get overlaps at level i, update intrinsic merit score (either using linear or quadratic overlap)
 - repeat for sufficient number of levels defined by "depth"
3. execute interview if reference questions and answers are available (supervised) or through getting important n-grams/context from Reference by algorithm described above (unsupervised) - at present restricted to 1-gram for keywords and bigrams for jaccard coefficient calculation
 4. compute value addition through definition graph edit distance between reference and candidate, and get the score.
 5. get percentage weighted sum of intrinsic merit, interview and value addition scores and get final score.
 6. **APPLY (2), (3) and (4) ABOVE TO UPDATE SUMMARIZATION:** If final score is above threshold, update the summary with candidate and publish top 5 percent of the sentences (sentence scoring is done by sum of tfidf scores of words in a sentence)

6.9 Results

25 out of 92 datasets were evaluated with interview algorithm described above. Some of the resultant summaries crossed 100-word limit but they were in the top 5 percent of the sentence scores. Results are as published in Guided Summarization Evaluations.

6.10 Conclusion

Results above demonstrate the application of interview algorithm to TAC 2010 update summarization task. Motivation for this excercise is to explore the possibility of finding a framework to assess the merit of a document with and without link graph structure in place with greater emphasis

on the latter. Citation graph maxflow measures the penetration of a concept (represented in a document), in a link graph while the Recursive gloss overlap objectively judges the document without getting inputs from any incoming links. Interview algorithm uses either of these two algorithms and abstracts some real world applications. Moreover the intrinsic ranking scheme given above need not be the only possible way of computing merit. Once we have definition graph for a document (whether multipartite or not), multitude of more ranking schemes can be invented - for example 1) modelling the definition graphs as expander graphs 2) k-connectedness of the definition graph 3) (multipartite) cliques of (multipartite) definition graph etc., Since definition graph construction is computationally intensive, there is a scope of improvement in improving the recursive gloss overlap algorithm by applying some parallel processing framework like MapReduce. Applying Evocation WordNet, implementing a MapReduce(e.g Hadoop) cluster and considering more than one relation are future directions to think about. Theoretical foundation for the recursive gloss overlap comes from WordNet itself which visualises the relatedness of words - Definition graph is just an induced subgraph of WordNet for a document. Accuracy of Recursive gloss overlap depends on the accuracy of WordNet, depth to which definition trees are grown and Word Sense Disambiguation.

7 Acknowledgements

Algorithms discussed in this article were part of author's master's thesis done during December 2009 to July 2010. Author would thank Professors B.Ravindran (Indian Institute of Technology, Chennai, India) and Madhavan Mukund (Chennai Mathematical Institute, Chennai, India) for guiding through and encouraging me to participate in TAC 2010 and above all submit to God for granting intuition.

References

- [1] Graph Similarity, Master's thesis by Laura Zager and George Verghese EECS MIT 2005
- [2] Edit distance and its computation, presentation by Joszef Balogh and Ryan Martin
- [3] Extended Gloss Overlaps as a measure of semantic relatedness, Satanjeev Banerjee and Ted Pedersen

- [4] Sematic Language Models for TDT, Ramesh Nallapati,University of Amherst
- [5] WordNet Evocation Project-
<http://wordnet.cs.princeton.edu/downloads/evocation/release-0.4/README.TXT>
- [6] SentiWordNet - <http://sentiwordnet.isti.cnr.it/>
- [7] WordNet - <http://wordnet.princeton.edu>
- [8] Navigli, R. 2009. Word sense disambiguation: A survey. ACM Comput. Surv. 41, 2, Article 10 (February 2009)
- [9] Temporal information in Topic Detection and Tracking - Juha Makkonen, University of Helsinki
- [10] Overview NIST Topic Detection and Tracking by G. Doddington ,
<http://www.itl.nist.gov/iaui/894.01/tests/tdt/tdt99/presentations/index.htm>
- [11] Topic Detection and Tracking Pilot Study - James Allan , Jaime Carbonell , George Doddington , Jonathan Yamron , and Yiming Yang UMass Amherst, CMU, DARPA, Dragon Systems, and CMU
- [12] The cognitive revolution: a historical perspective , George A. Miller Department of Psychology, Princeton University, TRENDS in Cognitive Sciences Vol.7 No.3 March 2003
- [13] On Bipartite and Multipartite Clique Problems, Milind Dawande, Pinar Keskinocak, Jayashankar M. Swaminathan and Sridhar Tayur,Journal of Algorithms 41, 388-403 (2001)
- [14] Python Natural Language Toolkit -
<http://nltk.sourceforge.net>
- [15] Partitioning CiteSeer's Citation Graph - Revised Version , Gregory Mermoud, Marc A. Schaub, and Gregory Theoduloz, School of Computer and Communication Sciences, Ecole Polytechnique Federale de Lausanne (EPFL), 1015 Lausanne, Switzerland
- [16] Semantic distance in WordNet: An experimental, application-oriented evaluation of five measures, Alexander Budanitsky and Graeme Hirst, Department of Computer Science, University of Toronto
- [17] The Official Python Tutorial -
<http://docs.python.org/tut/tut.html>
- [18] MapReduce: Simplified Data Processing On Large Clusters, Jeffrey Dean and Sanjay Ghemawat, Google Inc.,
- [19] Opinion Mining and Summarization - Sentiment Analysis , Bing Liu Department of Computer Science , University of Illinois at Chicago , Tutorial given at WWW-2008, April 21, 2008 in Beijing WWW 2008
- [20] Web Data Mining, Bing Liu, Department of Computer Science , University of Illinois at Chicago
- [21] Introduction to Algorithms - Second Edition, Thomas H.Cormen, Charles E.Liecerson, Ronald L.Rivest, Clifford Stein

Few algorithms for ascertaining merit of a document and their applications

Ka.Shrinivaasan
M.Sc(Computer science)-II
Chennai Mathematical Institute
shrinivas@cmi.ac.in



Motivation

- Is prestige based ranking perfect?
- Are there alternatives?
- Two judging traditions – majority voting and interactive – which is right? Subjective or objective?
- Can a document be analyzed independently to get its quality?

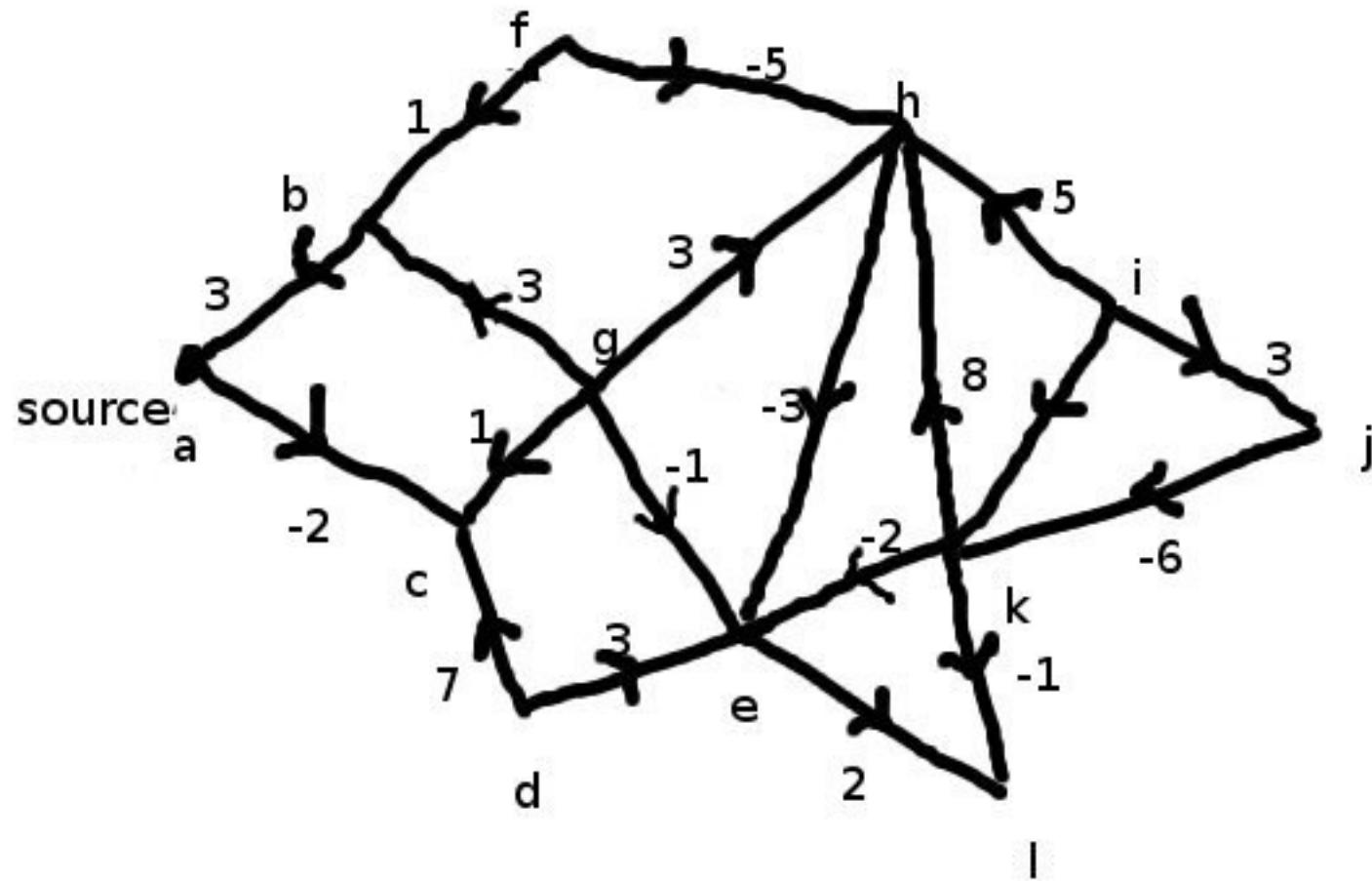
Three Algorithms ...

- **Citation Graph Maxflow and Path Lengths**
- **Definition Graph Convergence (or)
Generalized Recursive Gloss Overlap**
- **Interview algorithm**
- Application to Update summarization
- Application to Topic Detection and Tracking
- Application to Sentiment Analysis

Part I - Directed Graph of Citations

- Merit = influence on future documents = citations
- Construct a directed graph of citations
- Weight of an edge (u,v) = No. Of citations of u by v (is this only way to weight?)
- Polarity of (u,v) = Sentiment Analysis of Citation Context – Positive or Negative
- Number of nodes in all paths of fixed length from source s is a measure of merit (might mislead)

Citation digraph - How it looks



Mincut/Maxflow of Citation DiGraph

- Get Maxflow/Mincut from Ford-Fulkerson algorithm with each distinct vertex pair as (source, sink)
- Mincut of citation graph carries Maximum Flow of the concept from source document s - “most influenced by the source document s ”
- Average Maxflow out of a source s , is thus a measure of merit of s ($= (\sum \text{mx}f(s,t)) / (|V|-1)$)

Part II – Definition Graphs

- “Fruit”
- Evocative - What do we get reminded of after reading the above? (*plant, tree, sweet, taste, food, juice, result ...?*)
- Evocation WordNet

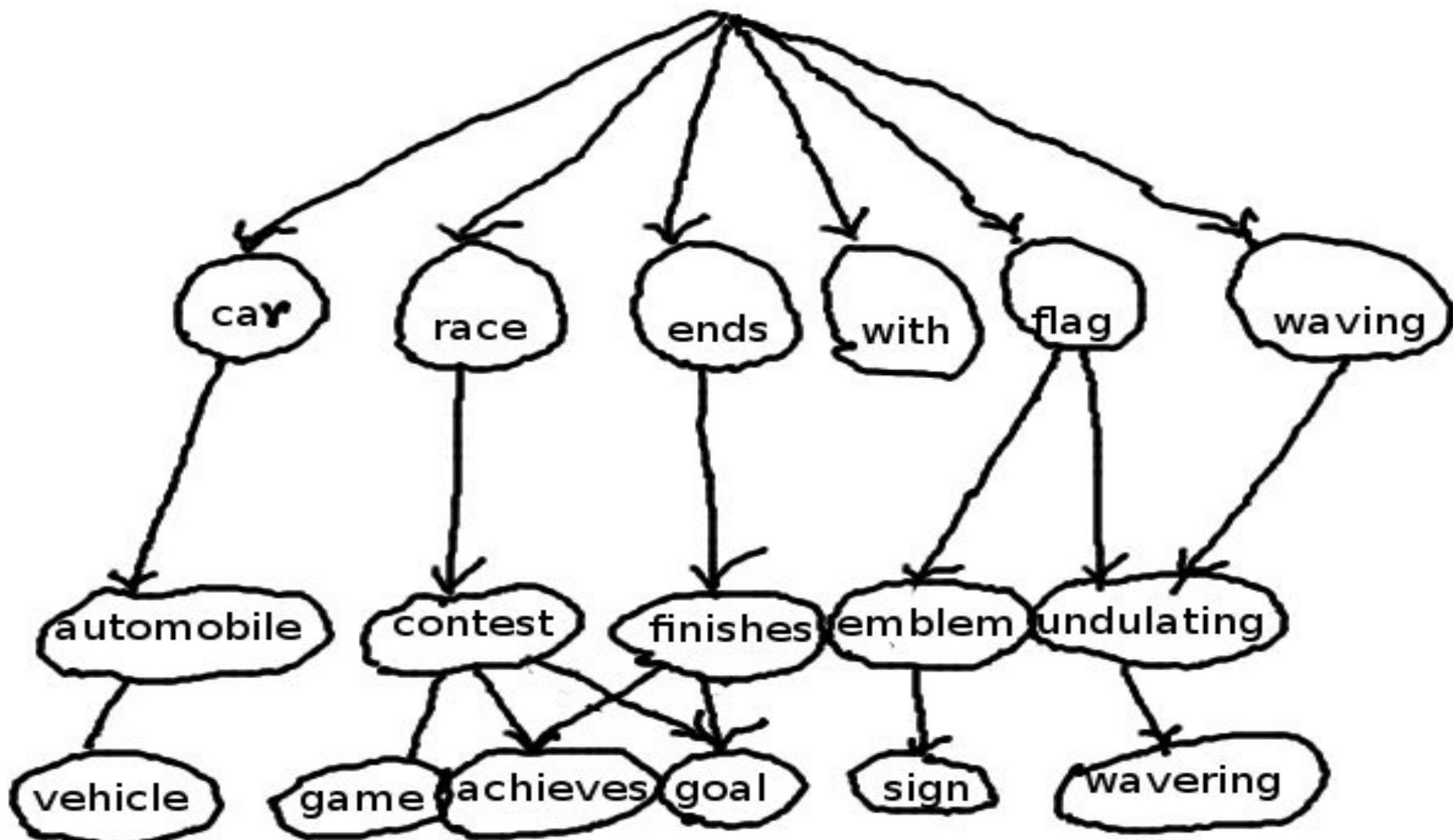
Human thought process and Definition Graphs

- Humans scan through the natural language text
- Relate the keywords – motivation behind WordNet
- **What distinguishes the merit of 2 documents X and Y? Grammatical correctness? No. Both X and Y written equally grammatically. Content and Complexity? Yes. How to measure?**

Recursive Understanding - An Example

- *Document: “Car race ends with flag waving”*
- What is “Car”? Car is an automobile
 - What is “automobile”? Fuel driven Machine
 - What is “Fuel”? Petroleum ...
- What is “race”? Race is ethnic group; contest
 - What is “contest”? Game
 - What is “Game” ? Play ...
- What is “end”? ...
- What is “flag”? ...
- What is “waving”? ...

Previous example visualised



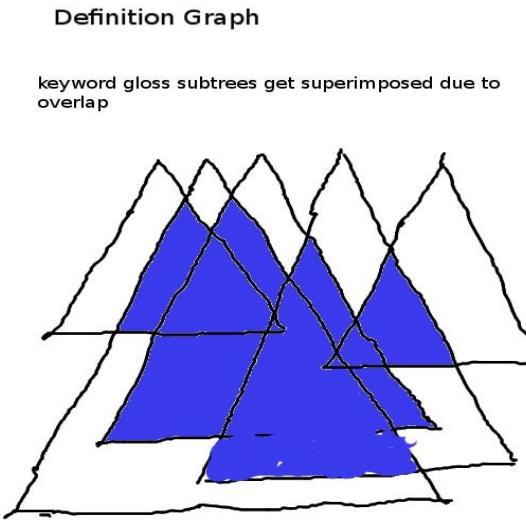
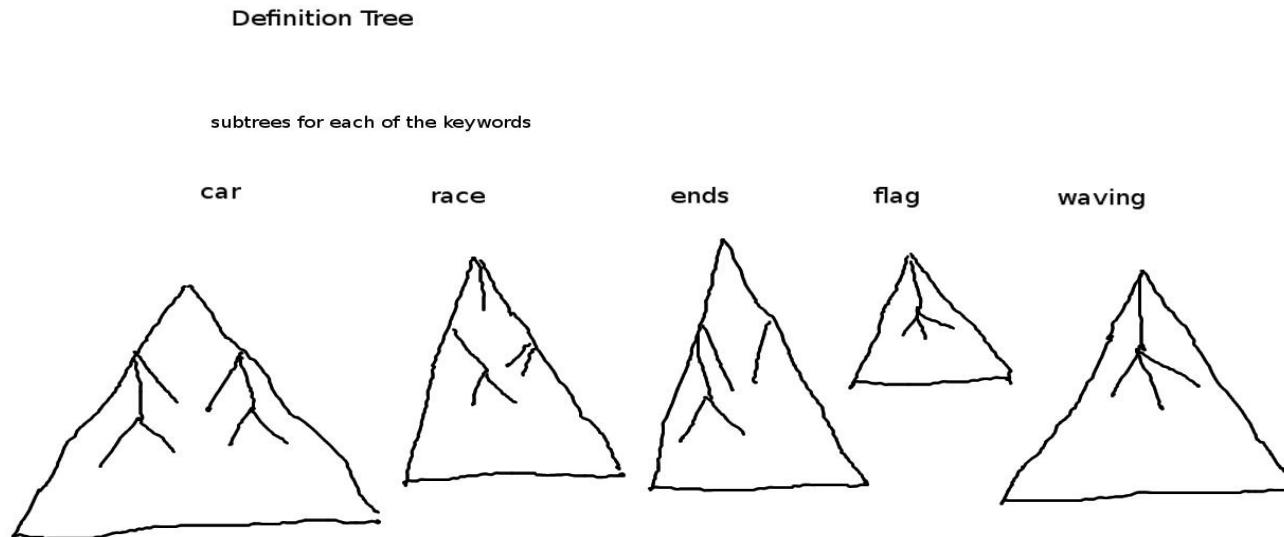
Definition Graph Convergence (or) Generalized Recursive Gloss Overlap

- Meaningfulness: “Meaningful” text has its keywords' Synsets within threshold WordNet distance (e.g Jiang-Conrath)
- WordNet relates words by relations - “is-a”, “has-a” etc., - SYNonymous SETs
- *Map a document to a **subgraph** of WordNet (Definition Trees/Graphs): $F(\text{Document}) = G(V, E)$*

Definition tree and Definition graph

- *DefinitionTree(keyword) =*
DefinitionTree(gkeyword1)
*DefinitionTree(gkeyword2)**DefinitionTree(gkeywor*
d3) ... DefinitionTree(gkeywordn) where
gkeyword1 through gkeywordn are in the
gloss(keyword)
- N subtrees obtained above overlap to form a
graph

Definition Tree and Graph - example



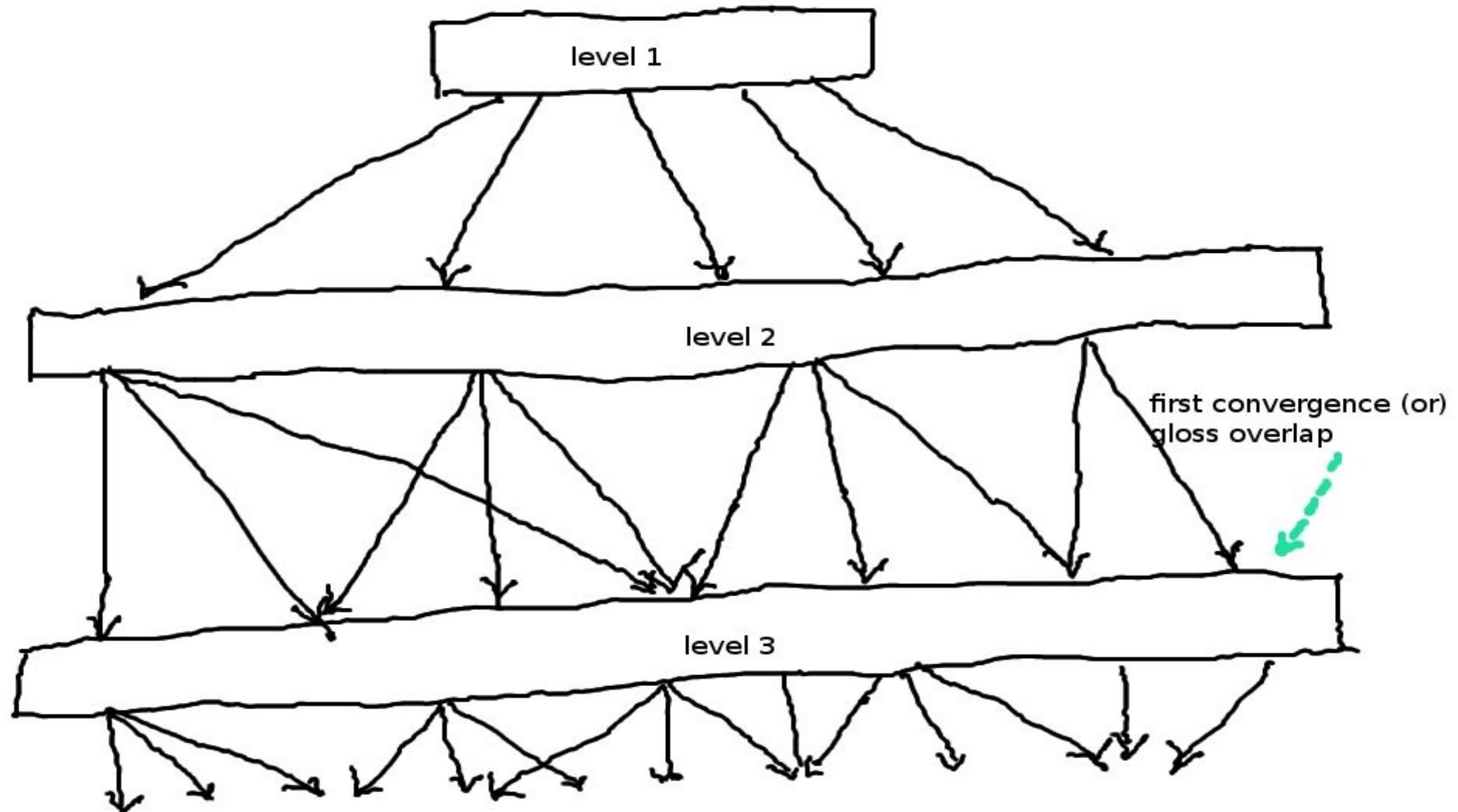
Properties of definition graph

- Definition graph is **multipartite**
- Difference in number of vertices in definition tree and definition graph = convergence factor
- Convergence factor is due to gloss overlap - indicator of relatedness
- Relatedness differentiates 2 documents
- We do not consider grammatical structure

Properties of Definition Graph(contd...)

- Multipartiteness - vertices are partitioned into sets; edges only amongst the sets – useful for preserving recursion level and multipartite-cliques
- Degrees of vertices can be thought of as “votes” for a “theme” keyword – unsupervised text classifier
- Context-sensitiveness still present - Word Sense Disambiguation is done during graph construction

Definition Multipartite Graph Visualised



Recursive Gloss Overlap algorithm

- 1) Get the document as input
- 2) $\text{keywordsatthislevel} = \{\text{keywords from the document through tf-idf filter (implementation uses 0.02)}\}$
- 3) `While (current_level < depth_required) {`
 - For each keyword from `keywordsatthislevel` lookup the **best matching definition(WSD) for the keyword** and add to a set of tokens in next level

Recursive Gloss Overlap algorithm(contd...)

- Remove common tokens with previous levels - an optimization
- Update the number of vertices(*unique tokens*), edges($(x,y) = 'y \text{ is in definition of } x'$) and relatedness (*linear overlap or quadratic overlap*)
- Update keywords at this level

Recursive Gloss Overlap algorithm(contd...)

} //end while

5) Output the *Intrinsic merit score* = $|\text{vertices}| * |\text{edges}| * |\text{relatedness}| / \text{first_convergence_level}$

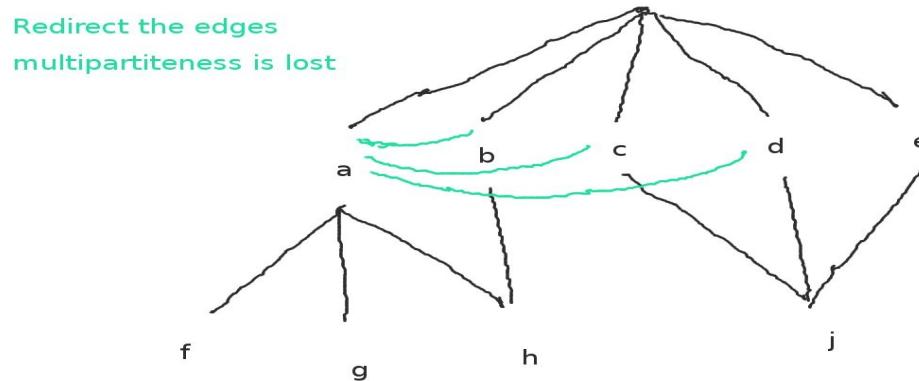
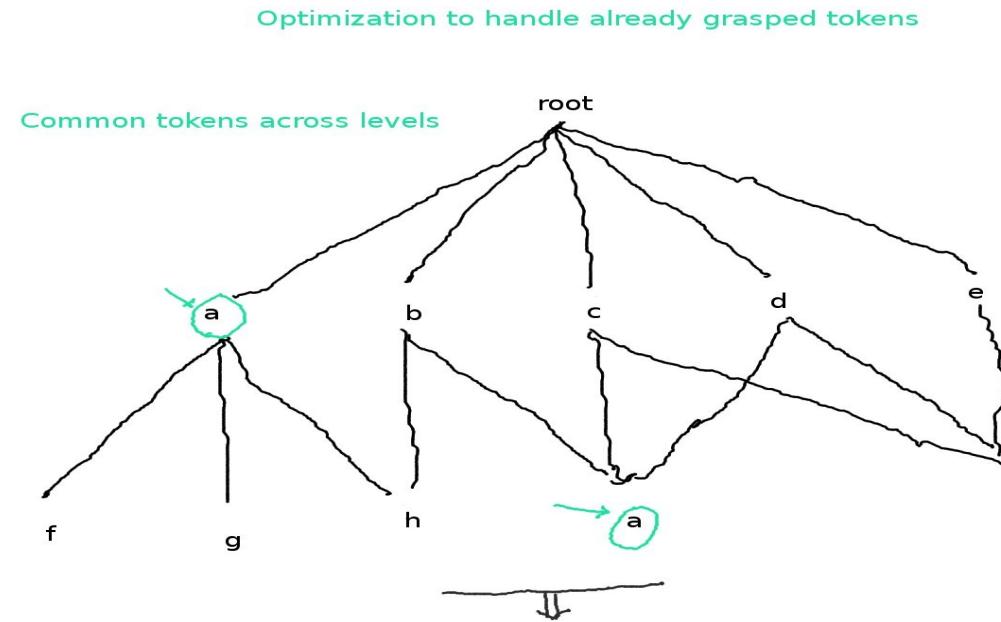
Where

a) *Relatedness* = **number of overlaps** (linear, also called as convergence factor) (or)

Relatedness = **number of overlapping parents * number of overlaps**2** (quadratic)

b) *First_convergence_level* = level of first gloss overlap

Snapshot of Definition graph

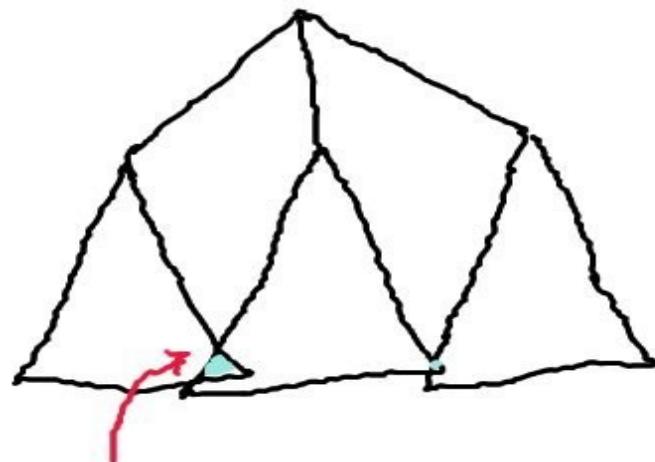


Intuition behind the intrinsic merit score

- vertices ~ knowledge represented by document
- edges ~ relationship among keywords (relation: 'x is in definition of y')
- relatedness ~ complexity quantified by overlap
- first_convergence_level ~ Mingling of definition subtrees
- Above suffice to quantify “meaningfulness” defined earlier (*proportional to V^*E^*R/f*)

Comparing two documents for merit

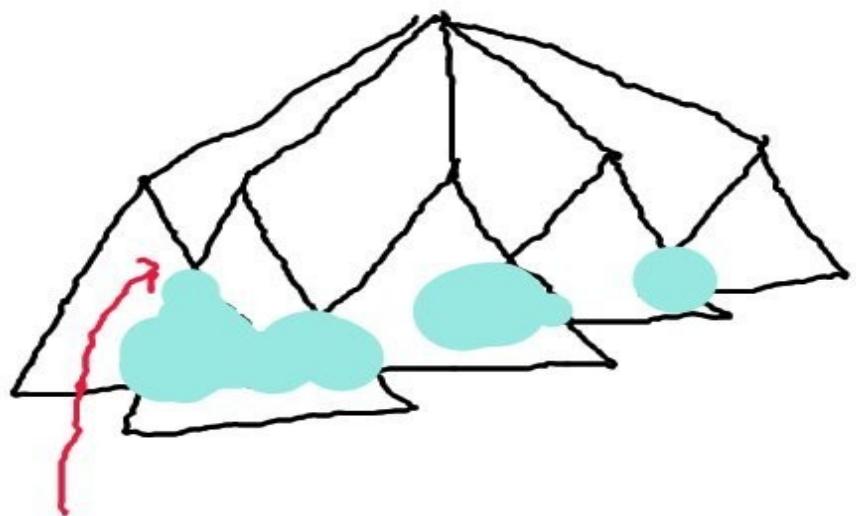
Document1 has less overlap



first convergence level = 5

Example: Car plies on sky

Document2 has more overlap



first convergence level = 2

Example: Cars and buses ply on road

BFS/DFS of definition graph

- Visiting all nodes of definition graph – $O(V+E)$
- But this does not take into account the relatedness
- Worst case complexity of constructing definition graph is $O(\hat{x}^d)$ (*where x is the average size of a keyword definition and d is the depth*)
- For a meaningful document, overlaps bring down this to great extent – no exponential blowup- $O(V)$

Pros and Cons

- No False negatives – more meaningful content will have greater intrinsic merit score than less meaningful document
- False positives exist – two documents with same keywords ignoring grammar will have same merit score.
- Other ranking schemes can be derived from definition graph – based on graph connectedness, completeness etc.,

Application of Recursive gloss overlap to sentiment analysis

- Needed - SentiWordNet - gloss with quantified positivity/negativity score for a keyword
- Example: “*That movie was fantastic. Graphics was awesome*”
- Def Graph level 1: {movie:*motion picture*;+0.1, fantastic:*great*;+0.7, graphics:*software technique*;+0.05, awesome:*great*;+0.7}
- Polarity of Overlap – {great} with positivity score +0.7

Parallelizability of Recursive Gloss Overlap

- Def Graph construction parallelizable - set of tokens of each level broken into subsets
- Assign each subset to a processor (Map)
- Get the results of gloss lookup for subsets and merge them (Reduce)
- To do – Apply MapReduce framework to Recursive Gloss Overlap – E.g Needs a Hadoop cluster

Part III – Interview Algorithm

- Reference “interviews” the candidate – both are documents
- Candidate is inducted into reference if the interview score is above threshold
- Interview is less invasive compared to definition graph construction
- Tree/Graph of interviews can be built (transitive)
 - e.g x interviews (y,z), y interviews w, z interviews p

Interview Algorithm (contd...)

- Intrinsic merit of candidate measured by either a) Citation Digraph or b) Recursive gloss overlap algorithms
- Interview - a) **supervised** (reference Q&A available) or b) **unsupervised** (reference Q&A are computed from reference – 'Q's are keywords / 'A's are contexts)
- Interview is the set of tuples = $\{t(1), t(2), \dots, t(n)\}$
 $t(i) = (question, answer, expected_answer, score)$

Interview Algorithm (contd...)

- Total interview score = $\sum(t(i).score)$ (where
 $t(i).score = |\text{shingles}(\text{answer}) \cap \text{shingles}(\text{expected_answer})| / |\text{shingles}(\text{answer}) \cup \text{shingles}(\text{expected_answer})|$
- Value addition = edit distance of
DefGraph(Reference) and DefGraph(Candidate)
(where $\text{EditDistance}(G, H) = |\text{edges added}| + |\text{edges removed}|$ to transform G into H)
- Final score = $w1 * \text{intrinsic_merit} + w2 * \text{interview_Q\&A_score} + w3 * \text{value_addition}$, where
 $w1, w2$ and $w3$ are weights

Application to Update summarization

- Fix a news summary as reference which has to be updated
- Fix the candidate news items
- Go through the Interview algorithm and get scores for candidates
- Choose the best candidate and update the summary after sentence scoring

Application to Topic Detection and Tracking

- 1) Interview score(n_1, n_2) decreases and editdistance(n_1, n_2) increases as n_2 becomes more irrelevant to n_1 . We have **link detection** - (Do two news stories discuss same topic?)
- 2) definition graph edit distance score for all possible pairs (N_x, N_y) in a topic and choose N_y with maximum pairwise distance(outlier). - **topic detection** (Does this story exist in correct topic?).
- 3) **Topic tracking** can be done by periodically constructing definition graph and finding vertices with high number of indegree. These keywords are voted high and point to the topic of the news story (**unsupervised text classifier**).

Test Results – Spearman Coeff Ranking for RGO Intrinsic Merit

- Spearman coefficient, correlations between Google ranking and Recursive Gloss Overlap IM score(quadratic overlap) are **73%, 4%, 9% and 25% for few Google queries**
- Spearman coefficient correlations between human ranking and Recursive Gloss Overlap IM score(quadratic overlap) are **38% and 90% for 2 judges and 1 judge respectively**

Test Result – Citation Graph Maxflow

- Link graph with 7 html files with some product review comments
- Average concept maxflow out of each page:
 - 'file2.html': 3.7142857142857144
 - 'file4.html': 3.2857142857142856
 - 'file5.html': 2.0
 - 'file3.html': 3.4285714285714284
 - 'file7.html': 0.0
 - 'file1.html': 3.4285714285714284
 - 'file6.html': 0.0
- File2 has greater average maxflow – implies that concept flowing out of file2 is maximum

Few Non-trivial Questions and Shell Turing Machines

Ka.Shrinivaasan (ka.shrinivaasan@gmail.com)

July 8, 2012

Abstract

In this article a fundamental philosophical question about nature of reality is raised and a new theory of Shell Turing Machines with dimensions is proposed to explain this phenomenon of reality and its multiple levels. Also a theory of an all pervasive Shell Turing Machine named Field Turing Machine is introduced (different from Universal Turing Machine) and few outstanding mundane questions about nature are shown to be undecidable by applying constructions based on Field Turing Machine.

1 Intoduction

Right from the beginning of creation, there has been a persistent debate on some deep questions about reality and search for answers to them. Thus if we consider the mundane philosophical questions - like "Does God exist?", "Are there physics experiments which lead to all possible truths in the universe?" - a conceptual abstraction is proposed based on Turing Machines to formulate these questions as special cases of a generalized fundamental question and its undecidability is proven (which places a theoretical limitation on mankind's ability to answer them). The physical reality is limited by spacetime dimensions. For example an object on 2-dimension has less knowledge and information than an object on 3-dimension with volume being a distinguishing factor. In this article an effort is made to generalize the concept of dimensions (as not just restricted to spacetime) and integrate that into Turing machines and apply this to answer fundamental questions.

2 Few Definitions Associated With Shell Turing Machine

Definition 1. *We define a logical dimension as any new variable that adds knowledge to an existing algorithm (or) algorithm gets more computational power and emits more knowledge when this dimension is added to existing set of dimensions.*

Definition 2. *Let us define a Shell TM($Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject}, n$) where first 7 parameters of the tuple have usual meaning from traditional turing machine and the eighth parameter is the maximum dimension this TM can transition to. Let us denote it TM(n). Simply TM(n) operates in a maximum logical dimension n where a turing machine TM is value-added with a dimension n as an additional parameter and a limited knowledge domain determined by the dimension (as described by previous definition). This Shell Turing Machine has transitions across these n dimensions and any transition takes additional parameter of dimension. For example a state transition can be $\langle state1, tapeinput1, dimension1 \rangle$ to $\langle state2, left(or)right, dimension2 \rangle$ which means that on input1 the TM moves from state1 of dimension1 to state2 of dimension2 in addition to moving left or right on tape. A Shell Turing Machine of dimension X has more computing power than a Shell Turing Machine of dimension $X - 1$*

Definition 3. *We define Reality as a set of entities and their interactions and events happening within a maximum box universe of logical dimension N (which is not engulfed by another box universe and there is no $M > N$)*

Definition 4. *We define Unreality as a set of entities and their interactions and events happening within a box universe of logical dimension P (which can be engulfed by another box universe and there is some $M > P$). Important thing to note is that we realize that something is unreal only if we exit the Unreality and events in Unreality can only be observed and not written-to by a universe in which unreality is embedded. (Example: waking up from a dream). Thus we can rephrase Unreality as a Reality in a lower logical dimension.*

Definition 5. *Let us define Unreal reality as a set of entities and events involving those entities which is real in some universe(or)context of lower dimension but unreal in some other universe(or)context of higher dimension*

Definition 6. *Let us define Real unreality as a set of entities and events involving those entities which is unreal in some universe(or)context of higher dimension but real in some other universe(or)context of lower dimension*

3 Some examples

1. A colloquial reallife example: If we consider a movie as a shell, the sequence of events in a movie can be encoded by turing machine states and same physical laws apply. But the audience of the movie which can be encoded as another turing machine cannot interact with the movie turing machine(movie turing machine is read-only). Thus we have a pseudoreality embedded within a reality. Extrapolating this, a movie within a movie is a pseudopseudoreality and so on. We can say that a movie turing machine is enacted in a universe of logical dimension of atleast 1 less than the reality turing machine.
2. Another example: When two mirrors are facing each other, we see an avalanche of mirror embedded within mirror embedded within mirror and so on, on each physical mirror.

4 Paradox or is it?

We now ask a question which is an etymological paradox (might sound as an erroneous sentence on first read) and try to answer:

$$\text{Real Unreality and Unreal Reality: are these two equal?} \quad (1)$$

5 Reality and Unreality - viewed through a thought experiment

1. Let us consider a thought experiment to demonstrate above paradox: The real life events of a family are videographed and streamed in real-time on to a TV set. Thus TV plays the events in realtime as they happen in reallife (The mirror experiment above can be related to this). Thus same events(or)algorithm are enacted in both reallife and by the TV. TV has a logical dimension atleast 1 less than the physical reality. Physical reality is represented by a Shell turing machine $TM_1(N)$ and the movie turing machine(which can be thought of as TV in this example) is represented by a Shell turing machine $TM_2(P)$ where $N \geq P + 1$

2. An Example of Unreal reality: above realtime movie is real for the characters within the movie turing machine universe but unreal in audience turing machine. Thus it is real yet unreal though both enact same events (simultaneously almost). Here direction of perception of events is from a higher dimension to a lower dimension.
3. An Example of Real unreality: above realtime movie is unreal for audience turing machine but real for the characters inside the movie turing machine. Thus it is unreal yet real though both enact same events (simultaneously almost). Here direction of perception of events is from lower dimension to higher dimension.
4. Thus we get a tree of shell turing machines where shell of dimension n is a parent of shells of dimension $n-1$
5. Now we can frame the paradox with logical dimension information as,

Is $\text{Real}(S)$ $\text{Unreality}(P)$ equal to $\text{Unreal}(Q)$ $\text{Reality}(R)$?

where S, P, Q and R are logical dimensions. Thus from above description of some examples real unreality and unreal reality and assuming execution of the same algorithm across dimensions we formalize this as giving a turing machine encoding as input to the same turing machine and stratify the reality by dimensional information. Important to note here is that we have separated a same Turing Machine executing same algorithm into two Turing Machines $TM_1(X)$ and $TM_2(Y)$ but differing only in their dimensions X and Y ($X \geq Y$).

6 What is Equality of Real Unreality and Unreal Reality?

Question of equality loses its meaning unless dimensions and algorithm executed by the two Turing Machines are spelt out clearly and depends on definition of equality

1. If we define equality as whether two turing machines execute same algorithm then they are equal .
2. If we define equality as whether two turing machines have same logical dimensions (despite executing same algorithm) then they are unequal. Thus the question depends on definition of equality.

7 Conclusion on above question

Real unreality and unreal reality and their equality depend on dimensional information and on definition of equality. Thus the paradox above gives an insight into stratification of Reality

8 Definition and a Hypothesis of a Field Turing Machine

Definition 7. *We define a Primordial Field Turing Machine(a special Shell Turing Machine) as a non-deterministic*

1. *Turing machine which enshrines all possible algorithms(or knowledge) in itself and pervades beyond them.*

2. Turing machine which exists in multiple dimensions as per the definitions of Shell Turing Machines (discussed previously) and can have transitions in all those dimensions. It is also the turing machine of maximum possible dimension.
3. Turing machine which can create descendant Turing Machines (who themselves can create descendants and create a tree of turing machines) and every descendant Turing machine has a set of "freewill" state transitions that interact with other turing machines where extent of "freewill" is decided by its dimension. Every turing machine descending from Field turing machine has transitions in dimensions always less than the maximum dimension of the Field Turing Machine.
4. Turing machine whose configuration graph is a superset of union of configuration graphs of all possible turing machines i.e $\text{configGraph}(\text{FieldTM}) \supset \bigcup_i \text{configGraph}(\text{TM}_i)$.
5. Turing machine which judges the descendant turing machines on their freewill transitions and rates them on how these freewill state transitions interfere with the freewill transitions of other descendant turing machines i.e freedom of a turing machine ends where the freedom of other turing machine starts. Based on this rating Turing machines are promoted or depromoted to higher or lower dimensions respectively by the Field Turing Machine. Descendant Turing machines gain (or experience) more knowledge by their interaction with other Turing machines through "freewill" transitions (more discussion on this is given later).
6. Turing machine whose configuration graph can be thought of as a multi-planar graph where each plane corresponds to a logical dimension. Thus a subgraph representing a descendant turing machine can have a transition from a state s_1 in lower plane of the configuration graph to a state s_2 in higher plane of the configuration and there can be state transitions from s_2 to all nodes in a larger subgraph of lower plane of the field turing machine. Stated otherwise, a transition to a higher dimension increases the "visibility" or "accessibility to greater knowledge"

9 Knowledge gained by freewill interactions

Knowledge gained by descendant Turing machines through freewill interactions is directly proportional to already inherent knowledge in the Turing machines and to the extent of freewill interactions. To put it otherwise, $\text{Knowledge of a Turing Machine} = \text{Intrinsic Knowledge of Turing Machine} * e^{\text{NoOfFreewillInteractions}}$

10 Question by a descendant turing machine

A descendant turing machine "asks" for a proof of the existence or non-existence of Field turing machine. Is this problem decidable? As an example a turing machine operating on 2 dimensional space tries to prove or disprove the existence of 3rd dimension which is, for example, volume. [Another analogy is a tree of unix shells where the root shell creates a tree of sub shells each with their own environment variables. A parent shell has the choice of exporting its variables to its descendant shells. Now one of the descendant shells tries to prove the existence or non-existence of its ancestors without any additional information provided to it. Suppose there exists a turing machine which proves or disproves the existence of ancestor shell. Is this problem decidable? An algorithm operating at dimension x will output that there is no ancestral shell turing machine to

shell S. An algorithm or Turing machine T at dimension $x+1$ will output that there is a descendant shell S with dimension x. Thus a contradiction is reached. So there is no turing machine to prove the existence or non-existence of an ancestor (e.g Field turing machine) turing machine.]

By above definition of Field Turing Machine, we can think of the field turing machine's configuration graph as the superset of union of configuration graphs of all possible turing machines. Thus every descendant turing machine's configuration graph is a subgraph of a giant infinite configuration graph of field turing machine. Thus if there exists a turing machine which answers the question "Does a field turing machine, which overloads all turing machines, exist?" then such question is equivalent to asking "Is the configuration graph of this turing machine a subgraph of an all-pervading turing machine's configuration graph?". To answer this question the turing machine 1) should take as input all configuration graphs of all possible turing machines and aggregate them to get the approximate configuration graph of field turing machine (approximate because the field turing machine might have transitions in higher dimensions not reachable and beyond this union) and 2) as next step test if the configuration graph of this Turing machine is a subset of configuration graph of Field Turing Machine.

Theorem 1. *Total number of possible turing machines, of same dimension n, is infinite.*

Proof. Given a turing machine $TM_1(n)$ of dimension n as input construct a new turing machine $TM_2(n)$ whose start state transitions to the start state of input turing machine and end state of the input turing machine has a transition to the end state of the new turing machine. Thus this construction churns out countably infinite turing machines - with some similarity to cantor set and godel numbering). \square

Hence this turing machine might loop (recursively enumerable) and never halt on the input thereby making the subset question undecidable. Moreover such a Turing Machine can enumerate configuration graphs of Turing machines of dimensions less than or equal to n. This Turing machine cannot enumerate configuration graphs of Turing machines of dimensions greater than n (since it is prohibited by definition of Field Turing Machine above)

Alternatively, each turing machine in the union above can have transitions in multiple dimensions. Since the field turing machine by definition has the greatest dimension, a turing machine (with a lower dimension) which asks and answers the question "is this configuration graph of TM a subgraph of Field Turing Machine's configuration graph?" would obviously have no transitions in higher dimensions than itself and is thus an impossibility, ruling out this TM. Thus questions like "Does God exist?", "Are there physics experiments which lead to all possible truths in the universe?" are undecidable and they can be described as special cases of above. This is because Field turing machine can be thought of as a theoretical definition of a Supereme Being and any entity asking the question is a descendant turing machine. This is not a circular reasoning due to the fact that the very definition of Field Turing Machine obstructs the paths to proving it since any algorithm answering it should consider all possible turing machines and should have the ability to transition to all possible dimensions. Since the definition of Field Turing Machine theorizes that FTM has dimension always greater than any of the descendant turing machines, any descendant turing machine which has a lower dimension than Field Turing Machine is thus prohibited from a state transition that would lead it to the proof of existence or non-existence. Thus Shell Turing Machines are a valuable construct to answer this question. Thus a theoretical limit is placed on ability to answer certain non-trivial questions as above. Above informal argument is formalized in theorem below.

Theorem 2. *A statement provable by a shell turing machine of dimension M cannot be proved by a shell turing machine of dimension N where $M > N$*

Proof. An mathematical or scientific proof of a theorem or hypothesis is a sequence of logical implications like $s1 \Rightarrow s2 \Rightarrow s3 \dots \Rightarrow sn$ where $s1, s2, \dots$ are the statements of the proof. If this shell turing machine works in a maximum of N dimensions then each of these n statements is a function of at most N dimension. If on the contrary some statement is a function of more than N dimension then that statement does not belong to a Turing machine of N dimension and is unprovable within N dimensions and should belong to a turing machine of more than N dimensions. \square

11 Acknowledgement

I dedicate this article to God.

12 Bibliography

References

- [1] Inspired by a Hindu religious philosophy of Maya where every participatory being in Maya thinks that Maya is real but the Supreme Soul outside the Maya enacts the Maya and thus has more dimensions than the Maya itself not realizable by beings part of Maya and liberation from Maya is called Moksha
- [2] Inspired by Turing's Halting problem undecidability result -particularly the gadget used for it which inputs an encoding of a Turing Machine to the same Turing Machine
- [3] Einstein's quote: A problem cannot be solved by same level of thinking that created it
- [4] Complexity theory - Sanjeev arora and Boaz Barak
- [5] Theory of computation - Michael Sipser