Course Authored By:
##
##
K.Srinivasan
Personal website(research): https://sites.google.com/site/kuja27/
NeuronRain Documentation,FAQ,Licensing: http://neuronrain-documentation.readthedocs.io/
##
##
This is a non-linearly organized, code puzzles oriented, continually updated set of course notes on Cloud computing frameworks and BigData analysis.
---

7 Februrary 2017

Apache Spark is a Cloud computing software for processing bigdata. It is based on concept of Resilient Distributed Datasets (RDD) which are partitions of a dataset executed in parallel. Spark is divided into 2 components: 1) Driver and 2) Executor. Driver splits the dataset into RDDs and allocates each RDD to an executor in parallel. Parallelization can be in two ways: 1) For objects like lists,arrays etc., 2) For data in HDFS,cassandra, S3 etc.,

While executing in parallel, there is a necessity to share mutable state across executors. This is done in two ways: Broadcast variables and Accumulators (only increment is allowed). Spark streaming is a feature that allows realtime processing of streaming data by an abstraction of Discretized Streams or DStreams. Following code in neuronrain asfer receives generic data from any URL, does an ETL on it and stores in RDDs: https://github.com/shrinivaasanka/asfer-github-code/blob/master/java-src/bigdata_analytics/spark_streaming/SparkGenericStreaming.java

There are 2 operations performed on RDDs: 1) Transformations - create a new set or subset of RDDs 2) Actions - do some iteration on transformed RDDs. Spark streaming allows custom streaming by implementing/overriding receive() method in Receiver interface. Receiver can be started and stopped by onStart() and onStop() methods. Receive method is overridden with customized code to access a remote URL, fetch HTML, parse it and do any ETL operation as deemed fit. From Spark 2.0.0 , support for lambda functions (new feature in Java 8) instead of *.function.* (Function objects) for RDD transformations has been included. Previous Spark Streaming code demonstrates these features and uses Jsoup GET RESTful API for ETL/scraping of remote URL data.

20 February 2017

Spark SQL + Hive (Shark) provides synergy of bigdata processing with an SQL storage backend. Hive is implemented on top of Thrift RPC protocol

which is modern version of Interface Definition Language based Web Service Architectures like CORBA, Google Protocol buffers , SOAP etc., Streamed data received is an iterable (e.g lines in SparkGenericStreaming.java implementation in https://github.com/shrinivaasanka/asfer-github-code/blob/master/java-src/bigdata_analytics/spark_streaming/SparkGenericStreaming.java) which is further transformed with map/filter operations quite similar to Java Streams. Java Streams work on similar concept of creating a stream from iterable (arrays, lists etc.,) and applying map/filter transformations. Spark's saveAsTable() saves the streaming data into a hive table in Spark Metastore or Hive metastore (this requires hive-site.xml in Spark conf directory). (MAC currency in AsFer+KingCobra electronic money cloud perfect forwarding move is implemented on Protocol Buffers.)

18 January 2018

Spark cloud processing framework has support for global variables in two flavours: 1) Accumulators and 2) Broadcast variables. Both of these are mechanisms to reflect global state across Resilient Distributed Data Set nodes in Spark clusters. Accumulators have a single operation add() which incrementally adds a value on a local RDD to the global accumulator variable and is reflected across all nodes in Spark cluster. Both Accumulators and Broadcast variables are instantiated from Spark Context. Broadcast variables are plain read-only global variables which are broadcast as the name suggests to all RDDs in Spark cluster. An example code and logs for how accumulators and broadcast work has been demonstrated in code/Spark_Broadcast_Accumulator.py and code/testlogs/Spark_Broadcast_Accumulator.log.18January2018. Accumulator constructor can be optionally passed on an object of type AccumulatorParam (or its subclassed types which override add()). Presently accumulators and broadcasts are only way to provide global state across nodes in Spark cluster.

---
4 October 2018 - Representational State Transfer - CRUD - RESTful and WebServices in Cloud
---
Traditional Client-Server Architecture in Distributed Computing involves client making a socket connection to
a listener server, completing a handshake and establishing a two-way message transport.Over the years, with the
advent of cloud, every application on web is deemed to be a finite state automaton of 4 states - Create-Read-Update-Delete (CRUD) respective HTTP primitives being PUT,GET,POST,DELETE which create a resource in server, update it, read it and delete. Every resource is identified by a URL or WebService. Though this indirectly wraps the underlying socket communication, benefit is in statelessness of each request - every request is independent of previous request and state is remembered only in client side and server is state oblivious. Nomenclature RESTful stems from the state getting transferred from client to server for every HTTP request and responded in JSON objects. An example of RESTful API is Facebook Graph API SDK for retrieving user profile information, connections, comments, likes etc., A REST Python Client which GETs/PUTs objects to Facebook wall has been described in code/GRAFIT_automatic_wallposter.py. It internally issues HTTP requests to Graph API REST endpoints. Invocation to put_object() has been commented. This is an updated version of https://github.com/shrinivaasanka/asfer-github-code/blob/master/python-

src/Streaming_FacebookData.py specific to GRAFIT (for Grafit Open Learning facebook profile https://www.facebook.com/shrinivaasan.ka which imports @NeuronRain_Comm - https://twitter.com/neuronrain_comm - automatic commit tweets). RESTful implies every cloud distributed computation is a string from alphabets {GET, POST, PUT, DELETE} amongst the nodes in cloud URL graph.

---

5 December 2018 - Apache 2 Web Server, Apache HTTPD modules, Configs and Hooks, Application Servers - example

---

Apache webserver provides facilities to plugin user developed modules. One specific standard example is mod_ssl which is an SSL plugin module for Apache webserver. An example Apache module implementation and related forum Q&A have been cited in the references. This implementation was a part of Sun Microsystems/Oracle iPlanet Application Server which had a 3-tier J2EE compliant middleware architecture (Clients <-> WebServer <-> ApplicationServer). Application Servers are ancestors of present day cloud implementations which are Service Oriented Architectures. HTTP requests are served by Apache Webserver and responded. Necessity for an Apache module arises when Apache webserver is used as a loadbalancer for a cluster of application servers and requests have to be routed to it. Example module snippet in the reference defines config parameters for Apache webserver - the loadbalancer XML file containing details about cluster of iPlanet Application Servers and Locale. The register_hooks() function has callback functions for init, name translation (e.g. URL rewriting for session ids and cookies), and handle requests (e.g routing requests to another application server). These config parameters are set by invoking ap_set_string_slot() functions denoted by assignment to take1 structure member (function takes 1 argument)

References:

1.Apache Modules Mailing list - Apache 1.3.27 and iPlanet Application Server - https://marc.info/?l=apache-modules&m=105610024116012 (Copyright: Sun Microsystems/Oracle)
2.Apache Modules Mailing list - Apache 2.0.47 and iPlanet Application Server - https://marc.info/?l=apache-modules&m=106267009905554 (Copyright: Sun Microsystems/Oracle)
3.Apache Config Directives - ApacheCon - http://events17.linuxfoundation.org/sites/events/files/slides/ConfigurationDirectiveAPI.pdf

---

752. (THEORY and FEATURE) 2 January 2019, 9 January 2019, 8 January 2020, 22 January 2020, 23 January 2020, 29 January 2020, 31 January 2020, 2 February 2020,4 May 2020,12 May 2020,15 May 2020, 2,3,4,7 July 2020,28 October 2020,3 November 2020,2 December 2020,5 December 2020 - Searching, Indexing, Computational Geometric point location queries, Factorization in Randomized NC - Binary Search Nuances, B-Trees, Sharding - related to 227,666,730,864,1123 and all sections of People Analytics,Histogram Analytics,Social Network Analysis,Majority Voting,Syllable vectorspace text analytics and Computational Geometric Planar Point Location Factorization in NeuronRain Theory Drafts

---

Traditional Binary Search still remains the standard for sifting huge datasets and has undergone significant
refinements. Usual Binary Search is centred around midpoint computation and branching:

```
midpoint = (left + right)/2
if query > midpoint:
      search interval [midpoint, right]
else:
      search interval [left, midpoint]
```

Bug in Midpoint computation by previous averaging in some earlier versions of languages like Java caused an overflow error in 32-bit architectures which necessitated replacing it by >>> operator (unsigned right shift):

```
midpoint = (right-left) >>> 1;
```

B-Trees are generalized Binary Search Trees in which each internal and root nodes contain more than one element and every element of a node acts as a separator for values of its subtrees as in example:

```
                    2,10
                    |
              1 ----- 3,9 ---- 11,12
                    |
                    4,5
```

B+-Trees extend B-Trees by additional linked-list of leaf nodes.

Indexing Bigdata e.g web pages involves storing them as huge set of key-value pairs of the form:

```
word  --- (document1, location1), (document2, location2) ...
```

which is termed as posting in inverted indices. Size of an Index in search engine could be unimaginably large requiring partitions into set of rows located across geographically distant servers on cloud. This is horizontal partition based on rows as against vertical partitions of columns in DBMS Normalizations. Each partition is named a Shard.

Fundamental question is: can items in a list be found efficiently by a search graph instead of binary search trees. Lower bound for search is Omega(logN). Binary Search Trees can be alternatively defined as recursive bipartite graph which is an indefinite recursive fractal bipartition of sets - Example:- Set of integers [2,4,5,1,3,7,9,10,8,6] are represented recursively as bipartite graph by assuming an initial midpoint pivot separator of each subset similar to quicksort which partitions a set into two halves of elements > pivot and elements < pivot in each depth of recursion:

```
pivot 6
[2,3,5,1,4] ----- [7,9,10,8,6]
pivot 3                pivot 8
[[4,5,3]--[2,1]] ---- [[6,8,7]--[10,9]]
pivot 4                pivot 7
[[[4,3]-[5]]--[2,1]] ---- [[[7,8]-[6]]---[[10]-[9]]]
pivot 4          pivot 2               pivot 7
[[[[4]-[3]]-[5]]--[[2]-[1]]] ---- [[[[7]-[8]]-[6]]---[[10]-[9]]]
```

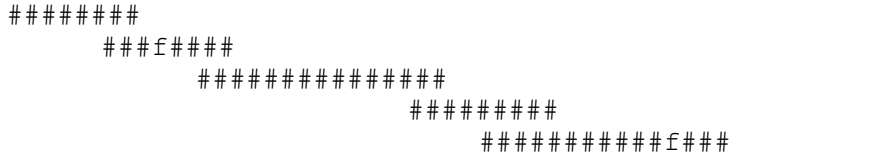
Edges are denoted by ----- which connect each parenthesized subset vertex for a subtree obtained by traversing the binary search tree. Last line encodes a recursive bipartite search graph which is the top view of the binary search tree and is a multidimensional tensor - a kind of binary space partition. Searching this tensor locates a point in the nested parentheses by comparing against pivots. Advantage of recursive bipartite graph representation of binary search tree is each subtree can be retrieved by array indexing.

Planar Point Location in Computational Geometry is the most fundamental generalization of conventional one dimensional list search to arbitrary dimensions. Computational Geometric Factorization algorithm implemented in NeuronRain AstroInfer is a two dimensional parallel geometric search which locates , in polylogarithmic time, factor points with in rectangular faces of planar straight line graph (PSLG) formed by rasterizing (pixel polygon approximation of an algebraic curve on a grid) hyperbolic arc bow on Parallel RAMs - this search involves two phases:

(*) Locating a rectangle in PSLG polygon subdivision containing factor point in polylogarithmic time by some Parallel RAM Planar Point Location algorithm - PSLG has ~(3N+1) vertices and ~4N edges and each polygonal face is a pixel array rectangle of dimensions 1 * N/[x(x-1)]

(*) Binary search the rectangle containing factor point - every rectangle in PSLG of rasterized hyperbolic arc is an arithmetic progression of pixels (ordinate products)

An example hyperbolic arc rasterization - schematic - pixel array polygonal faces
(arithmetic progression rectangles marked f denote faces containing factor points):



Such a Planar Point Location can be extended to arbitrary dimensions > 2 and any other algebraic curve - E.g. algebraic curve uvwxyz=N is a 6-dimensional hyperbola and requires rasterization in 6-dimensional hyperplane to locate factor point (u,v,w,x,y,z). Multidimensional Point Location factorization is the problem of Factorisatio Numerorum (multiplicative partition).

Randomized NC Geometric Search of a 2-dimensional grid and Linear Program Formulation

---

A Grid filling algorithm which maximizes linear program of sum of binary values of ordinate points in Randomized NC has been described in https://sites.google.com/site/kuja27/Analysis%20of%20a%20Randomized%20Space%20Filling%20Algorithm%20and%20its%20Linear%20Program%20Formulation.pdf and its Cellular Automaton, Apollonian Gasket-Circle Packing and Factorization-based Set-Partition-to-Lagrangian-Tile-Cover versions have been drafted in NeuronRain AstroInfer Design - https://github.com/shrinivaasanka/asfer-github-code/blob/master/asfer-docs/AstroInferDesign.txt . Grid Filling as constraint satisfaction problem which maximizes the number of points hit on a 2-dimensional space is a planar point location geometric search if BigData is visualized as values of ordinates within a 2-dimensional rectangle and a value for a query point has to be found by a parallel pseudorandom generator which simultaneously churns out multiple ordinate points (xi,yi) or circles of small radii centred around random ordinate points within the rectangle

(Monte Carlo Simulation). This Randomized NC pseudorandom generator plane sweep covers most of the 2-dimensional rectangle with high probability. Advantage of RNC Grid filling: if query points constitute a meagre percentage of total 2-dimensional rectangular space e.g polynomial curve points, Randomized NC grid filling is able to find query point with high probability in polylogdepth NC without sorting and binary search. It is noteworthy that planar point location for Integer Factorization in exact NC can also be formulated as less stringent Randomized NC Grid filling which locates a factor on hyperbolic curve with high probability. Chaos theoretic RNC parallel pseudorandom generator defined in https://sites.google.com/site/kuja27/ChaoticPRG.pdf could output pseudorandom ordinate bits in parallel.

Probability of hitting a query point within time t is a Bernoulli trial: Probability of hitting a query point within a rectangle = number of query points or length of query polynomial curve embedded in 2-dimensional plane (p) / Area of rectangle of sides a and b (ab) = p/ab

If in time t there are m trials, probability of hitting only query points by pseudorandom sampling = (p/ab)^m, m trials are parallelized by Parallel Chaotic Pseudorandom Bits Generator which generates m 2-dimensional ordinate points simultaneously in Randomized NC.

Previous linear program for maximizing sum of (values are 0 or 1, pseudorandomly flipped) q ordinates x1 + x2 + x3 + ... + xq is exactly the sum of binary random variables Xi: X1 + X2 + ... + Xq. Berry-Esseen Central Limit Theorem - https://en.wikipedia.org/wiki/Berry%E2%80%93Esseen_theorem - implies normalized sum (mean) of random variables tends to a Normal distribution bell curve. Excluding tails of Normal distribution area of bell curve integral(f(x)*dx) from -left_tail to +right_tail is the approximate expected sum of ordinates randomly set to 0 or 1: x1 + x2 + x3 + ... + xq

Probability of hitting a query point after m-th trial (success after m-1 trials):

(1-[p/ab])^(m-1)[p/ab]

In the context of factorization, geometric search reduces to finding factor points on a hyperbolic arc bow embedded in 2-dimensional rectangular space. Though less accurate compared to Exact NC-PRAM-BSP computational geometric planar point location, success amplification of RNC pseudorandom factor point location can be achieved by narrowing down the domain of pseudorandom search to a delta-rectangular strip vicinity in lower halfspace created by leading diagonal of the rectangle:

Length of the leading diagonal = sqrt(a^2 + b^2)

Area of delta-vicinity rectangular strip beneath the leading diagonal which contains the hyperbolic arc = delta * sqrt(a^2 + b^2)

Side of the rectangular strip - delta - is the distance between tangent of hyperbolic arc at (sqrt(N), sqrt(N)) and line connecting (1,N) and (N,1) which is a very small fraction of area of rectangle a*b.

Probability of hitting a query point within the rectangular strip = number of query points or length of query polynomial curve embedded in 2-dimensional plane / Area of rectangular strip = p/[delta * sqrt(a^2 + b^2)] >> p/[ab]

Probability of hitting a query point after m-th trial (success after m-1 trials):
      (1-[p/(delta * sqrt(a^2 + b^2))])^(m-1)[p/(delta * sqrt(a^2 + b^2))]
which amplifies success probability of finding a factor point.

For finding prime factors, number of query points p = kloglogN (Hardy-Ramanujan Theorem) and success probability of finding a factor point is substituted for p:
      (1-[kloglogN/(delta * sqrt(a^2 + b^2))])^(m-1)[kloglogN/(delta * sqrt(a^2 + b^2))]

For a square embedding of hyperbolic arc a=b and success probability amplifies to (delta is configurable):
      (1-[kloglogN/(delta * N * sqrt(2))])^(m-1)[kloglogN/(delta * N * sqrt(2))]

Neglecting delta vicinity, obvious randomization is to pseudorandomly choose an x-axis ordinate x between 1 and N and compute N/x which is a random ordered pair (x,N/x). For Randomized NC, x-axis is divided to N/(logN)^k intervals of width (logN)^k each. Each interval is assigned to a PRAM. Thus N/(logN)^k pseudorandom ordinate ordered pairs are simultaneously generated by a Chaotic Parallel PRG on N/(logN)^k PRAMs which is looped for (logN)^k iterations making it an RNC factorization.

NeuronRain AstroInfer implements a sequential Chaotic PRG based on 1-dimensional binary Cellular Automata which grow chaotically by growth rule fraction lambda, Undecidable Mandelbrot sets, Logistic and Lehmer-Palmore chaotic PRGs. Spread of memes in Social Networks, pandemics, cybercrimes et al all have common Chaos,Game theory and 2-Dimensional Cellular Automata underpinnings (FTrace analyzer in USBmd is a game theoretic botnet defense model and a design of mechanism to counteract - antigame - as well) - For example Verhulst-Pearl-Reed Logistic Law and its Biological variant (lambda*x(1-x) and N0*exp(lambda(1-Nt))) analysis of CoronaVirus2019 pandemic reveals Bifurcation parameter lambda hovering approximately at 4.829487535509905... > 3.57 when chaos sets for a very small initial condition which implies emergence of order in Chaotic spread explained by Period Doubling ratio limit of Bifurcation parameter [lambda(n-1) - lambda(n-2)] / [lambda(n) - lambda(n-1)] - Feigenbaum universality constant 1 = 4.669201609... (https://en.wikipedia.org/wiki/Feigenbaum_constants).

By pandemic cellular automaton model, spread of memes in Social networks can be formulated as RNC space filling problem - social network vertices spreading memes are parallelly chosen random points (people) on a rectangular space effecting a 2-dimensional cellular automaton of 8-neighbours (influenced by meme) per vertex. There are linear stretch PRGs (https://dl.acm.org/doi/10.1007/s00037-007-0237-6 - [Applebaum-Ishai-Kushilevitz]) subject to assumptions in NC0 from which ordinate points can be computed in NC by splitting bit sequence (klogN bit pseudorandom string in NC is split into k (x,y) ordinates of length (logN/2, logN/2) each). As memes automata evenly cover the surface there is an overlap of neighbours adjoining all or some random points. By increment growth rule, overlapped neighbour vertices are incremented implying a high value vertex (point on grid) has high value neighbours and vice versa simulating any parallel random process (this is in discrete contrast to the continuous plane sweep by circles of small radii mentioned earlier). Thus a cellular automaton leads to a random cellular automaton planar graph (CAGraph):

(*) Vertices are randomly chosen and labelled by instantaneous values of points on cellular automaton grid.

(*) New edges are created when a point and its neighbours have non-zero value after some generations of increment growth rule

(*) Decrement growth rule would conversely obviate edges amongst zero points and neighbours thus simulating Susceptible-Infected-Recovered Erdos-Renyi SIR random graph model.

(*) Between every non-zero valued point vertex and its non-zero neighbours (maximum degree = 8)
on grid there is an edge

(*) Clustering traits of previous CAGraph determine diffusion of a concept in community - strongly connected components of high value vertices are the most influenced giants.

Chaotic universality and Ramsey theoretic emergence of monochromatic arithmetic progressions in pseudorandomly colored sequences independently certify "Orderly Disorder" paradox. Recent Chaos Machine PRG by [Maciej A. Czyzewski] has a minimal example implementation of a Push-Hybrid-Pull Chaos Machine PRG computed by non-linear Chaos maps (Butterfly effect, Logistic map, Gingerbreadman map - Chaos Machine source code - https://eprint.iacr.org/2016/468.pdf):

Number of prime factors in [1,N] interval = aloglogN for constant a
Number of prime factors in unit interval = aloglogN/N

Number of prime factors in each of N/(logN)^k intervals of width (logN)^k = [aloglogN/N]*[(logN)^k]

Probability of chaotic pseudorandom choice of a prime factor within each (logN)^k width interval = [aloglogN/N]*(logN)^k/(logN)^k = aloglogN/N = Probability of parallelly locating a prime factor per interval in each Bernoulli trial.

Success amplification - probability of locating a prime factor in each of the (logN)^k Bernoulli trial iterations (Randomized NC) in parallel = (aloglogN/N)^(logN)^k which is exponentially small. Suitable choice of constant a and depth k of RNC circuit (number of trials) bounds the error and thus in BPNC.

Geometric search can be generic to any dataset beyond numeric including strings and texts. Words in text documents can be embedded in an m-dimensional vector space A^m for alphabet A (or language and script independent syllable space A^m if texts are syllable hyphenated) and length of longest word m which is a word embedding kernel for texts. E.g strings "word1" and "word2" in text are embedded as vectors [w,o,r,d,1,...] and [w,o,r,d,2,...] of m dimensions:

Size of the alphabet A (or syllables) = |A|

Length of the longest word in text (or longest syllable hyphenation) = m

Previous kind of alphabet-syllable word embedding of text in a space of A^m clusters similar words by usual numeric euclidean distance measures as opposed to levenshtein edit distance. Traditional Latent Semantic Indexing Singular Value Decomposes a term-document matrix and computes low rank approximation for document similarity. Similarly LSI and SVD could be computed for previous word embedding of text which is an N*m matrix of word_id-alphabets for a text of N words of maximum length m.

NeuronRain AstroInfer has a polynomial encoding implementation of texts which curve fits a text to a polynomial in R^2 [points on polynomial are ordered pairs (alphabet_location,alphabet_unicode)] and PRP error correcting polynomial implementations. Multiple encoded polynomials of texts can be embedded on a 2-D plane and planar point location could be underneath rank() and select() queries on strings. Similarly, People analytics involve intrinsic merit vectors of people e.g [academic credentials, work experience,...] for people clustering.

Finding closest pair of points in a set of points on a vectorspace has quite a few practical applications in sea and air traffic control and vehicle collision avoidance. Replacing the vehicle points by textual strings on a vectorspace (mentioned earlier) is helpful in finding closest pair of strings within a set of strings. Subquadratic closest pair of points detection is a geometric search problem and distance similarities amongst string datapoints could be ranked (by iteratively removing every closest point to a pivot point and recursively recomputing closest pair algorithm). Subquadratic string distance algorithms are important from complexity theoretic standpoint - Edit distance is a standard measure for string similarity (number of additions and deletions of alphabets necessary to morph one string to another) and subquadratic algorithm for edit distance might nullify SETH. If edit distance is expressible in terms of computational geometric algorithm for closest pair of string points, edit distance could be computed in subquadratic time.

Prerequisite for finding Closest pair of string vehicle datapoints by computational geometric divide and conquer earlier is: Number of strings (N) must equal Length of each string(n) (to attain O(nlogn) edit distance subquadratic bound).

Example:
---
string1 - abcde - embedded as point [a,b,c,d,e]
string2 - fghij - embedded as point [f,g,h,i,j]
string3 - klmno - embedded as point [k,l,m,n,o]
string4 - pqrst - embedded as point [p,q,r,s,t]
string5 - uvwxy - embedded as point [u,v,w,x,y]

Each alphabet is encoded as scalar from 1 to 26 and string vectors are embedded on 26^n vectorspace (similarly for ascii or unicode alphanumeric strings, vectorspace is 256^n or 512^n):
string1 - [a,b,c,d,e] - [1,2,3,4,5]
string2 - [f,g,h,i,j] - [6,7,8,9,10]
string3 - [k,l,m,n,o] - [11,12,13,14,15]

Edit distance between string1 and string2 = 5. Euclidean distance between string1 and string2 = sqrt(5^2+5^2+5^2+5^2+5^2) = 5*sqrt(5). In general, euclidean_distance >= function_of(edit_distance)

Earth mover distance and Word mover distance are current state-of-the-art similarity measures for BigData sets. Recently Earth Mover Distance has been approximated in linear time (Relaxed Word Mover Distance - [Kusner] - http://proceedings.mlr.press/v37/kusnerb15.pdf) though exact computation is quadratic or cubic depending on algorithm. If Edit distance is expressible in terms of Earth mover distance, edit distance can be approximated in linear time which is subquadratic (closer to proving SETH is false) and if exact edit distance could be derived from this approximation in subquadratic time SETH is false. Following is an example reduction from Earth mover distance to Edit distance by

tokenizing each string to 1-alphabet arrays or buckets creating a 2D array or histogram per string:

```
abcde - [[a],[b],[c],[d],[e]]
fghijk - [[f],[g],[h],[i],[j],[k]]
```

Earth mover distance or Relaxed Word Mover distance between 2 string histograms earlier is exactly equal to edit distance (work necessary in adding, updating or deleting 1-alphabet buckets for transforming one string to another). NeuronRain AsFer implements Earth Mover Distance similarity measure between two 2D syllabified string tensors.

References:

1.The Art of Computer Programming - Volume 3 - [Donald Knuth] - Sorting and Searching - 6.2.2
2.Beautiful Code - Finding Things - [Tim Bray - Sun Microsystems]
3.Google AI Blog - [Joshua Bloch] - https://ai.googleblog.com/2006/06/extra-extra-read-all-about-it-nearly.html
4.Topological Sorting - Fast Parallel Algorithms - [SA Cook] - https://www.sciencedirect.com/science/article/pii/S0019995885800413 - If set of integers are randomly assigned as labels of a random graph, topological sorting of vertices by some ordering can be done efficiently in NC^2 and a following binary search can find the element in additional logarithmic time. But the topologically sorted vertex list may not be sorted by total ordering always.Dekel-Nassimi-Sahni algorithm works by Repeated Matrix Multiplication logarithmically many times and sorting by longest paths.
5.Planar Separator Theorem for Graphs - [Richard J. Lipton and Robert Endre Tarjan] - https://epubs.siam.org/doi/abs/10.1137/0209046 - PSLG from rasterized hyperbolic arc xy=N can be partitioned to almost equal subgraphs by removing O(sqrt(3N+1)) vertices - quite useful in parallelizing and fair load balancing the rasterized polygon faces amongst PRAMs for point location queries
6.Planar Point Location in sublogarithmic time - [(Late) Mihai Patrascu] and [Timothy Chan] - https://ieeexplore.ieee.org/document/4031368
7.Algorithms - [Cormen-Leiserson-Rivest-Stein] - Page 845 - Number theoretic algorithms - Integer Factorization - 33.9 - Pollard's Rho Heurisitc - "...It is infeasible with today's supercomputers and the best algorithms to date to factor an arbitrary 200-decimal-digit number..."
8.On Implementing an o(log n) Planar Point Location Algorithm - [Yifei Zhang, Wei Yu, Decheng Dai] - https://dsa.cs.tsinghua.edu.cn/~deng/cg/project/2006f/2006f-b.pdf
9.Clarkson-Shor Random Sampling, Partitioning Theorem - http://courses.cs.tau.ac.il/0368-3249-01/michasodaslid.pdf - "...Q a set of n points in the plane R a random sample of r points of Q Then, with high probability, any triangle that does not contain a point of R contains at most cn/r * log r points of Q (c an absolute constant)...For any r < n, P can be partitioned into O (r) subsets P1, . . . , Pt, each of ≤ n/r points, so that any hyperplane h separates the points of only O (r^(1 − 1/d)) subsets..." - Query could be a hyperplane or triangle and not necessarily a point. Random sampling is quite necessary in pre-poll and post-poll forecast analytics - if voters are points on a 2-d plane, Clarkson-Shor estimate upperbounds the extent of voters excluded by a random sampling triangle. For plural multipartisan voting, Partitioning theorem has a direct bearing - Set of Voters P is partitioned to r subsets of candidates voted for and query hyperplane (forecast sample) separates only a fraction of subsets.

10.SIR ODE model of How Gossips and Rumours spread in Social network - https://scholarship.claremont.edu/cgi/viewcontent.cgi?article=1036&context=codee
11.Random Walks and Diffusion in Networks - Brownian Motion - https://www.sciencedirect.com/science/article/pii/S0370157317302946
12.Finding Closest Pair of Points - Subquadratic O(NlogN) divide and conquer algorithm better than naive O(N*N) bruteforce for obstacle avoidance - Section 35.4 - Chapter 35: Computational Geometry - Algorithms - [Cormen-Leiserson-Rivest-Stein] - Page 908 - "... System for controlling air or sea traffic might need to know which are the two closest vehicles in order to detect potential collisions ..."
13.Linear Complexity Relaxed Word Mover Distance - LC-RWMD - [Kubilay Atasu-Thomas Mittelholzer] - IBM Research, Proceedings of the 36th International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019. - http://proceedings.mlr.press/v97/atasu19a/atasu19a.pdf - Figure 2 on computing distance between 2 histograms as transportation cost minimization problem - Table 2 - LC-RWMD has average time complexity O(vhm + nh) and average space complexity O(nh + vm + vh) for n=Number of database histograms, v=Size of the vocabulary, m=Dimensionality of the vectors, h=Average histogram size - Table 5 on comparison of precisions of various distance measures and maximum precision of approximation is > 97% implying EMD-to-EditDistance reduction could be approximated in linear time with > 97% accuracy (or SETH is false with > 97% probability).
14. Implications of ETH being false - https://en.wikipedia.org/wiki/Exponential_time_hypothesis - "... However, if the exponential time hypothesis fails, it would have no implication for the P versus NP problem. There exist NP-complete problems for which the best known running times have the form O(2^n^c) for c < 1, and if the best possible running time for 3-SAT were of this form, then P would be unequal to NP (because 3-SAT is NP-complete and this time bound is not polynomial) but the exponential time hypothesis would be false...."

---
758. (THEORY and FEATURE) 21 January 2019 - FP Growth Algorithm for mining frequent patterns and mining timeout dictionaries example - this section is an extended draft on respective topics in NeuronRain AstroInfer design - https://github.com/shrinivaasanka/asfer-github-code/blob/master/asfer-docs/AstroInferDesign.txt
---
Timeout design pattern implemented as a separate chaining/dictionary of timeouts-to-processes has been described in ../AdvancedComputerScienceAndMachineLearning/AdvancedComputerScienceAndMachineLearning.txt. Discussion there is restricted to a process id present in only one bucket per timeout value. But there are possibilities when same process_id has to be multilocated in many timeout buckets e.g multiple threads spawned by a process can have different timer values for varied functionalities and all those threads per process have to be timedout.

Spark MLlib implements parallel version of FP-Growth algorithm which mines frequent itemsets in multiple baskets with no candidate generation (downward closure) by growing suffix trees. Survival Index Timeout Map is also a set of baskets (buckets) from which frequently occurring process subsets can be mined by Spark FPGrowth. Example Spark code for this has been committed to code/Spark_FPGrowth_SurvivalIndexTimeout.py and logs in code/testlogs/Spark_FPGrowth_SurvivalIndexTimeout.log.21January2019 which demonstrate minimum support for frequent occurrences and minimum

confidence for association rules. These frequently occurring process subsets point to some lurking relationship among the colocated processes in some way and thus a measure of system load and behaviour.

Multilocation of process id(s) in multiple time out buckets creates hyperedges amongst the timer bucket vertices and thereby a hypergraph.

References:

1.Spark MLLib Documentation - https://spark.apache.org/docs/2.3.0/ml-frequent-pattern-mining.html

---
19 March 2019 - String Search in Large Files

BigData or Large Text Files on clouds often require a functionality to quickly search for a string of patterns or text. There are standard string matching algorithms like Knuth-Morris-Pratt, Boyer-Moore etc., which are standard algorithms implemented in text editors for "find". For large filesystems, wavelet trees are fast alternatives which support rank(c,p) [number of occurrences of character c before position p], select(c,q) [q-th occurrence of character c], access(k)[access k-th position] operations in O(1) time for binary vectors. For substring pattern p of size n, repetitive n invocations of select(p[i],q) returns all substrings of pattern p in the large text file - Example Pseudocode below for first match:

```
           for k in xrange(n):
                pos=wavelettree.select(p[k],1)
                if prevpos + 1 != pos
                     return matchfound==false
                prevpos=pos
           return matchfound==true
```

---
26 March 2019 - Example MapReduce in Spark 2.4 Cloud - Bitonic Sequences of Integers
---
Spark framework parallelizes work by partitioning a bigdata set into Resilient Distributed Datasets
which are map()-ped to Spark cloud nodes by Spark Executor Driver and local computations in cloud node are unified by reduce(). Spark 2.4 + Python Spark code example in code/Spark_MapReduce.py defines two functions map() and reduce(). Spark context is instantiated and an array of integers is parallelized to
Spark resilient distributed dataset partitions and map() is invoked per RDD on Map function - Map() which returns a single element array as a tuple. This is followed by reduce() on Reduce function - Reduce() which takes as args two mapped tuples and merges the integer arrays after > or <= checks to create a subarray which is either strictly ascending or descending. Resultant array is checked for sortedness by a function and mapreduces further if not. Logs for this in
Spark_MapReduce.log.26March2019 demonstrate the bitonic sequence n1 + n2 which fluctuate (ascending-descending-ascending):
Reduce: (1, [15], 1, [16], 1, [12], 1, [7], 1, [6], 1, [3], 1, [2], 1, [4], 1, [5])

```
n1: (1, [15], 1, [16], 1, [12], 1, [7], 1, [6], 1, [3], 1, [2], 1, [4],
1, [5])
n2: (1, [23], 1, [32])
Reduce: (1, [15], 1, [16], 1, [12], 1, [7], 1, [6], 1, [3], 1, [2], 1,
[4], 1, [5], 1, [23], 1, [32])
Bitonic Sequence of Integers:
[15, 16, 12, 7, 6, 3, 2, 4, 5, 23, 32]
```

As can be seen, the reduce produces a huge tuple from which an integer array is extracted after typecheck by Python keyword "type".

---

757. (THEORY and FEATURE) 16 March 2019 - Set Partitions in SymPy, Survival Index Timeout OS Scheduler

- this section is an extended draft on respective topics in NeuronRain AstroInfer design - https://github.com/shrinivaasanka/asfer-github-code/blob/master/asfer-docs/AstroInferDesign.txt

---

Spark_FPGrowth_SurvivalIndexTimeout.py example previously described for FPGrowth mining of dictionaries
has been changed to print all possible partitions of set of processes in OS. Partition API in SymPy have been invoked for this. OS Scheduler/Timer is in itself a set partition histogram which maps timeout values to subsets of processes. This demonstrates the integer partition and LSH/set partition-separate chaining isomorphism. Logs in Spark_FPGrowth_SurvivalIndexTimeout.log.16April2019 show the all possible permutations of process set partitions.Rank of a partition is printed which is the difference between size of largest part and number of parts (or) equivalently side of largest square in Durfee Diagram.

References:

1.Durfee Square - https://en.wikipedia.org/wiki/Durfee_square
2.Rank of a partition and Partitions fitting within a rectangle - Gauss Binomial Coefficient - https://en.wikipedia.org/wiki/Rank_of_a_partition
3.Ferrers Diagram or Young Tableau of a Partition - https://en.wikipedia.org/wiki/Partition_(number_theory)#Ferrers_diagram

---

837. (THEORY and FEATURE) People Analytics and Economic merit (Spending Analyzer) - Credit Card Datasets - 9 July 2019 - Fraud Analytics in Spark 2.4.3

NeuronRain AstroInfer implements a primitive fraud analytics on a credit card transactions dataset - https://gitlab.com/shrinivaasanka/asfer-github-code/blob/master/python-src/FraudAnalytics.py which analyzes https://www.kaggle.com/mlg-ulb/creditcardfraud csv dataset. Alternatively Spark provides some basic statistics functions for analyzing DataFrames of BigData. An example Spark Python code on Spark 2.4.3 at code/Spark_FraudAnalytics.py does the following :

(*) Group the transactions by first few columns of cardholder identity and computes average amount withdrawn by aggregating column "Amount"

(*) Find Frequent Items in the DataFrame

(*) Describe the columns and compute basic statistics - mean, standard deviation etc.,

```
...
+-------+--------------------+--------------------+--------------------+-
------------------+--------------------+--------------------+-----------
---------+--------------------+--------------------+------------------+
|summary|                  V1|                  V2|                  V3|
V4|                  V5|                  V6|                  V7|
V8|                  V9|              Amount|
+-------+--------------------+--------------------+--------------------+-
------------------+--------------------+--------------------+-----------
---------+--------------------+--------------------+------------------+
|  count|              284807|              284807|              284807|
284807|              284807|              284807|              284807|
284807|              284807|              284807|
|   mean|3.742631994571313...|4.949726613980831...|-
7.91956258236933...|2.685625859585728...|-
1.52643182031150...|1.813835301123298...|-1.74517780292937...|-
2.00384093565998...|-3.14088095662162...| 88.34961925089794|
| stddev|  1.9586958038574895|  1.6513085794769824|   1.5162550051777769|
1.4158685749409246|  1.3802467340314435|  1.3322710897575698|
1.237093598182658|  1.1943529026692032|
1.0986320892243098|250.12010924018742|
|    min|-0.00012931370800...|-0.00010296722561...|-0.00010859127517...|-
0.00011921826106...|-0.00010366562678...|-0.00010234903761...|-
0.00010533581684...|-0.00010065655617...|-0.00010181309940...|
0|
|    max|7.55406974741191e-05|    9.99769856171626|9.67444968403876e-05|
9.92501936512661|9.99846034625664e-05|    9.91116576052911|
9.97044721041161|    9.90825458583455|9.96754672601408e-05|
999.9|
+-------+--------------------+--------------------+--------------------+-
------------------+--------------------+--------------------+-----------
---------+--------------------+--------------------+------------------+
```

---

756. (THEORY and FEATURE) 21 October 2019 - Advertisement Analytics - Recommender Systems - ALS Collaborative Filtering - this section is an extended draft on respective topics in NeuronRain AstroInfer design - https://github.com/shrinivaasanka/asfer-github-code/blob/master/asfer-docs/AstroInferDesign.txt

---

Collaborative Filtering based Recommender Systems works by plotting a matrix of users versus their item choices and predicting preferences of new users or missing preferences based on inferences from the user-items matrix. Advertisement Analytics by computing PageRank of viewer channel switch graph (converging markov random walk) is explained in https://github.com/shrinivaasanka/Grafit/blob/master/course_material/NeuronRain/AdvancedComputerScienceAndMachineLearning/AdvancedComputerScienceAndMachineLearning.txt. Spark MLLib predefines a function for alternating least squares (ALS) collaborative filtering on cloud. Code example in Spark_RecommenderSystems.py which is a modified version of documentation example in https://spark.apache.org/docs/latest/ml-collaborative-filtering.html adapts to channel recommendation systems based on user surveyed matrix of viewer-channel matrix.This is an alternative advertisment analytics.Channel ratings of viewers are read from text file AdvertisementAnalytics_RecommenderSystemsCF.txt having fields - viewer, channelid, rating and timestamp.

ALS factorizes the user-item matrix into User and Item matrix factors and minimizes a quadratic optimization function:

UserItem = User * Item (Factorization)
(UserItem - User * Item)^2 (Minimization)
ALS makes either User or Item factor constant alternatingly and converts to a quadratic optimization problem.

---
755. (THEORY and FEATURE) 23 October 2019, 1 November 2019 - Sequence Mining of Astronomical Datasets by Spark PrefixSpan - this section is an extended draft on respective topics in NeuronRain AstroInfer design - https://github.com/shrinivaasanka/asfer-github-code/blob/master/asfer-docs/AstroInferDesign.txt
---
NeuronRain Research version of AstroInfer in SourceForge mines astronomical data e.g encoded ephemeris degree locations of celestial bodies, planetary positions on zodiac etc., by native sequential implementation of GSP Sequence Mining algorithm. Spark MLlib provides cloud implementation of an advanced sequence mining algorithm - PrefixSpan which recursively projects a sequence dataset to smaller fine grained datasets and locally mines frequent patterns in smaller databases which are grown (https://ieeexplore.ieee.org/document/1339268 - Mining sequential patterns by pattern-growth: the PrefixSpan approach - Jian Pei,Sch. of Comput. Sci., Simon Fraser Univ., Burnaby, BC, Canada,Jiawei Han,B. Mortazavi-Asl,Jianyong Wang,H. Pinto,Qiming Chen,U. Dayal,Mei-Chun Hsu). An example code for Spark PrefixSpan pattern mining for encoded astronomical datasets (encoding is numeric 1-to-9 for each of the 9 planets spread across 12 zodiac houses delimited by #) is at code/Spark_PrefixSpan.py. It prints frequently occurring celestial juxtapositions of planets in the dataset. Large scale mining of celestial data can be used for variety of scientific applications - solving n-body gravitational differential equations, correlating frequent planetary patterns to terrestrial seismic events and weather forecast. Curiously enough, each encoded string of celestial configuration is a set partition and is an instance of balls-bins problem - set of celestial bodies are bucketed by some permutation amongst 12 houses - number of such celestial configurations are governed by Bell and Stirling numbers which is derived as below (background n-body problem choreography analysis is in NeuronRain FAQ - https://neuronrain-documentation.readthedocs.io/en/latest/):

Number of ordered partitions of length p of 9 celestial bodies = N(p)

Summation_p=1-to-9(N(p)) = B9 - 9th Bell Number (= binomial series summation of Stirling Numbers of second kind) = 7087261

Number of all possible choreographic arrangements of 9 celestial bodies amongst 12 zodiac degree divisions = Summation_p=1-to-m(12CN(p)) which is lowerbounded by B9 >= 7087261 = number of celestial patterns to be correlated to terrestial events. This is a finite number implying repetitive patterns in gravity induced events.

References:

1.Sage OrderedSetPartition - http://doc.sagemath.org/html/en/reference/combinat/sage/combinat/set_partition_ordered.html

---
754. (THEORY and FEATURE) 18 December 2019, 19 December 2019, 22 July 2020, 23 July 2020 - Intrinsic Performance Ratings and Sports Analytics,

Streaming Histogram analytics, Partition distance measures, Random graph streams, Logic and Game theory - this section is an extended draft on respective topics in NeuronRain AstroInfer design - https://github.com/shrinivaasanka/asfer-github-code/blob/master/asfer-docs/AstroInferDesign.txt

---

Non-perceptive intrinsic performance rankings (IPR) have been mentioned as one of the multiple classes of measuring intrinsic fitness/merit as part of People Analytics in https://neuronrain-documentation.readthedocs.io/en/latest/ without voting. IPRs are widely used to rank Chess players (e.g Elo ratings). As an alternative bigdata usecase, stream of cricket match statistics is analyzed for intrinsic merit (Example hawkeye statistics - wagon wheel, trajectories, pitch maps - in https://www.bcci.tv/events/15305/india-v-west-indies-2019/match/15309/1st-odi?tab=overview#hawkeye). NeuronRain Set Partition and Histogram analytics implementations elicit patterns from stream of histogram set partitions by multitude of distance measures - earth mover distance, partition rank, correlation among others:

(*) Batting, Fielding and Bowling figures per match for 11+11=22 players can be plotted as a histogram of player versus runs/wickets.

(*) Streaming Set partition analytics is a special case of Histogram analytics - stream of histograms can have varied total sum of parts while stream of set partitions have oscillating parts but constant total sum of parts.

(*) Stream of match statistics histograms (which always have constant 11 parts per team) as time series has the imprint of team performance over the period of time.

(*) Partition Rank which is defined as maximum part - number of parts is an indicator of skewness in the team performance (maximum runs/wickets scored by a player - 11 per team)

(*) Durfee square which is the side of largest square inscribed in Ferrer diagram of the match statistics histogram measures if the performance is equitable - larger the square, more players performed nearly equally.

(*) Time series of Earth mover distance between consecutive match statistics histograms in the stream sheds light on seasonal trend in team performance.

(*) Patterns in Wagon wheel graphics of shots played per player measures strengths of each player - stream of wagon wheels can be mapped to a histogram of shots and aforementioned metrics apply.

(*) Clustering of Bowling trajectories/pitch maps extract stereotypes.

Most games have been analyzed for their computational complexity - E.g Go, Chess are either in PSPACE or EXPTIME depending on rules. Cricket on the other hand could be theoretically formulated as two True Quantified Boolean Formulae Satisfiability problems in PSPACE - TQBF1 and TQBF2 - forall and existential quantifiers in each formula symbolize team1 and team2 per innings (adversary-defender game). Batting and Bowling Roles (respectively governed by existential and forall quantifiers) for 22 player statistics variables are interchanged in each innings. Maximum satisfied formula decides the winner and game is undecidable if drawn/tied.

Mining soccer patterns is complex because of random trajectories of the ball which draws a random directed graph on field - trajectory is a brownian motion random walk - whose vertices are 2-colored (bichromatic for vertices belonging to two opponent teams). Consecutive match statistics create a stream of 2-colored random graphs and frequent

subgraph patterns in this stream of random trajectory graphs show patterns across matches.

Reference:

1.Real Plus Minus of NBA - http://www.espn.com/nba/statistics/rpm - Player Efficiency Ratings in Basketball
2.Go, PSPACE, Combinatorial Game Theory - https://en.wikipedia.org/wiki/Go_and_mathematics
3.Computational Complexity of Games and Puzzles - https://www.ics.uci.edu/~eppstein/cgt/hard.html

---
753. (THEORY and FEATURE) Social Network Analytics, People Analytics, Preferential Attachment, Urban Sprawls, Graph Edit Distance and Intrinsic Merit Rankings - 20 February 2020,26 February 2020,29 April 2020,17 May 2020,18 May 2020,28 May 2020,15 August 2020 - related to 762, 821 and all sections on urban planning analytics, GIS analytics among other topics in NeuronRain Theory Drafts
---
NeuronRain AstroInfer codebase has implementations for urban planning analytics including analytics of Remote sensing imagery , GIS 2-D patches extraction, Watershed image segmentation specific to urban sprawls et al. Urban planning analytics could be termed a special category lying in intersection of People analytics and Social Network analytics - difference being people profiles and social profile vertices in world wide web are replaced by real humans and urban clusters. There exists a striking parallel between preferential attachment in social networks and formation and growth of urban agglomerations - Processes underneath why certain social profiles attract more incoming links and why people flock to megapolises (and megalopolises formed by countermagnets) have clustering similarities. Preferential attachment in social networks causes profiles of large followers to be larger or famous become more famous while big cities become megacities. It is self-explanatory that perception underlies preferential attachment (because population is swayed by and gravitates to majority). Average degree of a vertex in Social networks is replaced by population density in Urban sprawls.

On the contrary, rationale for ranking social profiles by their intrinsic least energy merit characterized by inverse lognormal sum of education(E), wealth(W) and valour(V) (energy of social profile vertex is directly proportional to 1/(logE + logW + logV)) defined and implemented in NeuronRain AstroInfer applies as well to the ability of an urban sprawl to attract population and ranking urban areas absolutely. Some of the following example standard of living factors could be classified into three divisions of lognormal sum (inverse lognormal sum could be broken-up into following finer contributing factors):

- Harbours, Coastal areas (Wealth)
- Infrastructure (Wealth)
- Manufacturing,IT and ITES jobs (Wealth)
- Education and R&D institutions (Education)
- Financial institutions (Wealth)
- Antiquity, Tourism and Culture (Valour)
- Salubrious and Cool climes (less likely)
- ...

Previous division of Social profile vertices into 3 categories may infact be 4 if Wealth category is segregated into Working class and Business profiles:

(*) Profiles of Academics and Researchers - Education (E)
(*) Profiles of Sports personalities - Valour (V)
(*) Profiles of Business - Wealth1 (W1)
(*) Profiles of Working class - Wealth2 (W2)

which would consummate to lognormal least energy sum logE + logV + logW1 + logW2. On the contrary such a division of social profiles may not be Four color theorem compliant - there is no or least avoidance of colors (categories) of neighbours (though W1 and W2 are repulsive). NeuronRain People Analytics implements Chaotic HMM state machine model for tenure transitions of a profile and an alternative weighted automaton model is put forth (state transitions between organizations are incentivized by weights). Attributing merit to people tenure transitions reduces to rating incentives for choices made e.g:

(*) transition from industry to academics is considered nobler than industry-to-industry.
(*) transition from academics to industry is majority choice
(*) transition from low remuneration to high is a majority choice
(*) transition from high remuneration to low is rare
(*) transition from employment to entrepreneur is rare and considered less risk-averse
(*) transition from low designation to high is majority choice

Weights of the transitions in Weighted Automata Tenure model are decided by ranking of rarity of choices earlier. Comparing state machines (ChaoticHMM or Weighted Automata) of tenures of two social profiles is the problem of Equivalence of Regular languages (verify if two automata are similar) solved by Table filling algorithm. In the context of "original music score" merit measurement, equivalence of weighted automata of two music compositions is verifiable by Table filling.

GIS imagery of any urban sprawl can be segmented by watershed algorithm or contour detection to following categories (or by approximate color of pixels):

(*) Manufacturing,IT and ITES
(*) Residential
(*) Commercial
(*) Agricultural lands, Water bodies (Groundwater and Surfacewater) & Greenery

which correspond to four color cartography of the urban sprawl. By Four color theorem, segments of urban sprawl master plan could be colored by any of the four categories in a way which prevents any neighbor of a segment to be of same category thereby ensuring equitable and sustainable development. Fair Division of resources within an urban sprawl (water,amenities,...) is a Pareto optimal Multiple Agent Resource Allocation (MARA) problem. An allocation is Pareto optimal if no agent is preferentially treated at the expense of its peers (Envy-free) while a Nash equilibrium occurs when every agent is at its best and can't do better. Four categories or colors of an urban sprawl GIS segmentation earlier correspond to Four agents among whom a resource has to be fairly divided - an analytics alternative to Four color theorem. Segments of an urban sprawl GIS imagery form a Dual graph induced by segment vertices (faces) and their adjacency. Four coloring theorem stems from 4-colorability of unavoidable sets and 1482 reducible configurations. On the lines of TCP congestion control in electronic networks, Bottleneck measures (e.g Expanders, Cheeger constant) of 4-colored FaceGraph of a segmented urban sprawl GIS imagery help in theoretically solving the traffic congestion problem in urban sprawls - Most transport in urban sprawls happens amongst 3 face categories of FaceGraph - Manufacturing-

IT-ITES,Residential and Commercial. Four colored Facegraph of segmented urban sprawl GIS imagery has an inherent feature which decongests traffic snarls - every s-t path between vertices of Facegraph is 4-colored (two consecutive vertices in a path avoid same color) implying every route in transport network traverses perfect mix of 4 face categories earlier. Graph Edit Distance between FaceGraphs of 2 urban sprawls is a theoretical measure of similarity which quantifies Urbanization and thus could be used in Ranking Urban agglomerations. FaceGraph as a Flow network (Maxflow-Mincut problem) of face vertices maximizes traffic between source and sink faces through minimum cut edges which if removed would disconnect source from sink. Apart from these population dynamics of urban sprawls are of statistical importance and are necessary for redrawing and expanding urban boundaries. Ricker non-linear logistic N(t+1) = N(t)e^(r(1-N(t)/K) formalizes growth of population from generation t to (t+1) from which future population projections for an urban sprawl could be inferred (implemented as R script in GRAFIT course material). Increasing population implies increasing population density (average degree) for constant sprawl.

Extent of correlation between Rankings of Urban Sprawls by Population (mostly caused by preferential attachment - Urban Area Rankings in Section 675 of https://github.com/shrinivaasanka/asfer-github-code/blob/master/asfer-docs/AstroInferDesign.txt) and Intrinsic Lognormal Sum Least Energy Merit is a measure of Fame-Merit coincidence.

References:

1.Preferential attachment,Bose-Einstein Condensation in Networks,Double Jeopardy in business intelligence (low market share brands have low buyers and low brand loyalty - low market share brand is not necessarily of low merit) - https://en.wikipedia.org/wiki/Preferential_attachment
2.Automata - [Hopcroft-Motwani-Ullman] - Page 157 - 4.4.2 - Table Filling algorithm for Equivalence of two automata (NFA and DFA) - If two states are not distinguished by Table filling, they are equivalent
3.NASA SEDAC GIS Population Estimator - https://sedac.ciesin.columbia.edu/mapping/popest/gpw-v4/ - PDF of 6-colored Terrain Population Density of Settlement Points in Southern India (2015) and Convex Hull Population estimator - Gridded Population of the World (GPW) Version 4 - has been committed to code/testlogs/NASA_SEDAC_Population_Estimator.6coloredGIS.16August2020.pdf

---
821. (THEORY and FEATURE) Word Embedding on a Vectorspace - Spark SkipGram Log Likelihood - Text Similarity, Dissimilarity and Merit Measure of Originality - 10 May 2020,11 May 2020 - related to 762,815 and all sections on Kernel Lifting, Intrinsic Merit of Academic publications/Large Scale Visuals/Music/People, Psychoanalysis and Shell Turing Machines
---
Word2Vec Embedding in Spark MLLib maps words in a text to vectors in a Vectorspace which facilitates
clustering of synonymous words - Words of similar meaning and context are in proximity on the vector space. This kind of kernel lifting from one dimensional text to multidimensional vectorspace of words in Spark MLLib is implemented as a SkipGram. SkipGram maximizes average loglikelihood for a word wt and a context training window for wt of width 2k - w(t-k),...,w(t+k):

1/T ∑t=1,T ∑j=-k,k logp(w(t+j)|wt)

Code example Spark_Word2Vec.py defines a function bibliometrics_word2vec() which analyzes text file bibliographies from Semantic Scholar(https://www.semanticscholar.org/author/Ka.-Shrinivaasan/1861803), DBLP(https://dblp.dagstuhl.de/pers/hd/s/Shrinivaasan:Ka=) and GoogleScholar (https://scholar.google.co.in/citations?user=eLZY7CIAAAAJ&hl=en) profiles of author on quadcore and Python 3.7.5 and tries to find synonyms and clustering similarities (by cosine distance). Sections 762 and 815 propound an alternative merit measure of "Originality" (how dissimilar a work is from other authorships and common theme similarity amongst author's works) which can be best captured by Word2Vec vectorspace embeddings of texts wherein computation of euclidean distance clustering similarity between word vectors is easier. Analyzing academic publications is a field in itself - Bibliometrics and Scientometrics - once touted to be an alternative to peer review. It is worth noting that TextGraph representation of texts offers Graph Isomorphism and Graph Edit Distance as graph theoretic concept similarity alternatives. Originality thus distinguishes seminal papers (differentiated by huge conceptual distance from earlier works), independent of H-index (Fame measure based on voting - citations), which ushered a revolution when they were published. H-index is the Durfee Square of the Integer Partition of Citations amongst papers and approximately equals 0.54*sqrt(N) for total citations N. Concept similarity is defined by archetypes (Thought motif from which copies are made) in Carl Jungian Psychoanalysis (Pattern of Thought in collective unconscious).

References:

821.1 H-index - https://en.wikipedia.org/wiki/H-index
821.2 Bibliometrics for Internet Media: Applying H-index to YouTube - https://arxiv.org/abs/1303.0766
821.3 Spark MLLib Word2Vec - https://spark.apache.org/docs/latest/api/python/pyspark.ml.html?highlight=word2vec

---
822. (FEATURE) Word Embedding of Text on Vectorspace - Merit of Academic Publications - an alternative to H-index - implementation changes - 11 May 2020 - related to 821

1.Spark_Word2Vec.py has been refactored to include separate mapreduce functions for tokenizing the text
to words after utf-8 encoding.
2.To circumvent Spark Java String to Iterable cast error in Word2Vec fit(), list comprehension in mapreduce creates one element arrays per word (instead of array per line earlier).
3.Parsed data is printed by DataFrame show() and printSchema() - StringType Spark type has been imported.
4.GoogleScholar bibliography has been included.
5.Spark logs are at testlogs/Spark_Word2Vec.log.11May2020

---
826. (FEATURE) Spark SkipGram Word2Vec Bibliometrics - increased vocabulary and accuracy - 9,11 June 2020
- related to 821,822

1.Spark_Word2Vec.py has been changed to parse an expanded Bibliography text file (Spark_Word2Vec_Bibliography.txt) which is a concatenation of BibTeX from various Bibliography sources (Google Scholar,CiteSeerX,Microsoft Academic,Semantic Scholar,Harvard NASA ADS,DBLP) instead of parsing a single source.
2.Because of enlarged BibTeX training data, vocabulary of Word2Vec bibliometrics has improved and more meaningful words e.g author,year,Science,merit,title,document have been vectorized
3.All BibTeX are from author search of self - "Ka Shrinivaasan" - from URLs:
Google Scholar - https://scholar.google.co.in/citations?user=eLZY7CIAAAAJ&hl=en
DBLP - http://dblp.dagstuhl.de/pers/hd/s/Shrinivaasan:Ka=
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
Semantic Scholar - https://www.semanticscholar.org/author/Ka.-Shrinivaasan/1861803
CiteSeerX - https://citeseerx.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
4.Vector size and Seed size have been set to 5 and 42 respectively
5.Vocabulary could be further improved by word2vec embedding of entire article instead of BibTex abstracts which is a concept cloud of vectors.

---
827. (THEORY and FEATURE) Intrinsic Merit of Academic Publications - Bibliometrics - First Order Logic, Monoidal Categories, Girard Geometry of Interactions (GoI), Sequent Calculus, ProofNets - related to 624,821,822,826 - 10,11,13,14,15,16,17 June 2020

There exists a fine division in quantifying merit of a natural language text and academic publications cutting across disciplines - theoretical and experimental. Natural language texts are measurable by mental representation of meaning and its lambda function approximation (Recursive Lambda Function Growth) whereas every academic publication could be construed as sequence of first order logic formulae connected by logical implications - from conjecture to QED. In Sequent Calculus proof has a tree structure and every line of a proof is a conditional tautology on previous lines. Girard Geometry of Interactions maps every theorem-proof to geometry of logical statements (ProofNets) by defining trip operator (Long Trip Criterion) on proof network. For instance, Coq Theorem Prover has been used to verify proof of Four Color Theorem by [Appel-Haken]. Thus Concept distance for Originality merit measure between any two academic publications could be defined as the distance between their ProofNet GoIs (e.g graph edit distance between ProofNets). On a related note, Distance between two First order logic (FOL) statements is the problem of equivalence of 2 FOL formulae F1 and F2: If F1 == F2, distance is 0 else distance is the number of mismatches in truth table - To prove F1 == F2, both F1 => F2 and F2 => F1 have to be established. There are two other prominent proof calculi - Hilbert and Natural Deduction. Analyzing academic publications and theorem-lemma-proofs therein thus has two facets: 1) Natural Language Processing - Lambda Function approximation, Word2Vec embedding of concepts and their clustering 2) Formal Logic - Proof theory - Sequents, Gentzen Cut-

Elimination (A |- B,C and D,B |- E then A,D |- C,E), ProofNet-GoIs . There is an alternative Category theoretic formalism of Proofs wherein every proof entailment A |- B is a morphism A -> B between objects A,B in a monoidal category M having a binary operator * - entailment morphisms thus constitute a ProofNet directed graph (a quiver in Category jargon - https://ncatlab.org/nlab/show/quiver). Geometry of Interaction in Linear Logic and Type theory considers A |- B as an endomorphism.

References:

827.1 Coq Proof Assistant and Four Color Theorem - https://www.ams.org/notices/200811/tx081101382p.pdf
827.2 Proof Nets - Sequent Calculus and Girard Geometry of Interactions - https://en.wikipedia.org/wiki/Proof_net
827.3 E Theorem prover - https://wwwlehre.dhbw-stuttgart.de/~sschulz/E/E.html
827.4 Concept Cloud Wordle example for publications - https://ui.adsabs.harvard.edu/search/q=author%3A%22Shrinivaasan%2C%20Ka.%22&sort=date%20desc%2C%20bibcode%20desc/concept-cloud - prominent concepts in a set of publications are highlighted
827.5 Equivalence of FOL statements - Biconditional - http://iiitdm.ac.in/old/Faculty_Teaching/Sadagopan/pdf/Discrete/Logic.pdf - Indian Institute of Information Technology Design and Manufacturing, Kancheepuram, Chennai 600 127, India
827.6 n-Category Lab - Geometry of Interaction - https://ncatlab.org/nlab/show/Geometry+of+Interaction

---
829. (THEORY and FEATURE) Intrinsic Merit of Academic Publications - Bibliometrics - SkipGram Word2Vec of publication full text - Concept Cloud Wordle - TOP Categories, Shell Turing Machines and Kernel Lifting,Lambda functions and Beta Reduction, Meaning representation, Graph Edit Distance - related to 827 - 13,14,15,16,17 June 2020,13 July 2020, 6 December 2020

1.This commit revamps Spark_Word2Vec.py implementation to parse an entire article text to extract a concept cloud wordle from it by Spark SkipGram Word2Vec. Word2Vec is computationally intensive and corpora-hungry for producing accurate embedding.
2.New text file Spark_Word2Vec_PublicationFullText.txt is read which contains a draft publication of author on "Majority Voting" - https://5d99cf42-a-62cb3a1a-s-sites.googlegroups.com/site/kuja27/IndepthAnalysisOfVariantOfMajorityVotingwithZFAOC_2014.pdf
3.Concepts and respective vectors for them are pretty-printed. Concept cloud of full text is quite meaningful and captures the essential words in the text. Clustering of related concepts is implied by Synonyms printed.
4.Recursive Gloss Overlap and Recursive Lambda Function Growth algorithm implementations in NeuronRain could benefit from Word2Vec in filtering less important words while computing textgraphs, euclidean distance among them and their dense subgraph k-cores which serve as unsupervised text classifiers.
5.Extracting a composition of Lambda Functions from Natural Language Text of a Publication and First order logic (FOL) ProofNet representation of Theorems therein are empirically equivalent - former is a Turing machine while the latter is its FOL version. Example Beta Reduction ((lambda.x

t)s) in reference 829.1 translates a natural language sentence to lambda function:

Whiskers disdained catnip => disdained(Whiskers,catnip)

which could be written as FOL statement:

There exist x There exist y (x disdained y)

and model {Whiskers,catnip} satisfies it. Recursive Lambda Function Growth algorithm works by Beta Reduction and is motivated by Grounded Cognition in Cognitive Psychology. Thus ProofNet or Category morphisms representation of texts is not just limited to academic publications but pervades any natural language text.

Meanings of Natural Language Texts captured by Recursive Gloss Overlap and Recursive Lambda Function Growth TextGraph representation algorithms could be compared by Graph Edit Distance algorithms which is NP-Hard and APX-Hard (hard to approximate) though there are polynomial approximations for Graph Edit Distance. Graph Edit Distance is the problem of Inexact Graph Matching for dissimilar graphs (while Graph Isomorphism is Exact Graph Matching for similar graphs) solved by computing optimal edit paths through A*-algorithm. Graph Edit Distance generalizes String edit distance - every string is a connected directed acyclic graph of maximum indegree 1 and outdegree 1.

6.Vectorspace embedding of concepts in an article basically does a kernel lifting of the model to arbitrary dimensional Topological space and from equivalence of Lambda functions-First order logic mentioned earlier, every first order logic statement in the ProofNet is indirectly kernel lifted and implication entailments between them are replaced by monoidal category morphisms between one logical statement embedded in space S1 to another embedded in S2 (A(S1) |- B(S2) is a morphism A(S1) -> B(S2) in a theorem monoidal category). Word2Vec typically embeds all concepts in single space and thus S1=S2.

7.Thus every theorem-proof could be stated as TOP category of first order logic objects in some topological space and morphisms amongst them.

8.Spark_Word2Vec.py has been made Python 3.7 compatible by autopep8 and 2to3.

References:

829.1 Meaning Representation by First order logic and Lambda calculus - https://www.cs.mcgill.ca/~jcheung/teaching/fall-2015/comp599/lectures/lecture14.pdf

829.2 Logical Representations of Sentence Meaning - Chapter 16 - Speech and Language Processing - [Jurafsky-Martin] - https://web.stanford.edu/~jurafsky/slp3/edbook_oct162019.pdf - Model theory, Currying and Beta-reduction in Lambda calculus,Tense and Temporal Logics - Modus Ponens on FOL sentences and Lambda function meaning and event representation - Examples: " ... VegetarianRestaurant(AyCaramba) ⇒ Serves(AyCaramba,VegetarianFood) ... ∃ e Eating(e) ∧ Eater(e,Speaker)∧ Eaten(e,TurkeySandwich) ∧ Meal(e,Lunch) ∧ Location(e,Desk)..."

829.3 Graph Edit Distance - https://en.wikipedia.org/wiki/Graph_edit_distance

829.4 Graph Edit Distance - Basics and Trends - https://www.slideshare.net/LucBrun2/graph-edit-distance-basics-trends

829.5 Comparing Stars: On Approximating Graph Edit Distance - Polynomial time approximation of Graph Edit Distance - https://web.archive.org/web/20170810170852/http://www.vldb.org/pvldb/2/vldb09-568.pdf

---

836. (THEORY and FEATURE) People Analytics - Theoretical (Drone) Electronic Voting Machines - Population Count - Number of 1s (Decimal parity) in a bitstream - Various algorithms - Parallel computation in Spark - 3 July 2020

Finding parity of a boolean string is a major open problem in Computational Complexity which is not known to be in AC0 (constant depth) but in NC1 (log depth parallel computing).
BigData version of parity is the problem of Population Count which pertains to finding number of 1s in a stream of bits. Lot of algorithms for population count have been invented in Hardware and Artificial Intelligence literature - Divide and Conquer, HAKMEM, CSA(Carry Save Adder) etc., . Code example Spark_PopulationCount.py reads a CSV file containing a huge binary string and computes the number of 1s in it by sequential and parallel mapreduce. String is split into words of length 32 and parallelized to RDDs. Each word is then scanned by a mapbitstream() and population_count() functions which sequentially compute the function:

pop(x) = -1 * Summation((x << n)) for each bit position n.

Bit left shift has been replaced by access to literal in bitstream string. Reduce function reducebitstream() gathers and sums each 32-bit word. Population count gains importance in the context of efficiently counting number of votes in a theoretical Electronic Voting Machine.

References:

1.The Quest for an Accelerated Population Count - Beautiful Code - [Oram-Wilson] - Chapter 10 - Population Count by Divide and Conquer, HAKMEM, CSA
2.HAKMEM - [Beeler-Gosper-Schroeppel] - MIT Artificial Intelligence Laboratory AIM 239, February 1972 - https://w3.pppl.gov/~hammett/work/2009/AIM-239-ocr.pdf
3.Computer Architecture: A Quantitative approach - [Hennessy - Patterson] - Full Adder of 3 1-bit inputs and 2 1-bit outputs of sum and carry - CSA is sequence of full adders

---
839. (THEORY and FEATURE) Bibliometrics - Patents as ProofNets and Search - Novelty detection and Originality merit measure, Latent Semantic Analysis, Low Rank Approximation - 6,7,8 July 2020 - related to 829

Patents are uniquely granted to individuals and corporations who have invented an original concept which could be underneath any STEM artifact. Patent Search is a time consuming process and could take years to validate a patent application and involves human effort to find Concept similarity (Novelty detection,Prior art). Novelty detection and Originality are thus synonymous - Latent Semantic Analysis by SVD (X = U(Eigen)Z^T) and low rank approximation (choose creamy layer of UEZ^T) of term-document matrix of patents mines concept words related to each other. Alternatively, First order logic ProofNet format of patent applications written in natural languages could be an apt model to quantify conceptual distance and patent search. Patent database is a massive open online dataset (MOOD) and Concept similarity search could be automated by a ProofNet distance computations amongst pairs of patent GoIs on a cloud thus replacing humans. This commit computes a Concept Cloud wordle of set of example team software patents author was involved as former staff (Sun Microsystems-Oracle)

References:

839.1 Sun Microsystems-Oracle iPlanet Application Server (iAS) 6.5 patents - http://patft.uspto.gov/netacgi/nph-Parser?Sect1=PTO2&Sect2=HITOFF&p=1&u=%2Fnetahtml%2FPTO%2Fsearch-adv.htm&r=0&f=S&l=50&d=PTXT&Query=%22kannan+srinivasan%22+AND+%22sun+microsystems%22
839.2 Patent Novelty Detection - https://www.sciencedirect.com/science/article/abs/pii/S147403461830421X - Latent Semantic Analysis by Low Rank Approximation of Term-Document matrix

----
840. (THEORY and FEATURE) Fraud Analytics - Cybercrime analyzers - related to 752, 770 (USBWWAN wireless traffic and Ftrace analyzers) - 6,7,8 July 2020

Cybercrimes are a scourge on society causing immense economic damage. Distinguishing a spoofed email from genuine origin is an artificial intelligence problem and is still less solved. Spam filters and Firewalls in Mail service providers have machine learning algorithm implementations which segregate botnet senders from humans to some extent (Spam folders in mailboxes use Bayesian filters to differentiate frequency of words in human and robot senders). A commonsense workaround to spoofs: As against keyboard typed emails, text could be a handwritten image (scanned or touchscreen stylus written) backedup by a private hardcopy which is digitally watermarked and sent through usual HTTPS/IMAP/POP/SMTP infrastructure. Written emails are less spoof-prone and less forgeable (because of private hardcopy duplication) and introduce an additional layer of reCAPTCHA Turing tests to discern a robot from human. Old school Post office mails are written and stamped hardcopies. Most cybercrimes are difficult to probe not just because of their international impact but as well due to complex traffic routing (e.g Onion routers) which obfuscate IP, munge headers and fudge dates. Parallels between pandemics and cybercrime transmission are fascinating:

(*) Pandemics are air,vector,contact borne while Cybercrimes are ISP-borne, proliferate by address book lookups (e.g Worm spreads by infecting an id and recursively traversing contacts - contacts per id could be a huge number and thus infection graph could be an expander of high degree)

(*) Both pandemics and cybercrimes follow Game theoretic (Botnet adversary-defender game), SIR and SIS models of infections

Predominant attacks which render RSA dysfunctional are Radio-Frequency Van Eck Phreaking (capturing RF signals from monitors),Keystroke loggers (Trojan/Malware capturing keystroke signals), SQL and JavaScript XSS Injection attacks which execute malicious code in client side et al. Some example past cybercrime incidents of author's credentials (compromise of linkedin id by password reset, spoofs of yahoo webmail and past institutional affiliation ids) and their traffic pattern (headers) have been committed to code/testlogs/Cybercrime_examples/

References:

840.1 Van Eck Phreaking of Electronic Voting Machines - https://yro.slashdot.org/story/09/11/22/027229/brazilian-breaks-secrecy-of-brazils-e-voting-machines-with-van-eck-phreaking

---
841. (THEORY and FEATURE) Digital Watermarking of Large Scale Visuals in OpenCV - related to 581,840 and all sections on Large Scale Visual Recognition Analytics, Handwriting Recognition and Fraud analytics - 10,11,12 July 2020

This commit implements an OpenCV python function which reads a large scale visual (Video) and a watermark
image and writes out a new MP4 video overlaying the watermark image on each frame of the visual. New
MP4 video captures the OpenHub,SourceForge,GitHub and GitLab NeuronRain repository profiles of Krishna iResearch Free Open Source Software initiative of the author having both images and text (testlogs/Krishna_iResearch_NeuronRain_Repositories-2020-07-10_13.17.20.mp4). Function watermark() in DigitalWatermarking.py reads each frame of the video and watermark image in BGRA 4-channel mode (Blue-Green-Red-Alpha). Alpha channel controls the opacity of each pixel.An overlay ndarray is created of the same dimension as each frame of the video. Pixels of overlay array are filled by watermark image. OpenCV addWeighted() function computes a weighted sum of each pixel of watermark and frame images in BGRA 4-color channel - opacity of watermark is controlled by weights. This example watermarks first 2 frames of the video by a Sanskrit name text image (in testlogs/). Alternatively, Image text could be a handwritten signature. Previous watermark is quite primitive and overlay array could be filled by an arbitrary complex pixel cryptographic hash checksum function prior to addWeighted() invocation. Such a steganographic information hiding finds applications in Copyright Protection, Licensing violations, Software Piracy, Currency watermarking, Fraud and Tamper protection, Antitheft Cybercrime software inter alia.

References:

841.1 Digital Watermarking - overview - https://www.sciencedirect.com/topics/computer-science/digital-watermark - Movie Piracy example - every frame of DVD is copyright watermarked and can be tracked. On the similar lines watermarked open source software repositories should be able to track each fork-off and flag OSS violations if necessary.

---
848. (THEORY and FEATURE) 1 August 2020 - Number crunching - Sublogarithmic Prime power encoding of huge integers by Computational Geometric Factorization and 2060 bits integer factorization quadcore benchmark - related to 2 and all sections on Text Compression and Compressed Sensing, Factorization and its applications, Multiplicative Integer Partitions of NeuronRain Theory Drafts
---
It is often necessary to store numeric bigdata succinctly - Numeric compression vis-a-vis Text Compression. Usual binary storage requires logN bits. From Unique Factorization Theorem every integer can be expressed uniquely as product of powers of consecutive primes. Number of prime factors of an integer N is kloglogN by Hardy-Ramanujan theorem. Thus huge integers could be stored as concatenated strings of exponents of consecutive primes (primes are implicit and prior factorization is assumed). Identifying the powers of prime factors is the multiplicative partition problem and sum of prime powers in unique factorization is

defined by Big Omega. Thus storing the powers of prime factors (partition of Big Omega - all possible huge integers could be generated by partition function of Big Omega) and prime factors could be efficient (sublogarithmic space) compared to storing a huge integer N by logN bits if following holds:

Average length (bits) of each prime exponent in Big Omega of multiplicative partition = l1

Average length (bits) of each prime in multiplicative partition = l2

Total number of bits required to store huge integer = (l1 + l2) * kloglogN

if (l1 +l2) * kloglogN < logN, prime exponent encoding of integers is sublogarithmic:

=> (l1 + l2) < logN / kloglogN

=> For integers of low prime factors (Little Omega) prime exponent encoding is quite feasible because of high upperbound for l1 + l2

Code example code/Spark_PrimePowersEncoding.py invokes the Computational geometric PRAM-NC-BSP factorization for 2060 bits integer
(993335395099872743147480777777777777777777777777777777777777777777777777
7777777777777777777777777777777777777777777777777777777777777777777777777
7777777777777777777777777777777777777777777777777777777777777777777777777
7777777777777777777777777777777777777777777777777777777777777777777777777
7777777777777777777777777777777777777777777777777777777777777777777777777
7777777777777777777777777777777777777777777777777777777777777777777777777
7777777777777777777777777777777777777777777777777777777777777777777777777
7777777777777777777777777777777777777777777777777777777777777777777777777
7777777777761444238267790503463029217) implemented in NeuronRain by subprocess Python feature on quadcore + Python 3.7.5 and factors are persisted to a JSON list as under (only few small and large factors have been found including non-prime factors). This is the largest integer factored by far in NeuronRain - factors are printed from 22 seconds onwards and depth of the non-uniform logdepth circuit is 186:
[103,
9644032962134686826674570658036677454153182308522114347357065803667745415
3182308522114347357065803667745415318230852211434735706580366774541531823
0852211434735706580366774541531823085221143473570658036677454153182308522
1143473570658036677454153182308522114347357065803667745415318230852211434
7357065803667745415318230852211434735706580366774541531823085221143473570
6580366774541531823085221143473570658036677454153182308522114347357065803
6677454153182308522114347357065803667745415318230852211434735706580366774
5415318230852211434735706580366774541531823085221143473570658036677454153
1823085220984895517163014596728439, 673,
1475981270579305710471739640085851081393429090308733696549446920917946177
9758956579164602938748555390457322106653458807990754498926861482582136371
1408287931319134885256727753013042760442463265643057619283473666831764900
1155687634142314677232953607396400858510813934290903087336965494469209179
4617797589565791646029387485553904573221066534588079907544989268614825821
3637114082879313191348852567277530130427604424632656430576192834736668317
6490011556876341423146772329536073964008585108139342909030873369654944692
0917946177975895657916460293874855539045732210665345880799075449892686148
2582136371116559046460312783748929]

Theoretically, by equivalence of Parallel RAM and Nick's class (depth of NC circuit = polylog PRAM time, size of NC circuit = number of PRAMs), previous integer requires O(N/(logN)^186) = constant*2^(2060)/ (2060)^186 = constant*5527 PRAMS

References:

848.1 Prime Omega Functions - Little Omega and Big Omega - https://en.wikipedia.org/wiki/Prime_omega_function - Little Omega in terms of partition function
848.2 Prime Big Omega - http://oeis.org/wiki/Omega(n),_number_of_prime_factors_of_n_(with_multiplicity)

----

859. (THEORY and FEATURE) Numeric Compression by Unique Integer Factorization - updated - Spark
NC-PRAM-BSP 2063 bits integer factorization - Euclidean Semiprimes - 11 August 2020, 16 August 2020

Spark_PrimePowersEncoding.py has been updated to parse the JSON factors file written by DiscreteHyperbolicFactorizationUpperbound_TileSearch_Optimized.py after renaming ./DiscreteHyperbolicFactorizationUpperbound_TileSearch_Optimized.factors to Spark_PrimePowersEncoding.factors in subprocess.call("mv"). Factors are parsed from JSON and their bits length are printed. Small semiprime integer 6541 of 13 bits has been factorized and the length of factors 211 and 31 are printed (code/testlogs/Spark_PrimePowersEncoding.log2.11August2020). Euclid's algorithm has been applied to create semiprime integer 6541 - by multiplying consecutive primes and incrementing by 1:

```
      (2*3*5 + 1) * (2*3*5*7 + 1) = 31 * 211 = 6541
```

In previous example sum of lengths of prime factors (12.675295499) is slightly exceeded by length of integer (12.67529549909406):

```
Length of integer -  6541  - factorized (bits): 12.67529549909406
=======================================
Length of factor -  1  (bits): 0.0
Length of factor -  211  (bits): 7.721099188707186
Length of factor -  6541  (bits): 12.67529549909406
Length of factor -  31  (bits): 4.954196310386876
```

Similarly huge semiprimes could be found and the encoding is independent of radix (Numeric compression is valid for any log N to base M M-ary representation for M >= 2 because prime factors and their exponents can be encoded in base M - compression ratio remains same).

Huge 2063 bits integer (arbitrary) -
99999999999999999999999999999999999999999999999999999999999999999999999999
99999999999999999999999999999999999999999999999999999999999999999999999999
99999999999999999999999999999999999999999999999999999999999999999999999999
99999999999999999999999999999999999999999999999999999999999999999999999999
99999999999999999999999999999999999999999999999999999999999999999999999999
99999999999999999999999999999999999999999999999999999999999999999999999999
99999999999999999999999999999993928382389289382989928382983982983928983928
93892893892898829389283989223982983989999999999999999999999999999999999999
92988283888238223232323299999459999 - has been factorized (largest integer factorized thus far in NeuronRain) finding compression scheme for which is time consuming because of huge multiplicity, complexity of isolating prime factors and finding their exponents - excerpts from code/testlogs/Spark_PrimePowersEncoding.log.2063bits.11August2020 print two factors in 14 seconds:

('Factors are: (', 29, ',',
344827586206896551724137931034482758620689655172413793103448275862068965517241379310344827586206896551724137931034482758620689655172413793103448275862068965517241379310344827586206896551724137931034482758620689655172413793103448275862068965517241379310344827586206896551724137931034482758620689655172413793103448275862068965517241379310344827586206896551724137931034482758620689655172413793103448275862068965517241379310344827586206896551724137931034482758620689655172204426978941013206549254585654585652723583756514790997551320479063444559994132410344827586206896551724137931034482758378906340973731835597352862068779 31, ') (at ', 'Thu, 02 Jul 2020 08:19:01 GMT', ')')

Following are few examples of Sublogarithmic space Numeric Compression by Unique Factorization:

---

*) 1024 = 2^10 in binary has 10 bits. But by encoding 2^10 in unique prime factor basis - set of ordered pairs of the form (prime_factor,exponent) - as (2,10) which requires 2 + 4 = 6 bits only.
*) 60466196 = 2^10*3^10 = 6^10 is 26 bits integer while unique prime factor encoding (2,10)(3,10) requires only 2 + 4 + 2 + 4 = 12 bits (space for ordered pair parenthesization is an additional kloglogN which has been neglected). If each parenthesis is of 3 bits for "(","," and ")" delimiters, total space increases to 12 + 3 + 3 = 18 bits in unique factor scheme but still less than 26.
*) CPU instruction set microcodes might have to compute exponentiation and products of prime exponents which increases per instruction time in spite of compressed space.