UNIVERSITY
OF
PADUA

DEPARTMENT
OF INFORMATION
ENGINEERING

USTAR2: Fast and Succinct Representation of k-mer Sets Using De Bruijn Graphs

Enrico Rossignolo Matteo Comin

BIOINFORMATICS2024

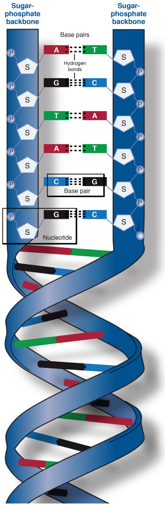
Outline

- Introduction
  - DNA and k-mers
  - Applications
  - Problems
- Methods
  - k-mers set compression
  - Vertex-disjoint path cover problem
  - Finding the minimum CL
  - USTAR2 strategy
- Experimental setup
- Results
  - CL
  - Compression
  - Time and memory
- Conclusions

DNA and k-mers

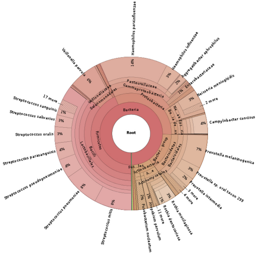
- DNA sequence: string with alphabet A,C,T,G
- k-mers: DNA **substrings of length *k***

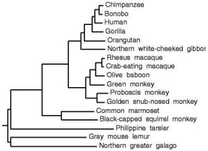
  -
    ```
    ACTTAGC
    ACT
     CTT
      TTA
       TAG
        AGC
    ```
- counters: k-mers multiplicity
  - small values: read error
  - big values: repeat



k-mer based applications

- Assembly: Eulerian walks in graph of *k*-mers that can be found **efficiently** (Spades)
- Phylogenetics: Mash[1] creates trees using *k*-mers
- Database searching (BIGSI)
- Metagenomics: Taxonomy labelling (Kraken[2])
    - 900 times faster than tools based on alignment

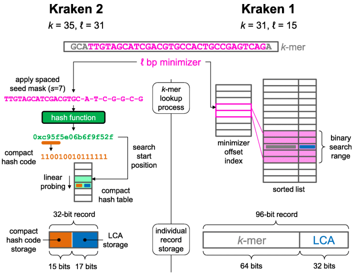




[1] *Ondov, B.D., Treangen, T.J., Melsted, P. et al. Mash: fast genome and metagenome distance estimation using MinHash. Genome Biol 17, 132 (2016)*
[2] *Wood, D.E., Salzberg, S.L. Kraken: ultrafast metagenomic sequence classification using exact alignments. Genome Biol 15, R46 (2014)*

Applications: Kraken

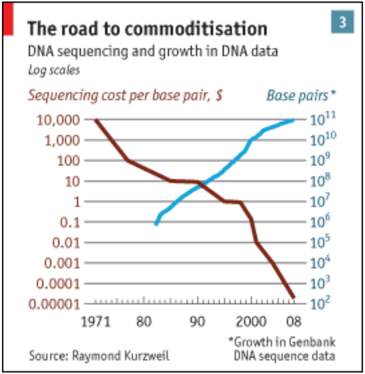
- Kraken need to store large k-mers tables ($k = 31$)
- It extensively uses k-mers in the pipeline



taxonomic unit XYZ
ATCGATCGATCGATCGATCGATCGATCGA
TCGATCGATCGATCGATCGATCGATCGAT
CGATCGATCGATCGATCGATCGATCGATC
GATCGATCGATCGATCGATCGATCGATCG
GATCGATCGATCGATCGATCGATCGATCG
ATCGATCGATCGATCGATCGATCGATCGA
TCGATCGATCGATCGATCGATCGATCGAT
. . .
CGATCGATCGATCGATCGATCGATCGATC

Problems

- cost per genome drops $\implies$ **a lot of data** (BIGSI: 13 TB)
- huge datasets need medium to long term storage for analysis
- need an efficient data structure that allows queries



The Economist

k-mers set compression (1/2)

- minimum number of bits needed to compress n k-mers[1]:

$$log \binom{4^k}{n}$$

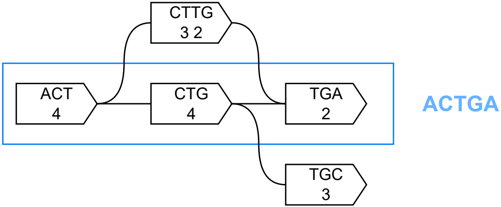
- k-mer tables and counts
  - KMC: disk files
  - DSK: hash tables
  - Squeakr: Bloom filters

- k-mer compressors:
  - approximate algorithm: UST and USTAR
  - exact algorithm: Eulertigs and Matchtigs

[1] *Conway, T.C., Bromage, A.J.: Succinct data structures for assembling large genomes. Bioinformatics 27(4), 479–486 (01 2011)*

k-mers set compression (2/2)

- A *k*-mer set compressor can use a de Bruijn Graph
  - nodes: k-mers
  - arcs: between k-mers that share a $k-1$ string
- k-mers are glued: $k-1$ characters saved per gluing
- spectrum-like property: **k-mers can be extracted** from longer strings

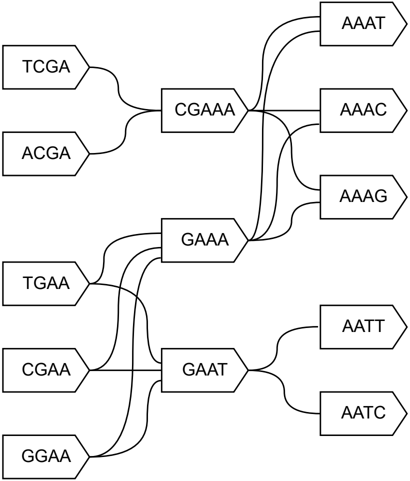


Vertex-disjoint path cover (1/3)

- *k*-mer set compression: find a path cover that contains every k-mer only once. Formally we need to solve:
- The minimum **vertex-disjoint path cover problem**: find the minimum number of vertex-disjoint paths that cover the graph.
- UST and USTAR search for an approximate solution using a greedy approach
  - the goal of USTAR is to compress sequences and counts, exploiting its skewed distribution
- Eulertigs uses an efficient and optimal algorithm: the problem is **not NP-HARD** for dBGs[1]

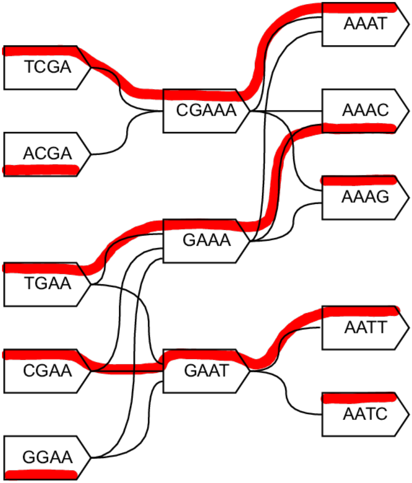
[1] *Schmidt, S., Alanko, J. N. Eulertigs: minimum plain text representation of k-mer sets without repetitions in linear time. Algorithms for Molecular Biology (2023)*

Vertex-disjoint path cover (2/3)



- Compacted dBG
- $k = 4$
- number of nodes: 13
- $CL = 4 \cdot 12 + 1 \cdot 5 = 53$

Vertex-disjoint path cover (3/3)



- USTAR chooses the following paths:
  `TCGAAAT`
  `ACGA`
  `AAAG`
  `TGAAAC`
  `CGAATT`
  `GGAA`
  `AATC`
- number of path: 7
- $CL = 4 \cdot 4 + 1 \cdot 7 + 2 \cdot 6 = 35$

Finding the minimum CL

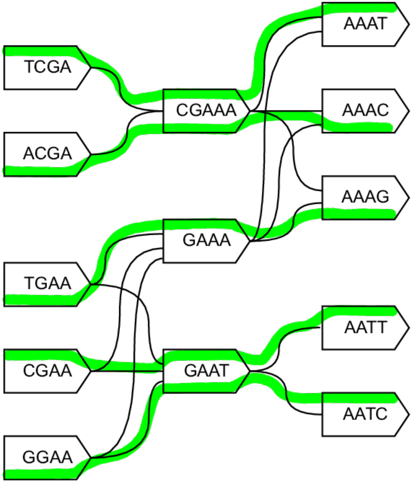
- The minimum number of paths does not necessarily correspond to the minimum cumulative length (CL)
- The “disjoint” constraint can be relaxed
  $\implies$ k-mers can be repeated
- The minimum **vertex path cover problem**: find the minimum number of vertex paths that cover the graph
  - i.e. find the minimum set of strings containing all the k-mers (with possible duplicates)
- Algorithms:
  - Matchtigs[1]: compute exact solution in $O(n^3m)$
  - Greedy Matchtigs: compute an approximate solution

[1] *Schmidt, S., Khan, S., Alanko, J. N., Pibiri, G. E., Tomescu, A. I. Matchtigs: minimum plain text representation of k-mer sets. Genome Biology (2023)*

USTAR2 (1/2)

- USTAR2 main steps:
  - select a good starting node (seed)
  - extend the seed in a full path
- the seed is choosen among the most umbalanced nodes ($|deg_{in} - deg_{out}|$) in order to start from dead ends
- the next node is chosen among the less connected ones in order to avoid congested nodes and **reduce isolated nodes**
- if there are no unvisited adjacent nodes, it explores the neighborhood with BFS until it finds a free node

USTAR2 (2/2)



- USTAR2 chooses the following paths:
  TCGAAAT
  ACGAAAC
  TGAAAG
  CGAATT
  GGAATC
- number of path: 5
- $CL = 2 \cdot 7 + 3 \cdot 6 = 32$

- $CL_{USTAR2} = 32 < CL_{USTAR} = 35 < CL_{dBG} = 53$

Experimental setup

- 20 dataset taken from major papers on k-mer compression

dataset	#15-mers	#21-mers	#31-mers	#41-mers
SRR001665_1	13,889,837	14,286,068	10,343,472	-
SRR001665_2	16,371,558	16,895,362	12,058,109	-
SRR061958_1	225,788,025	388,490,798	404,149,685	392,492,657
SRR061958_2	265,935,616	482,235,278	495,804,915	475,405,235
SRR062379_1	109,810,585	152,875,155	160,692,477	160,746,342
SRR062379_2	108,958,432	151,987,994	159,905,793	158,802,318
SRR10260779_1	84,250,397	113,667,728	123,624,245	127,090,699
SRR10260779_2	93,032,179	128,074,943	139,633,894	143,150,103
SRR11458718_1	89,998,269	126,431,861	137,995,280	143,397,012
SRR11458718_2	94,018,791	134,997,414	150,549,990	159,144,668
SRR13605073_1	43,488,336	54,085,000	55,764,573	54,682,553
SRR14005143_1	11,307,338	13,223,059	15,005,192	16,272,583
SRR14005143_2	23,691,810	28,456,533	31,850,681	33,872,511
SRR332538_1	10,624,064	11,404,027	11,382,816	10,666,430
SRR332538_2	18,741,106	25,674,930	28,880,136	27,477,871
SRR341725_1	132,442,790	188,913,254	185,618,107	176,391,089
SRR341725_2	136,484,353	196,035,961	192,133,588	181,970,438
SRR5853087_1	159,744,051	316,438,109	382,773,071	399,026,650
SRR957915_1	126,236,121	208,110,514	239,200,400	250,988,377
SRR957915_2	188,867,779	335,926,750	364,597,018	361,352,380

- Given the k-mer set S, we used the following metrics:
  - $CL = \sum_{s \in S} |s|$
  - *Compression*: size of compressed sequence file

Results: CL (k=21)

K=21	CL					
	without repeated k-mers			with repeated k-mers		
	UST	**USTAR**	**Eulertigs**	**USTAR2**	**GMatchtigs**	**Matchtigs**
SRR001665_1	36,357,928	36,324,848	**36,324,848**	33,638,588	33,858,376	**33,380,903**
SRR001665_2	45,751,142	45,694,102	**45,694,102**	41,864,643	42,201,273	**41,478,989**
SRR061958_1	623,862,618	191,039,506	**191,038,846**	**178,434,526**	179,301,460	
SRR061958_2	767,654,838	211,459,109	**211,458,409**	**198,026,097**	198,820,674	
SRR062379_1	252,418,995	248,519,235	**248,517,935**	**226,998,129**	228,491,551	
SRR062379_2	246,073,774	241,478,754	**241,477,514**	**220,352,087**	221,708,614	
SRR10260779_1	188,012,488	184,854,088	**184,851,568**	**170,629,253**	171,477,677	
SRR10260779_2	214,245,523	210,202,663	**210,200,303**	**192,382,255**	193,409,534	
SRR11458718_1	189,827,141	185,070,581	**185,068,101**	**170,218,475**	171,003,884	
SRR11458718_2	202,891,014	196,865,834	**196,863,334**	**179,815,285**	180,632,752	
SRR13605073_1	86,006,020	84,974,720	**84,973,100**	**81,822,046**	81,970,005	
SRR14005143_1	19,355,339	19,020,479	**19,020,479**	17,477,215	17,546,167	**17,376,011**
SRR14005143_2	42,328,593	41,492,693	**41,492,693**	37,213,243	37,481,708	**36,908,792**
SRR332538_1	18,649,027	18,382,747	**18,382,407**	**17,615,333**	17,688,889	
SRR332538_2	49,648,910	46,689,430	**46,689,210**	**41,053,913**	41,226,736	
SRR341725_1	245,548,134	243,816,714	**243,815,254**	**236,221,721**	236,557,911	
SRR341725_2	258,344,641	256,477,401	**256,475,521**	**247,741,588**	248,138,629	
SRR5853087_1	587,246,289	551,618,109	**551,616,269**	**484,650,727**	486,008,368	
SRR957915_1	377,292,074	366,210,794	**366,208,274**	**325,707,476**	327,968,686	
SRR957915_2	579,294,390	562,058,930	**562,056,990**	**501,809,029**	505,129,713	
average	251,540,444	197,112,537	**197,111,258**	**180,183,581**	181,031,130	–

- Eulertigs achieved optimal results with no repeated k-mer
- Even if they achieve the optimal solution, it is not possible to run Matchtigs for large datasets
- USTAR2 consistently obtains **smaller values**

Results: CL varying k

Cumulative Length		
K	**USTAR2**	**GMatchtigs**
15	122,088,205	**118,257,100**
21	**180,183,581**	181,031,130
31	**217,688,497**	218,851,662
41	**269,841,393**	271,412,813

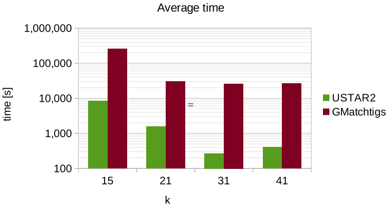
- Consider the top two tools, we vary the k-mer length
- Average results over datasets
- For $\mathbf{k} \geq \mathbf{21}$ USTAR2 achieved smaller CL

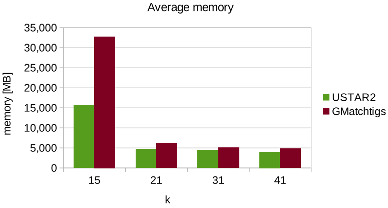
Results: Compression varying k

Compression		
K	**USTAR2**	**GMatchtigs**
15	30,659,296	**29,313,703**
21	42,115,904	**41,890,264**
31	**49,427,296**	49,672,063
41	**50,595,570**	51,137,503

- Average results over all datasets
- USTAR2 compressed better with $\mathbf{k} \geq \mathbf{31}$
- in most application it is used $k = 31$ or higher

Results: time and memory





- Greedy Matchtigs is a greedy approach but it requires a lot of time
- for $k = 15$, USTAR2 is **30**× faster while using half the memory of Greedy Matchtigs
- for $k = 31$, USTAR2 is **96**× faster while using about the same amount of memory

Conclusions

- Introduction of USTAR2, a tool for compressing k-mers sets that solves a path cover problem on a de Bruijn graph
- Achieved **compression ratios** surpass established tools like UST and USTAR, and more effective than Greedy Matchtigs for $k \geq 21$
- USTAR2 execution time is **remarkably faster** than other tools, up to 96x faster than Greedy Matchtigs, and requires **less memory**
- USTAR2 offers an effective and resource-efficient solution for compressing k-mer sets, with potential for further performance enhancement through parallelization.

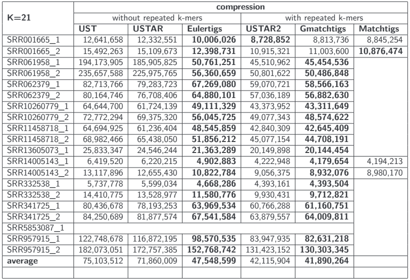
The End

Thanks for your attention!

Datasets

Dataset	Description	Read Length	#Reads	Size [GB]
SRR001665	Escherichia coli	36	20,816,448	9.304
SRR061958	Human Microbiome 1	101	53,588,068	3.007
SRR062379	Human Microbiome 2	100	64,491,564	2.348
SRR10260779	Musa balbisiana RNA-Seq	101	44,227,112	2.363
SRR11458718	Soybean RNA-seq	125	83,594,116	3.565
SRR13605073	Broiler chicken DNA	92	14,763,228	0.230
SRR14005143	Foodborne pathogens	211	1,713,786	0.261
SRR332538	Drosophila ananassae	75	18,365,926	0.683
SRR341725	Gut microbiota	90	25,479,128	1.254
SRR5853087	Danio rerio RNA-Seq	101	119,482,078	3.194
SRR957915	Human RNA-seq	101	49,459,840	3.671

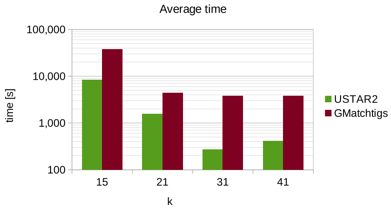
Results: Compression (k=21)



K=21	**compression**					
	without repeated k-mers			with repeated k-mers		
	UST	**USTAR**	**Eulertigs**	**USTAR2**	**Gmatchtigs**	**Matchtigs**
SRR001665_1	12,641,658	12,332,551	**10,006,026**	**8,728,852**	8,813,736	8,845,254
SRR001665_2	15,492,263	15,109,673	**12,398,731**	10,915,321	11,003,600	**10,876,474**
SRR061958_1	194,173,905	185,905,825	**50,761,251**	45,510,962	**45,454,536**	
SRR061958_2	235,657,588	225,975,765	**56,360,659**	50,801,622	**50,486,848**	
SRR062379_1	82,713,766	79,283,723	**67,269,080**	59,070,721	**58,566,163**	
SRR062379_2	80,164,746	76,708,406	**64,880,101**	57,036,189	**56,882,630**	
SRR10260779_1	64,644,700	61,724,139	**49,111,329**	43,373,952	**43,311,649**	
SRR10260779_2	72,772,294	69,375,320	**56,045,725**	49,077,343	**48,574,622**	
SRR11458718_1	64,694,925	61,236,404	**48,545,859**	42,840,309	**42,645,409**	
SRR11458718_2	68,982,466	65,438,050	**51,856,212**	45,077,154	**44,708,191**	
SRR13605073_1	25,833,347	24,546,244	**21,363,289**	20,149,898	**20,144,454**	
SRR14005143_1	6,419,520	6,220,215	**4,902,883**	4,222,948	**4,179,654**	4,194,213
SRR14005143_2	13,117,896	12,655,430	**10,822,784**	9,056,375	**8,932,076**	8,980,170
SRR332538_1	5,737,778	5,599,034	**4,668,286**	4,393,161	**4,393,504**	
SRR332538_2	14,410,775	13,528,977	**11,580,776**	9,930,431	**9,712,821**	
SRR341725_1	80,436,678	78,193,253	**63,969,534**	60,766,288	**61,160,751**	
SRR341725_2	84,250,689	81,877,574	**67,541,584**	63,879,557	**64,009,811**	
SRR5853087_1						
SRR957915_1	122,748,678	116,872,195	**98,570,535**	83,947,935	**82,631,218**	
SRR957915_2	182,073,051	172,757,385	**152,768,742**	131,423,152	**130,303,345**	
average	75,103,512	71,860,009	**47,548,599**	42,115,904	**41,890,264**	

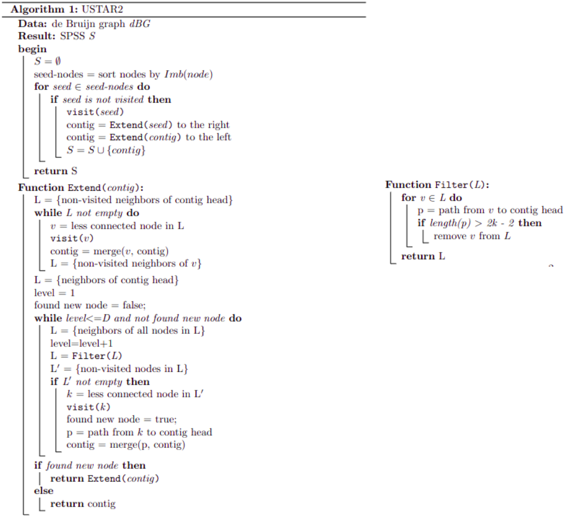
- Eulertigs achieved better compression with no repeated k-mer
- USTAR2 and Matchtigs obtained best compression only for one dataset
- GMatchtigs works best in this case (k=21)

Multithread: time



- GMatchtigs is a multithread tool
- GMatchtigs threads: 16
- USTAR2 threads: 1
- USTAR2 is still 14.06× faster

Pseudocode



```
Algorithm 1: USTAR2
Data: de Bruijn graph $dBG$
Result: SPSS $S$
begin
    $S = \emptyset$
    seed-nodes = sort nodes by $Imb(node)$
    for $seed \in seed\text{-}nodes$ do
        if $seed$ is not visited then
            visit($seed$)
            contig = Extend($seed$) to the right
            contig = Extend($contig$) to the left
            $S = S \cup \{contig\}$
    return S

Function Extend($contig$):
    L = {non-visited neighbors of contig head}
    while $L$ not empty do
        $v$ = less connected node in L
        visit($v$)
        contig = merge($v$, contig)
        L = {non-visited neighbors of $v$}
    L = {neighbors of contig head}
    level = 1
    found new node = false;
    while $level <= D$ and not found new node do
        L = {neighbors of all nodes in L}
        level=level+1
        L = Filter($L$)
        L′ = {non-visited nodes in L}
        if $L'$ not empty then
            $k$ = less connected node in L′
            visit($k$)
            found new node = true;
            p = path from $k$ to contig head
            contig = merge(p, contig)
    if found new node then
        return Extend($contig$)
    else
        return contig

Function Filter($L$):
    for $v \in L$ do
        p = path from $v$ to contig head
        if $length(p) > 2k - 2$ then
            remove $v$ from $L$
    return L
```

UNIVERSITY
OF
PADUA

DEPARTMENT
OF INFORMATION
ENGINEERING

USTAR2: Fast and Succinct Representation of k-mer Sets Using De Bruijn Graphs

Enrico Rossignolo Matteo Comin

BIOINFORMATICS2024