

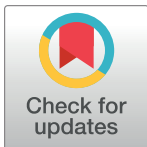
EDUCATION

What do Eulerian and Hamiltonian cycles have to do with genome assembly?

Paul Medvedev^{1,2,3*}, Mihai Pop^{4,5}

1 Department of Computer Science and Engineering, Pennsylvania State University, University Park, Pennsylvania, United States of America, **2** Department of Biochemistry and Molecular Biology, Pennsylvania State University, University Park, Pennsylvania, United States of America, **3** Center for Computational Biology and Bioinformatics, Pennsylvania State University, University Park, Pennsylvania, United States of America, **4** Department of Computer Science, University of Maryland, College Park, Maryland, United States of America, **5** Center for Bioinformatics and Computational Biology, University of Maryland, College Park, Maryland, United States of America

* pzm11@psu.edu



Abstract

Many students are taught about genome assembly using the dichotomy between the complexity of finding Eulerian and Hamiltonian cycles (easy versus hard, respectively). This dichotomy is sometimes used to motivate the use of de Bruijn graphs in practice. In this paper, we explain that while de Bruijn graphs have indeed been very useful, the reason has nothing to do with the complexity of the Hamiltonian and Eulerian cycle problems. We give 2 arguments. The first is that a genome reconstruction is never unique and hence an algorithm for finding Eulerian or Hamiltonian cycles is not part of any assembly algorithm used in practice. The second is that even if an arbitrary genome reconstruction was desired, one could do so in linear time in both the Eulerian and Hamiltonian paradigms.

OPEN ACCESS

Citation: Medvedev P, Pop M (2021) What do Eulerian and Hamiltonian cycles have to do with genome assembly? *PLoS Comput Biol* 17(5): e1008928. <https://doi.org/10.1371/journal.pcbi.1008928>

Editor: Francis Ouellette, University of Toronto, CANADA

Published: May 20, 2021

Copyright: © 2021 Medvedev, Pop. This is an open access article distributed under the terms of the [Creative Commons Attribution License](https://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

Funding: This material is based upon work supported by the National Science Foundation under Grants given to PM numbered 1439057, 1453527, and 1931531. MP was supported in part by the NIH, grant R01-AI-100947, and the National Science Foundation under grants IIS-0812111 and IIS-0844494. The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Competing interests: The authors have declared that no competing interests exist.

Introduction

When you learned about genome assembly algorithms, you might have heard a story that goes something like this:

In the overlap-layout paradigm, solving the assembly problem requires solving the Hamiltonian cycle problem in the overlap graph. This is difficult, because the Hamiltonian cycle program is NP-hard. On the other hand, if we break our reads up into k-mers, we can build the de Bruijn graph. Then, the assembly problem becomes the problem of finding an Eulerian cycle in the de Bruijn graph, which is easily solvable in linear time. Thus, by formulating the assembly problem in terms of the de Bruijn graph, we can solve the much easier Eulerian cycle problem and not have to solve the NP-hard Hamiltonian cycle problem.

In this paper, we explain that while de Bruijn graphs have indeed been very useful, the reason has nothing to do with the complexity of the Hamiltonian and Eulerian cycle problems.

We will first define the terms necessary to understand the above story. A Hamiltonian cycle in a graph is a cycle that visits every vertex at least once, and an Eulerian cycle is a cycle that

visits every edge once. In general graphs, the problem of finding a Hamiltonian cycle is NP-hard, while finding an Eulerian cycle is solvable in polynomial time. Consider a set of reads R . An overlap graph constructed from R is a directed graph where every vertex is a read, and there is an edge from x to y if and only if a suffix (of a certain minimal length, which is given as a parameter) of x is equal to a prefix of y . Under certain idealized assumptions of the input, which we will not get into here, the genome is spelled by a cycle that visits every read exactly once, i.e., a Hamiltonian cycle of the overlap graph. Alternatively, a de Bruijn graph of R is a directed graph with a vertex for every k -long substring of R (called a k -mer) and an edge for every $(k+1)$ -long substring of R . Under certain idealized assumptions, the genome is spelled by a cycle that visits every edge of a de Bruijn graph exactly once, i.e., an Eulerian cycle.

The answer to the question

Every Eulerian cycle in a de Bruijn graph or a Hamiltonian cycle in an overlap graph corresponds to a single genome reconstruction where all the repeats (long sequences that appear more than once) are completely resolved (i.e., their place in the genome determined). For example, Fig 1 shows 2 different Eulerian cycles in the same graph (a similar example could be constructed for Hamiltonian cycles in an overlap graph). Each cycle corresponds to a different arrangement of segments between the repeats. The presence of multiple Eulerian or Hamiltonian cycles implies that the genome structure is ambiguous given the data available. In other words, using the same set of reads, one can reconstruct different genomes, each of which is fully consistent with the data (Fig 1 gives an example). Choosing one of these reconstructions arbitrarily would be foolhardy since only one of them is the original genome. No sane assembly algorithm would do this, and that is one of the major reasons why an algorithm for finding Eulerian or Hamiltonian cycles is not part of any assembly algorithm used in practice.

Instead, assemblers output contigs—long, contiguous segments which can unambiguously be inferred to be part of the genome. Finding such segments is a very different computational problem than finding a single Eulerian or Hamiltonian cycle (see Endnote 1). In fact, it was shown that finding all possible contigs can be done in polynomial time, regardless of whether the genome reconstruction is modeled as a Hamiltonian or Eulerian cycle [1]. The algorithm used in practice (the unitig algorithm) is linear and nearly identical in the 2 graph models.

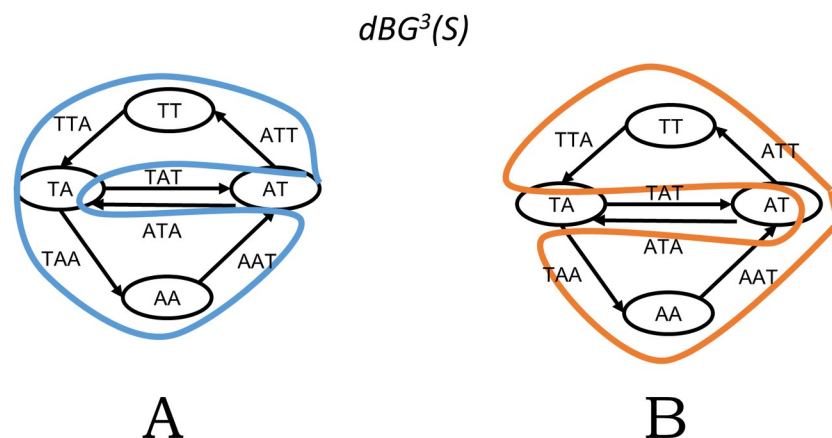


Fig 1. A worked out example for a set of reads $R = \{TATTA, TAATA\}$ and $k = 3$. Here, the set of all k -mers is $S = sp^k(R) = \{TAT, ATT, TTA, TAA, AAT, ATA\}$. Panel A shows $G_1 = dBG^k(S)$ and one possible Eulerian cycle of G_1 (in blue). Panel B show the only other Eulerian cycle in G_1 (in orange). The genome reconstruction corresponding to the blue cycle is $ATTAATAT$ and to the orange cycle is $ATTATAAT$ (note that because the genome is circular, the last 2 characters of each string are equal to the first 2 characters).

<https://doi.org/10.1371/journal.pcbi.1008928.g001>

Perhaps you are not convinced by the above reasoning? Fine. For the sake of argument, let us imagine that we really are interested in finding a single, arbitrary, genome reconstruction. But even in this case, the distinction between Eulerian and Hamiltonian cycles is misleading. We make our point with this theorem, which we first state informally (a formal statement and proof will come later):

Main Theorem (informal): The following problems are equivalent and solvable in linear time:

1. Find an Eulerian cycle in the de Bruijn graph where the edges correspond to k -mers in the reads.
2. Find a Hamiltonian cycle in the de Bruijn graph where the edges correspond to all the possible $(k+1)$ -mers that can be obtained from the reads' k -mers.

The first part of the theorem should not be surprising. It states one half of the story we started with, namely that we can solve the assembly problem in linear time by finding an Eulerian cycle in a de Bruijn graph. The second part of the theorem, though, adds a twist. It is about finding a Hamiltonian cycle, but it differs from the initial story in 2 ways. First, it is a Hamiltonian cycle in a de Bruijn graph, not in an overlap graph. This might seem strange, but there is no special connection between overlap graphs and the Hamiltonian cycle problem—one is free to find a Hamiltonian cycle in any graph they wish. Second, the problem is solvable in linear time in this case, even though it is NP-hard in general. This might also seem strange, but in fact it is common for NP-hard problems to have polynomial time solutions for a restricted class of inputs. For example, the satisfiability problem is NP-hard in general but is polynomial time solvable when the clauses are restricted to have only 2 variables.

What the theorem states, then, is that one can solve the assembly problem in linear time by finding a Hamiltonian cycle within an appropriately defined de Bruijn graph. The fact that the Hamiltonian cycle problem is NP-hard in general graphs is not directly relevant. What is important is the underlying structure of the de Bruijn graph which makes the Hamiltonian cycle problem easy to solve (for more details, see Endnote 2). Hence, the initial story was right in the sense that using de Bruijn graphs is a good idea but wrong to imply that the complexity of the Hamiltonian cycle problem is a reason. All of this is of course assuming we are, for some reason, interested in an arbitrary genome reconstruction, which, as we argued earlier, we typically are not.

Formal statement and proof of main theorem

We will now give some definitions to prove the main theorem. Let t be a string, k be a positive integer, and S be a set of k -mers. Let R be a set of reads, i.e., strings of length $\geq k$. We define $pre_i(t)$ and $suf_i(t)$ as the prefix and suffix, respectively, of length i of t . The k -spectrum of R , denoted by $sp^k(R)$, is the set of all k -mer substrings of the strings of R . The de Bruijn graph of order k of S , denoted as $DBG^k(S)$, is defined as follows. The vertex set is $sp^{k-1}(S)$, and for every k -mer $x \in S$, we add an edge from $pre_{k-1}(x)$ to $suf_{k-1}(x)$. Fig 1 shows an example of a de Bruijn graph. See Endnote 3 for more for some context about how the term de Bruijn graph is used more broadly. We define the *closure* of S , denoted $closure(S)$, to be the set of all $(k+1)$ -mers y such that $pre_k(y) \in S$ and $suf_k(y) \in S$. Informally, it is the set of all $(k+1)$ -mers that can be constructed from S .

The main theorem follows almost directly from definitions. The proof here is based on first principles, for expository purposes, but it is actually a corollary of deeper results (see Endnote 4).

Main Theorem (formal): Let R be a set of strings whose smallest length is l . Let k be a positive integer less than l . Then, there is a one-to-one correspondence between Eulerian cycles in

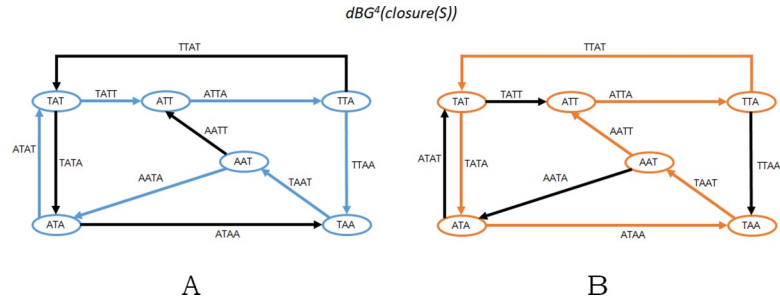


Fig 2. Using the same example as in Fig 1, this figure shows $G_2 = dBG^{k+1}(closure(S))$ and the 2 possible Hamiltonian cycles in G_2 . Notice that the k -mer sequence corresponding to the blue cycle in Panel A is the same as in Fig 1A; similarly, the k -mer sequence for the orange cycle in Panel B is the same as in Fig 1B.

<https://doi.org/10.1371/journal.pcbi.1008928.g002>

$dBG^k(sp^k(R))$ and Hamiltonian cycles in $dBG^{k+1}(closure(sp^k(R)))$. Moreover, an Eulerian or Hamiltonian cycle can be found in its respective graph in $O(|sp^k(R)|)$ time.

Proof: Let $S = sp^k(R)$. Let $G_1 = dBG^k(S)$ and $G_2 = dBG^{k+1}(closure(S))$. First, we show that the vertex set of G_2 is S , i.e., $sp^k(closure(S)) = S$. Clearly, $sp^k(closure(S)) \subseteq S$, since no new k -mers are created during the closure process. Now, let x be a k -mer in S . It must appear in some read r , and since the length of r is greater than k , x must be a prefix or suffix of some $(k+1)$ -mer y in R . Moreover, y must be in $closure(S)$, since its prefix and suffix are both in r and hence in S . Therefore, $x \in sp^k(closure(S))$, completing our proof that $S \subseteq sp^k(closure(S))$ and the vertex set of G_2 is S .

Observe that a sequence of k -mers $C_1 = x_0, \dots, x_{n-1}$ is a sequence of edges defining an Eulerian cycle in G_1 if and only if the set of k -mers of C_1 is exactly S (without any repetitions) and, for all i , $suf_{k-1}(x_i) = pre_{k-1}(x_{i+1 \bmod n})$. Also, observe that a sequence of k -mers $C_2 = x_0, \dots, x_{n-1}$ is a sequence of vertices defining a Hamiltonian cycle in G_2 if and only if the exact same criteria holds, i.e., the set of k -mers of C_2 is exactly S (without repetitions) and, for all i , $suf_{k-1}(x_i) = pre_{k-1}(x_{i+1 \bmod n})$. Thus, there is a one-to-one correspondence between Eulerian cycles in $dBG^k(S)$ and Hamiltonian cycles in $dBG^{k+1}(closure(S))$. Fig 2 shows an example.

For the running time, an Eulerian cycle can be found in time linear in the number of edges using a classical algorithm, e.g., Hierholzer’s Algorithm. Giving the one-to-one correspondence above, the vertex labels of a Hamiltonian cycle in G_2 can be found by outputting the edge labels of an Eulerian cycle in G_1 . Hence, the running times for the 2 problems are equivalent.

Conclusions

So why are de Bruijn graphs so popular for short read assembly, if not for the difference in the complexity of finding Eulerian or Hamiltonian cycles? The answer is complex, which might explain why the initial simple story was appealing. It may have to do with the simplicity of their implementation, the appeal of the k -mer abstraction, the ease of error correction, or with something else. In fact, the difference between using de Bruijn graphs and overlap graphs is poorly understood and is a fascinating open research problem. But, the Eulerian and Hamiltonian cycle dichotomy is not really relevant to assembly or to the popularity of de Bruijn graphs.

Endnotes

1. For those who are curious, the problem of finding all possible contigs was first mentioned in [2] and later explored by [1] and other follow-up papers. In this line of research, the set of substrings which must appear in all possible genome reconstructions are called omnitigs.

For a broader discussion on how to formulate the genome assembly problem, see the tutorial by Medvedev [3].

2. There is much more that has been said on the complexity of finding a single genome reconstruction. For a starting point, see the papers [2,4–6]. The last 2 papers [5,6], especially, show how the complexity depends more on the repeat structure of the underlying genome than on the graph model used.
3. The definition of de Bruijn graph that we give here is sometimes referred to as the edge centric dBG, not to be confused with a node centric one (see <https://www.biostars.org/p/175058/#256741> for an explanation of the 2 terms). Historically, the term “de Bruijn graph” had a different definition, which is still used today in other contexts, especially outside of bioinformatics. There, the “full” de Bruijn graph of order k is not defined with respect to a set of k -mers S , but, instead, has the edge set as the universe of all k -mers [7,8]. The de Bruijn graph as defined here is then the subgraph induced by the set of edges S in the full de Bruijn graph.
4. The main theorem can be viewed as a corollary of a similar result for full de Bruijn graphs (see Endnote 3) and of the relationship between Eulerian cycles in a digraph G and Hamiltonian cycles in the line digraph of G . The full de Bruijn graph of order $k+1$ is the line digraph of the full de Bruijn of order k (Lemma 2.5.1 in [9]). Using this fact, de Bruijn already observed in his original paper that there is a one-to-one correspondence between Eulerian cycles in the full de Bruijn of order k and Hamiltonian cycles in the full de Bruijn graph of order $k+1$ [3]. The proof in this paper can be viewed as first arguing that G_2 is the line digraph of G_1 and then using the same type of argument as de Bruijn [3] to show the one-to-one correspondence.

Acknowledgments

PM would like to thank Rayan Chikhi for feedback on the text and Alexandru Tomescu and Michael Brudno for many helpful discussions on this topic (and Michael specifically for introducing him to the problem a long time ago).

References

1. Tomescu AI, Medvedev P. Safe and complete contig assembly through omnitigs. *J Comput Biol*. 2017; 24(6):590–602. <https://doi.org/10.1089/cmb.2016.0141> PMID: 27749096
2. Nagarajan N, Pop M. Parametric complexity of sequence assembly: theory and applications to next generation sequencing. *J Comput Biol*. 2009; 16(7):897–908. <https://doi.org/10.1089/cmb.2009.0005> PMID: 19580519
3. Medvedev P. Modeling biological problems in computer science: a case study in genome assembly. *Brief Bioinformatics*. 2019; 20(4):1376–83. <https://doi.org/10.1093/bib/bby003> PMID: 29394324
4. Medvedev P, Georgiou K, Myers G, Brudno M. Computability of models for sequence assembly. In: *International Workshop on Algorithms in Bioinformatics*. Berlin, Heidelberg: Springer; 2007. p. 289–301.
5. Kingsford C, Schatz MC, Pop M. Assembly complexity of prokaryotic genomes using short reads. *BMC Bioinformatics*. 2010; 11(1):21. <https://doi.org/10.1186/1471-2105-11-21> PMID: 20064276
6. Bresler G, Bresler M, Tse D. Optimal assembly for high throughput shotgun sequencing. *BMC Bioinformatics*. 2013; 14(S5):S18. <https://doi.org/10.1186/1471-2105-14-S5-S18> PMID: 23902516
7. De Bruijn N. A Combinatorial Problem. *Koninklijke Nederlandse Akademie V Wetenschappen*. 1946; 49:758–64.
8. Good IJ. Normal recurring decimals. *J Lond Math Soc*. 1946; 21(3):167–9.
9. Bang-Jensen J, Gutin G. *Digraphs: theory, algorithms and applications*. Springer Science & Business Media; 2008.