JOURNAL OF COMPUTATIONAL BIOLOGY
Volume 16, Number 7, 2009

Pp. 897–908
DOI: 10.1089/cmb.2009.0005

Parametric Complexity of Sequence Assembly: Theory and Applications to Next Generation Sequencing

NIRANJAN NAGARAJAN and MIHAI POP

ABSTRACT

In recent years, a flurry of new DNA sequencing technologies have altered the landscape of genomics, providing a vast amount of sequence information at a fraction of the costs that were previously feasible. The task of assembling these sequences into a genome has, however, still remained an algorithmic challenge that is in practice answered by heuristic solutions. In order to design better assembly algorithms and exploit the characteristics of sequence data from new technologies, we need an improved understanding of the *parametric complexity* of the assembly problem. In this article, we provide a first theoretical study in this direction, exploring the connections between repeat complexity, read lengths, overlap lengths and coverage in determining the "hard" instances of the assembly problem. Our work suggests at least two ways in which existing assemblers can be extended in a rigorous fashion, in addition to delineating directions for future theoretical investigations.

Key words: genome assembly, sequencing, parametric complexity, Chinese Postman problem.

1. INTRODUCTION

RESEARCH ON GENOME ASSEMBLY (the task of piecing together sequence "reads" into a reconstruction of the genome) has been re-invigorated by recent technological advances in DNA sequencing. Within the last 2 years, six new *de novo* genome assemblers have been described in the literature, targeted specifically at the assembly of data generated by new short-read sequencing technologies (Warren et al., 2007; Jeck et al., 2007; Dohm et al., 2007; Butler et al., 2008; Hernandez et al., 2008; Zerbino and Birney, 2008). Note that these programs do not implement radically new algorithms, but rather represent heuristic improvements to assembly algorithms developed in the 90s. As new sequencing technologies are being developed it is now imperative that the theoretical foundations of the assembly problem be revisited in order to better clarify the relationship between the characteristics of the data being generated and the computational difficulty of obtaining a correct assembly. Some questions that come to mind include: Can we trade-off read length for coverage and still retain the ability to assemble a genome? Can technologies that promise ultra-long reads make the algorithmic task of assembly less challenging? What kind of mate-pair libraries are best suited to aid in assembly?

Center for Bioinformatics and Computational Biology, Institute for Advanced Computer Studies, University of Maryland, College Park, Maryland.

In general, the assembly problem is considered to be computationally challenging and several formulations have been shown to be NP-hard, either as a generalization of the Shortest Common Superstring problem or in the context of an overlap graph formulation (Kececioglu and Myers, 1995). These hardness results have, at least partly, driven the development of heuristic solutions for assembly, several of which have been quite successful—for example, Phrap (Green, 1994), Celera Assembler (Myers et al., 2000), ARACHNE (Batzoglou et al., 2002)—and have had a significant impact on modern biological research. The success of heuristic approaches to genome assembly suggests that the hardness results indeed represent worst-case scenarios, situations that rarely occur in data-sets encountered in practice. As observed by Bodlaender et al. (1995), several problems in computational biology are such that a small range of values for certain parameters cover the interesting cases. Is this also the case for genome assembly?

In this article, we provide a first theoretical analysis of the impact of several parameters of the data being assembled on the computational complexity of the assembly problem. We specifically focus on the interplay between the lengths of sequencing reads and the length of repetitive sequences in the genome being assembled. Furthermore we explore the impact of sequencing coverage and read-overlap constraints (defined below) on the difficulty of the assembly problem. A better understanding of the impact of these parameters on the complexity of genome assembly can lead to the development of new assembly algorithms that rely on heuristics only when necessary. More importantly, however, these results can help establish boundaries on the parameters of sequencing data that enable their efficient and accurate assembly, thereby driving experimental design.

1.1. Background

The popular whole-genome shotgun (WGS) approach to sequencing, relies on sequencing randomly sheared fragments of DNA to produce **reads** whose order and orientation (the strand which they came from) are unknown. To ensure that every region of the genome is represented in reads, the genome is oversampled several times, a number referred to as the **coverage** of the genome. Several sequencing technologies also allow for the generation of **mate-pairs** where reads come in pairs with a known approximate distance between them. The **assembly** process uses these sources of read information to computationally reconstruct the genome.

The assembly task has been modelled using various mathematical and graph-theoretic formulations: one of the earliest approaches modelled the problem as one of finding the **Shortest Common Superstring** (SCSP) of the reads, based on a parsimony assumption. SCSP is a well-known NP-hard problem and its connection to sequence assembly spurred the development of several approximation algorithms for it (Blum et al., 1994; Sweedyk, 2000; Kaplan and Shafrir, 2005). An important drawback of the SCSP formulation is that it handles repeat sequences unintuitively as tandem repeats in the genome get collapsed in the shortest superstring reconstruction.

Other approaches to assembly have sought to correct for this, by formulating the assembly problem as one of finding suitable paths in graphs constructed from the reads. In the overlap graph/string graph approach (Kececioglu and Myers, 1995; Myers, 2005), nodes correspond to reads and edges define valid tilings of the reads. A reconstruction of the genome then corresponds to a path that uses all the reads (traverses all the nodes in the graph). An alternative approach, requires the construction of a de Bruijn graph (see Section 2) of all k-mers (Pevzner et al., 2001). Reads now correspond to paths in this graph and the genome corresponds to a suitable super-path (tiling of paths) based on the reads.

In recent work, Medvedev et al. (2007) studied the complexity of sequence assembly in the string graph as well as the de Bruijn graph framework. In the string graph case, they model the problem as one of finding a minimum s-walk (a walk that satisfies edge constraints of the form ''optional/required'') on a general weighted directed graph and show that this problem is NP-hard. They however provide no evidence to suggest that string graphs constructed from reads could lead to arbitrary directed graphs, leaving the complexity question open on string graphs.[1] In the de Bruijn graph case, Medvedev et al. (2007) show that finding the shortest super-walk, a path that has all the read paths as sub-paths, is an NP-hard problem. The super-walk formulation allows for assemblies that are read-incoherent—parts of the super-walk could be

[1]Braga and Meidanis (2002) provide an exponential time algorithm to construct strings that produce a desired overlap graph.

made of incomplete fragments of the reads—but this can be easily fixed as we show next. We begin by defining the graphs and paths of interest to us.

2. DEFINITIONS

2.1. Overlap and string graph

Given two strings x and y over the alphabet $\sum$, x is said to **overlap** y if a suffix of x matches[2] a prefix of y. If $o(x,y)$ is the length of the longest such match then the **overlap graph** of a set of strings S is a weighted directed graph[3] where each string is a vertex and the edge $x \rightarrow y$ is in the graph with a weight of $|y| - o(x,y)$ if $o(x,y) \geq k$, for some minimum **overlap** threshold k. For any path in this graph, its **path-string** is given by the concatenation of the strings corresponding to the nodes on the path, where only one copy of the overlap string is kept. The **string graph** $SG^k(S)$ of a set of strings S can be obtained from the overlap graph by removing contained reads (reads that are substrings of other reads) and transitively inferable edges (an edge between strings x and y, s.t. the corresponding string is identical to the string for an alternate path from x to y), in polynomial time (Myers, 2005) (**transitive reduction**).

For a set of strings S we formulate the **Assembly Problem** (AP) as one of finding a generalized Hamiltonian path[4] (every node is visited at least once) in $SG^k(S)$ s.t. the length of the path-string is minimized. We will refer to the solution of AP, or specifically, the path-string corresponding to it as the **genome** of interest. While AP is a special case of the Travelling Salesman Problem it can still be shown to be NP-hard.

Theorem 1. *The Assembly Problem is NP-hard.*

Proof *(reduction from SCSP similar to Medvedev* et al.*, 2007).* Let S_I be a set of strings from an instance of SCSP. We produce an instance of AP as follows: for a string of the form abc in S_I we create a read $R^k a R^k b R^k c R^k$, where R is a special character (in other words, we insert R^k between every character in the original string and at the ends of the string). Clearly every generalized Hamiltonian path with path-string of length l produces a superstring of length $(l-k)/(k+1)$. Conversely, it is easy to verify that any minimal superstring (a string s.t. the removal of any substring destroys its superstring property) corresponds to a generalized Hamiltonian path in the string graph. Hence the absence of a generalized hamiltonian path of length $\leq l$ implies that there does not exist a superstring of length $\leq (l-k)/(k+1)$. ■

The string R^k in the reduction plays the role of a **repeat** in the genome, i.e., a maximal substring of length $\geq k$ that occurs more than once in the genome. Note that because of the way we have defined repeats the string $ab^{k+5}cb^{k+5}d$ has only one repeat of length $k+5$. Also, we define a **repeat instance** as being one of the copies of the repeat in the genome and associate it with the corresponding minimal path in the string graph used to construct it (repeat instances can be overlapping substrings of the genome).

2.2. Superpaths and the de Bruijn graph

The **de Bruijn graph** $DG^k(S)$ of a set of strings S is given by a node set composed of all length k substrings of strings in S. Two nodes x and y have an edge between them if the length $k-1$ suffix of x is a prefix of y. Let r_{max}, r_{min} be the lengths of the longest and shortest string respectively in S. A read in S now corresponds to a path in $DG^k(S)$ of length in the range $[r_{min} - k,\ r_{max} - k]$.

The analagous formulation for assembly in de Bruijn Graphs is to find a super-path (a path constructed from overlapping paths) that uses each of the read paths at least once. We call this the **de Bruijn Superpath Problem** (DBSP), to distinguish from the superwalk problem (BSP) proposed in Medvedev et al., (2007), where a superwalk could in principle contain subwalks that are read-incoherent. DBSP and AP are essentially identical problems and the NP-hardness proof for AP also translates into one for DBSP.

---

[2]This includes inexact matches as well, though in some cases we assume an exact match and this will be clear from context.

[3]For simplicity we work with directed graphs though our results should extend to the bidirected case as well.

[4]For handling circular genomes, we can switch to Hamiltonian cycles and prove similar results.

3. GENOMIC COMPLEXITY AND THE ASSEMBLY PROBLEM

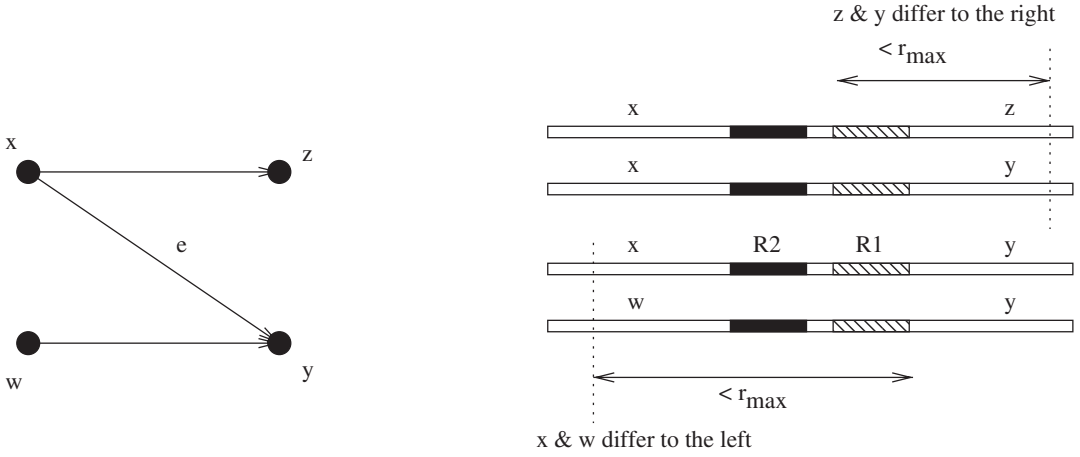
An important determinant of the complexity of the Assembly Problem is the repeat structure of the genome being sequenced. For example, the string graph corresponding to a circular genome with no repeats is a cycle and the Assembly Problem has a trivial solution. In practice, some genomes are more difficult to correctly assemble than others and typically this is attributed to the presence of repeats that are much longer than the typical read length. This difficulty to assemble, however, does not arise from computational complexity. Somewhat counter-intuitively, the Assembly Problem can actually be solved in polynomial time for a genome where all repeats are long (specifically, $\geq 2r_{max} - k - 1$) and we formalize this in the following theorem:

Theorem 2. *If all repeats in the genome are long ($\geq 2r_{max} - k - 1$), then AP can be solved in polynomial time.*

Proof. Consider all the edges in the string graph that are unique in-edges or out-edges to a node. Since we need a generalized Hamiltonian path, each of these edges is part of the path corresponding to the genome and we call them **required edges**. Any edge that is not a required edge in the graph is considered an **optional edge**. Suppose the string graph contains an optional edge $e = x \rightarrow y$. By definition, $\exists w \neq x$, $w \rightarrow y$ and $\exists z \neq y$, $x \rightarrow z$ (Fig. 1a). Let R_1 be the string corresponding to the k-mer suffix of x. By construction R_1 is a substring of y and z and thefore the genome must have at least two copies of R_1. To see this, note that if there is only one copy in the genome then either the edge $y \rightarrow z$ or $z \rightarrow y$ must be in the graph and transitive reduction would have removed one of $x \rightarrow y$, $x \rightarrow z$. Now, the repeat defined by R_1 extends by atmost $r_{max} - k - 1$ to the right as $y \neq z$. The repeat must therefore extend by at least $r_{max} - k$ to the left because otherwise we would have a repeat of length $< 2r_{max} - k - 1$, and therefore it must contain the overlap of x and y. By a similar argument, if R_2 is the k-mer prefix of y then it defines a repeat that extends by atmost $r_{max} - k - 1$ to its left and contains the overlap of x and y. Since these repeats are limited in extent on opposite ends, intersecting these repeats gives a repeat of length $< 2r_{max} - k - 1$ (Fig. 1b), and hence we get a contradiction. Hence, all edges in the string graph are required and AP translates into finding a **Chinese Postman path** which can be solved with a well-known polynomial time algorithm. ■

The following corollary to this theorem clarifies the picture for repeats longer than r_{min} and can be proven using a similar argument:

Corollary 1. *If all repeats in the genome are longer than r_{min} and all repeat instances with length $\geq r_{min}$ and $< 2r_{max} - k - 1$ have a read contained within them, then AP can still be solved in polynomial time (as an instance of the Chinese Postman problem).*



a String graph induced by an optional edge **b** Repeat instances induced by the optional edge

FIG. 1. Repeats and optional edges in the string graph.

While finding *a* Chinese Postman path is easy, the string graph may have many paths visiting all vertices, with the same minimum length. In such cases, the Assembly Problem is essentially underconstrained—any of these paths could be a reconstruction of the genome. Additional information such as mate-pairs can then be used to resolve the ambiguity in the genome and we discuss this issue in Section 6. In this context, it would be useful to have a procedure to test for the presence of multiple Chinese Postman paths. One such procedure was proposed by Skiena and Sundaram (1993) that can test for uniqueness of paths of length l in $O(l^3)$ time. Here we propose an alternate procedure: a well-known solution to the Chinese Postman problem requires finding a minimum weight bipartite matching in a graph constructed from unbalanced (# in-edges $\neq$ # out-edges) vertices in the original graph. This matching is then used to augment the original graph with multi-edges such that an Eulerian tour in the multi-graph gives a Chinese Postman tour in the original graph. It is easy to verify that in order for the Chinese Postman tour to be unique both the matching and Eulerian tours have to be unique. The uniqueness of weighted bipartite matchings can be tested using a linear-time algorithm proposed by Gabow et al. (1999), while Hutchinson and Wilf (1975) give a formula for counting the number of Eulerian Tours in a graph. Suprisingly, counting the number of Chinese Postman tours is a harder problem (see Appendix for proof):

Theorem 3. *The counting version of the Chinese Postman problem is #-P complete.*

An important practical issue here is that despite ambiguities in global reconstruction, there could still be regions in the string graph which are part of every Chinese Postman reconstruction. In this case it would be reasonable to report these partial assemblies. In the case of Eulerian tours, reconstructing sub-tours that are part of every Eulerian tour is feasible in polynomial time (see Theorem 7.5 in Waterman, 1995) based on finding acyclic subgraphs in the cycle-graph decomposition[5] of the original graph. Constructing assemblies in this fashion can dramatically increase the size of the resulting sequences and this has practical implications for existing short-read assemblers as we discuss in Section 7. Note that reconstructing sub-tours that are part of every Chinese Postman tour can be achieved as a two-step process:

1. Finding the sub-matching in the bipartite graph of unbalanced nodes that is part of every pefect matching (using augmenting paths).
2. Augmenting the graph with this sub-matching and finding sub-tours that are part of every Eulerian tour of the balanced subgraph.

We now turn our attention to the case of short repeats (length $< r_{min}$)—clearly their presence makes some overlap edges in the string graph optional. However, just having short repeats in the genome adds structure to the Assembly Problem as any solution must now visit every node exactly once (to see this note that if a node is visited twice the genome will have a repeat $\geq r_{min}$). Loosely speaking, the problem is translated from a **Chinese Postman superpath problem** to an **Eulerian superpath problem**. The presence of short repeats is generally not considered a hinderance in the reconstruction of the genome, especially if there are reads "spanning" the repeat. We formalize this notion below:

Definition 1. *A* **spanned instance** *of a repeat is one such that the path associated with it in the string graph is a single node. A* **uniquely spanned instance** *is one such that the corresponding node has only required edges into and out of it.*

Note that in the case of short repeats, the string graph can be simplified by iteratively removing optional edges coming out of a node that has a required edge coming out (and correspondingly for in-edges). The following theorem can then be applied to the simplified string graph (see Appendix for proof):

Theorem 4. *AP can be solved in polynomial time (with a unique solution) if all repeats are short (length* $< r_{min}$*) and all repeat instances are uniquely spanned.*

Despite the good news in the above theorem, the Assembly Problem, even when restricted to genomes with relatively short repeats (length $< 2r_{max} - 1$), is a computationally hard problem as we show next. Put

[5]Edge-disjoint decomposition into simple cycles where the cycles are represented as nodes s.t. two nodes have an edge between them if the corresponding cycles in the original graph share a node.

another way, what this means is that contrary to popular belief, the ability to produce very long perfect reads will not necessarily make the computational problem of assembly solvable in polynomial time.

Theorem 5. *The shortest superstring of a set of strings S cannot have a repeat of length* $\geq 2r_{max} - 1$.

Proof. Suppose it does, then consider an instance of the repeat I. Let o_p be the longest prefix of I overlapping a strict suffix of a string in S (i.e., not containing the whole string). Correspondingly, let o_s be the longest suffix of I overlapping a strict prefix of a string in S. By definition $|o_p| + |o_s| \leq 2r_{max} - 2$ and therefore removing the substring between o_p and o_s in I produces a shorter superstring (all strings not contained completely in I are still substrings by construction and those that are contained in I are present in other copies of the repeat) which gives a contradiction. ■

Corollary 2. *AP is NP-hard even when all repeats in the genome are short (length* $< 2r_{max} - 1$*).*

Proof. This follows directly from the above theorem and the reduction in Theorem 1. ■

4. WHY DOES COVERAGE MATTER?

Despite the computational complexity of the Assembly Problem, the "take home message" from Corollary 1 and Theorem 4 is that with high enough coverage, the ambiguities introduced by short repeats (length $< r_{min}$) and slightly longer repeats (length in the range $[r_{min}, 2r_{max} - k - 2]$) can be resolved. In the general Assembly Problem with both long and short-repeats, we no longer have to use all edges (as in the long repeat case) or visit each node exactly once (as in the short repeat case). Despite this, with the right kind of reads, the problem becomes easy. We formalize this in the following Corollary which is a direct consequence of Corollary 1 and Theorem 4.

Corollary 3. *The Assembly Problem can be solved as an instance of the Chinese Postman problem if all short repeat instances (length* $< r_{min}$*) are uniquely spanned and all longer repeat instances (length in the range* $[r_{min}, 2r_{max} - k - 2]$*) have a read contained within them.*

The ability to formulate a polynomial time exact solution in the AP model of assembly also encourages us to consider more general models of assembly. In these models, the relative coverage of a region of the genome also provides information as to how often the region is likely to appear in the genome. While taking this information into account can be valuable to produce a more realistic reconstruction of the genome based on the data, it can also be misleading if coverage is low or non-uniform. Fortunately, new sequencing technologies often afford very high ($>20X$) and largely uniform coverage making such analysis feasible (see Section 7 for an example).

To incorporate coverage information into AP a natual approach is to estimate a **copy-number** for every region of the genome (the number of times a node must be visited in the string graph) and to find a path that matches those constraints. However, this leads to an NP-hard problem as even finding a Chinese Postman path of a given length (inferred from the copy-numbers) is NP-hard (Skiena and Sundaram, 1993). An alternate approach is to use soft-constraints. Such an approach was proposed by Medvedev and Brudno (2008) essentially as a way to estimate copy-numbers. For an assignment of copy-numbers to nodes in the string graph, they scored the assignment as the likelihood under a multinomial distribution parameterized by the observed read counts. They then describe a procedure to efficiently optimize this scoring function by using min-cost flows. The min-cost flow formulation allows for lower-bounds on the flow on edges[6]: setting these to 1 allows us to compute a flow that directly gives us a Chinese Postman path that optimizes the scoring function (to see this, note that disjoint flow cycles are not possible as the graph is connected and every edge has non-zero flow). Thus the high-coverage regime for assembly allows us to not only effectively tackle the path-finding model for assembly, but it also allows us to generalize to a model where relative coverage of reads is accounted for.

[6]In Medvedev and Brudno (2008), in the absence of additional information, they were forced to set these to 0.

5. ASSEMBLY WITH SHORT-READS

Our investigation of the complexity of the Assembly Problem until now has largely ignored the overlap parameter k. However, clearly, this parameter plays an important part in the computational complexity of the problem – if $k = r - 1$ then reads correspond to edges in the de Bruijn graph formulation and DBSP translates into a Chinese Postman problem (where $r_{min} = r_{max} = r$). For the parameter, $D = r - k$, we therefore know that AP is easy if $D = 1$ and NP-hard for arbitrary D. (Note that the reduction in Section 2.1 required the construction of reads s.t. $D = k * (l + 1)$ where l is the length of the strings.) This opens up the tantalizing possibility of a polynomial-time (in the number of reads) algorithm for AP for a fixed D. This question is particularly relevant in the case of short-read assembly where to avoid false overlaps, the minimum overlap parameter can be quite high (say, $k = 20$) and in combination with the short reads ($r \approx 30$) this can lead to potentially feasible values for D ($D \approx 10$ in this case). The parameter D is also related to coverage: with high coverage we can afford to set a small D without worrying about fragmenting the assembly. We clarify the complexity of the Assembly Problem as a function of D in the following theorems (see Appendix for proof of Theorem 6):

Theorem 6. *The Assembly Problem is NP-hard when $r \geq 2k + 4$ (fixed-size alphabet, $|\sum| = 2$).*

Theorem 7. *The Assembly Problem is NP-hard even when $r - k \geq 4$ (without the restriction to a fixed-size alphabet).*

Proof *(reduction from HAM-CYCLE, similar to Blum* et al.*, 1994).* Given an instance of the Hamiltonian Cycle Problem we construct the reads for AP as follows: for each vertex v in the graph we add two letters v and v' to the alphabet and construct a read $N_v = vR^{r-2}v'$ where R is a special character. Let $w_0, \ldots, w_{n-1}$ be the vertices incident to v. Then, we also have reads of the form $E_{v,w_i} = v'w_iR^{r-4}v'w_{i-1}$ and $R^jv'w_iR^{r-2-j}$, $2 \leq j \leq r - 4$ where subscript arithmetic is modulo n.

The string graph for this set of reads contains a subgraph of the form shown in Fig. 2 for each vertex v. Given a solution to HAM-CYCLE, i.e., a path $v_1, \ldots, v_n$ we can construct a path in the string graph as follows: for each edge (v_i, v_{i+1}) in the Hamiltonian path we create a sub-path that traverses through the widget associated with the adjacency list of v_i and exits at v_{i+1} (Fig. 2). Putting these paths together we get

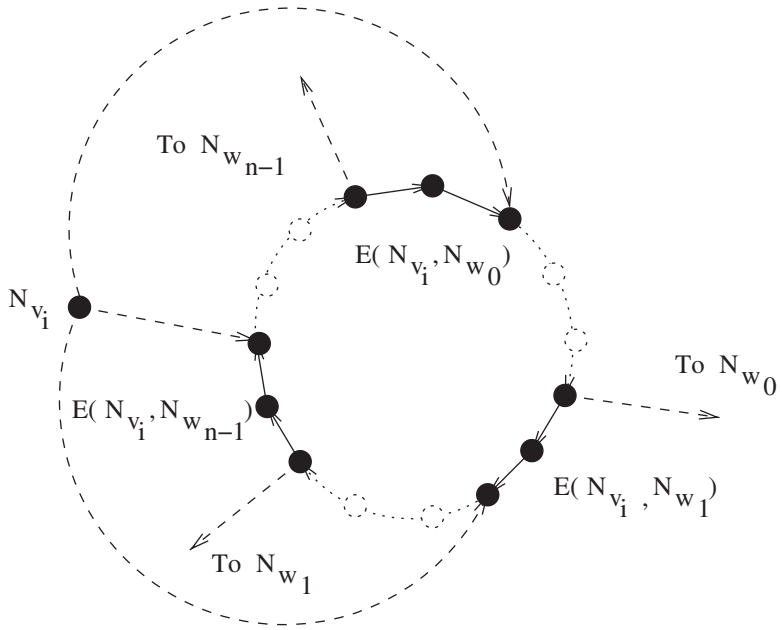


FIG. 2. Widget for each vertex v_i in Thm. 7. Note that we use parentheses to indicate subscripts.

a path in the string graph whose associated string is of length $r|V| + (r-2)|E|$ where V,E are the vertex and edge sets, respectively. It is easy to verify that no path with length $r|V| + (r-2)|E|$ visits a node corresponding to a read of the form $vR^{r-2}v'$ more than once.

Conversely, given a solution to AP whose associated string is of length $r|V| + (r-2)|E|$ we construct a solution to HAM-CYCLE as follows: for a vertex v_i let v_{i+1} be the vertex s.t. after visiting N_{vi}, the path in the string graph, first visits the vertex following $E_{vi,vi+1}$ (among the labelled vertices). Note that this is well-defined as after visiting N_{v_i} the path has to visit all the nodes associated with the widget for v_i (N_{v_i} is visited only once and it is the only entry point for the widget). ■

Corollary 4. *There is no algorithm polynomial in 2^{r-k} for AP unless P = NP (even for an alphabet of size 3).*

Proof. The alphabet required in the previous reduction is of size $2|V| + 1$. These characters can be coded into a binary alphabet in $O(log(|V|))$ space where the representations of the original characters are seperated by a third special character #. In this new representation, a string of length $r-k$ in the previous reduction has length $O(log(|V|))$ and hence an algorithm polynomial in 2^{r-k} provides a solution to HAM-CYCLE that is polynomial in $|V|$ which is not possible unless P = NP. ■

Conjecture 1. *There is a polynomial-time algorithm for AP if $D < k$ is considered a constant.*

6. ASSEMBLY WITH MATE-PAIRS

In the absence of very long reads ($\gg$1 Kbp), mate-pairs with large inserts have played an essential role in the assembly of large and repeat-rich genomes. A standard use of mate-pairs is to link together ungapped sequences (**contigs**) into larger (**scaffold**) sequences that may have gaps but represent an ordered and oriented region of the genome (this task was shown to be NP-hard in Huson et al., 2001). Another popular use of mate-pairs is to disambiguate repeats by finding unique paths in the de Bruijn graph that satisfy the distance constraints (Pevzner et al., 2001) (this task reduces to the NP-hard longest-path problem). An alternate approach, that follows naturally from the discussion in Section 3, is to use them to constrain AP in cases where there are many possible solutions. While a detailed study of this formulation is beyond the scope of this paper, we consider the ideal case when AP can be transformed into a Chinese Postman problem.

As mentioned in Section 3, a first step in an algorithm to find a Chinese Postman path in a graph G is to identify a min-weight bipartite matching in a bipartite graph G' constructed from the unbalanced nodes in G. The edges in the bipartite graph have weights corresponding to the shortest path in G between the corresponding unbalanced nodes. In this framework, the information from mate-pair distances can be naturally incorporated by removing edges where the path doesn't satisfy these contraints. Similarly, once the graph has been augmented (based on the matching) to produce an instance of the Eulerian Path problem, mate-pairs can be used to constrain the problem. Specifically, ambiguities in the Eulerian Path problem manifest as cycles in the cycle-graph decomposition. For each such cycle, we can then test edges in the cycle to see if any of the shortest paths that traverse the edge (in the original graph) are too long to satisfy the distance constraints. We thus have a polynomial-time solution for using mate-pair information to constrain the ideal case of the Assembly Problem; we hope to extend it in our future work.

7. DISCUSSION

The results presented in the previous sections have several practical implications and we highlight a few here:

Common subpaths. A common paradigm in assembly, especially in the context of short-read assemblers is to assemble regions of the string graph comprised of nodes with single in and out edges into contigs. The implicit assumption here is that repeats, particularly those longer than the read length, create a fundamentally unresolvable problem. However, as discussed in Section 3, this assumption is not necessary and we could potentially reconstruct much longer sequences. A particularly striking example of this can be seen in the Newbler assembly (www.454.com) for the 1.1-Mbp genome of the bacterium *R. Prowazekii Madrid*

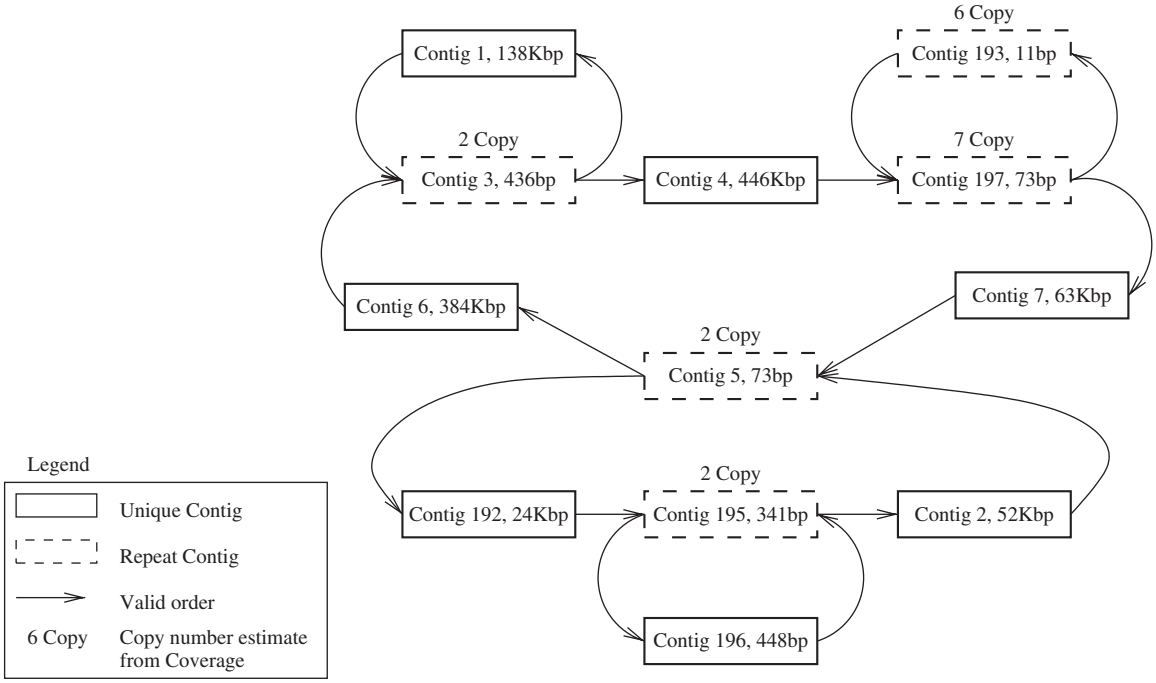


FIG. 3. *R. Prowazekii* Contig graph.

E (Fig. 3). As can be seen from a visual inspection of Figure 3, the graph has a unique Eulerian tour, giving the genome in "one piece" despite the presence of long repeats, e.g., Contig 3 (read length $\approx$ 200bp). An implementation of the procedure described in Section 3 to identify subpaths contained in every Chinese Postman path can therefore serve as useful tool to produce much larger contigs from the output of existing short-read assemblers.

Coverage or read-length? The example in Figure 3 also serves to highlight the benefits of the high-coverage regime for assembly ($> 100X$ in this case). Corollary 3 essentially tells us that we should expect the assembly task to be much easier in this regime and this also explains the typical behavior seen in short-read assembly where assemblies improve dramatically in quality beyond a certain coverage threshold. As discussed in Section 4, the high-coverage regime also opens up the possibility of investigating more realistic models to incorporate coverage information in the assembly process and this is an interesting direction for future work. Note that, even in the presence of low-coverage, the results in Section 3 suggest that subgraphs composed of required edges are easier to resolve and this needs further study. Finally, Corollary 2 clarifies that longer reads do not necessarily produce an easier computational problem.

Mate-pairs and future work. A consequence of our work is that it clearly demarcates directions for future theoretical investigations: while an efficient algorithm for AP for large values of D seems unlikely, we would still like to settle Conjecture 1 about the existence of a polynomial time algorithm for constant values of D, the **overhang** parameter. As sequencing becomes cheaper, high coverage can be taken as a given. The effective use of mate-pairs to further constrain the assembly problem is then of particular interest.

8. APPENDIX

Proof of Theorem 3

Proof. We provide a reduction from the #-P complete problem of counting perfect bipartite matchings (Valiant, 1979). Given a directed bipartite graph $G = (L \cup \mathrm{R},\ E)$, $|L| = |R|$, we construct an instance of the Chinese Postman problem as follows: the original bipartite graph is copied and updated by connecting each vertex v, of in-degree d, in R with $d + 1$ out-edges from a super-node x. Vertices in L are similarly linked with

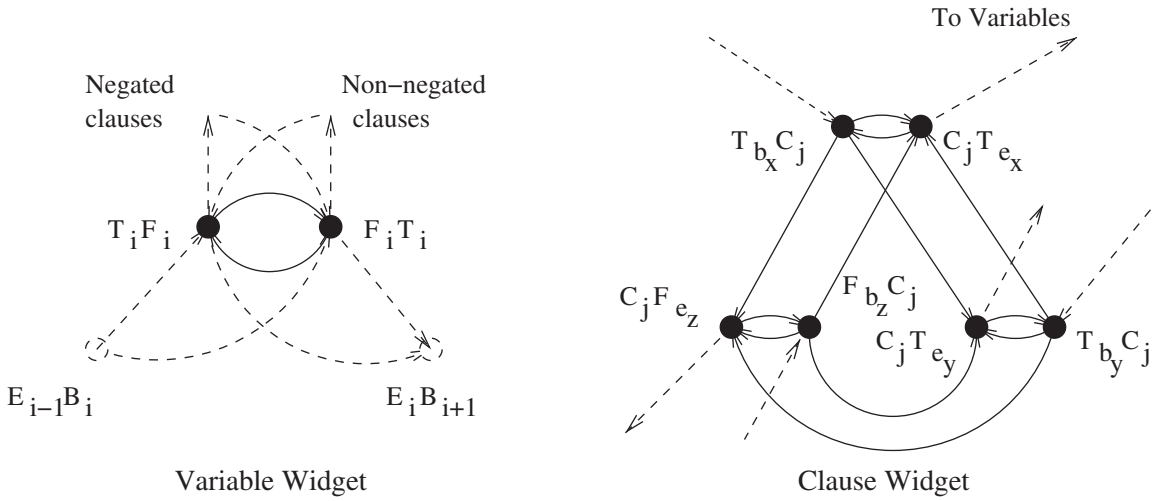


FIG. 4. Variable and clause widgets for the reduction in Theorem 6.

in-edges to a super-node y and finally we add $|E| + |L|$ edges $x \to y$ to get the graph G'. Now note that every perfect bipartite matching in G provides an "augmentation" s.t. duplicating these edges in G' makes every node balanced. The resulting graph is Eulerian and every augmentation leads to the same constant number of Eulerian tours. Since every Chinese Postman tour in G' can be constructed as an augmentation followed by an Eulerian tour, counting Chinese Postman tours in G' leads to a count for perfect matchings in G. ■

Proof of Theorem 4

Proof. Let $e = x \to y$ be an optional edge in the string graph that is part of the path representing the genome. From the argument in the proof for Thm 2, e contains an instance of a repeat (i.e., e is the path corresponding to the repeat instance). Since all repeat instances are uniquely spanned, there exists a node v s.t. $f = x \to v$, $g = v \to y$, f and g are required edges and this path can be used to replace e (in fact, e is transitively inferable and hence will not be part of the string graph). From this argument, we can see that the simplified string graph has no optional edges. Since all remaining edges are required and since every node is visited exactly once, we know that every node has a unique edge coming out of it i.e. the edges in the graph give a unique reconstruction of the genome. ■

Proof of Theorem 6

Proof *(reduction from 3-SAT*[7]*).* Consider an instance of 3-SAT consisting of size 3 clauses $C = c_0, \ldots, c_{m-1}$ over variables $V = v_0, \ldots, v_{n-1}$. Let $k = 4n + 3m$, $o = k + 2$, $r = 2o$ and $s(i) = 1^{o-i}0^i$, $e(i) = 01^{o-i}0^{i-1}$, $b(i) = 1^{o-i-1}0^i1$. Now, for each variable v_i we have 8 marker strings $B_i = s(4i)$, $F_i = s(4i+1)$, $T_i = s(4i+2)$, $E_i = s(4i+3)$, $Fb_i = b(4i+1)$, $Fe_i = e(4i+1)$, $Tb_i = b(4i+2)$, $Te_i = e(4i+2)$ and for each clause c_j the marker string $C_j = s(4n + 3j + 2)$. The reads for AP are constructed from the marker strings as follows: each variable v_i gives rise to two reads T_iF_i and F_iT_i and if v_i is a variable in clause c_j then we construct the reads, $Fb_i\ C_j, C_jFe_i$ if v_i is negated and Tb_iC_j, C_jTe_i otherwise. Finally, we link the variables v_i, v_{i+1} using reads of the form E_iB_{i+1}.

The string graph for these reads is composed of variable and clause **widgets** as shown in Figure 4. Suppose that a satisfying truth assignment exists, then associate each clause with exactly one of the true literals. We can now construct a Hamiltonian path in the string graph as follows: for each variable, v_i we

[7]For a similar reduction, see Middendorf (1998), where they show that the shortest superstring problem is NP-complete even when strings are restricted to be of the form 10^p10^q.

construct a sub-path and string together these sub-paths (using reads of the form E_iB_{i+1}). If v_i is **true** in the satisfying assignment then the subpath visits the node corresponding to F_iT_i followed by the nodes for each of the clause widgets satisfied by v_i, finally ending in T_iF_i, otherwise the path begins with T_iF_i and ends with F_iT_i. It is easy to verify that the path-string for this path is of length $3o(2m+n)+3n+6m$ and this is the shortest string reconstructible.

Conversely, if there is a solution to AP providing a path of length $3o(2m+n)+3n+6m$, then we can construct a satisfying truth assignment as follows: for each variable widget, if F_iT_i is visited first, after visiting $E_{i-1}B_i$, then we set the variable to **true** and **false** otherwise. Note that since every node can be visited exactly once (its easy to verify that otherwise you get a longer string), the node following $E_{i-1}B_i$ is the only one that can be followed by a path into clause widgets and these specify the clauses satisfied by this variable. ■

ACKNOWLEDGMENTS

The authors were supported in part by IIS-0812111 and by the Henry Jackson Foundation grant 0000150300, both to M.P.

DISCLOSURE STATEMENT

No competing financial interests exist.

REFERENCES

Batzoglou, S., Jaffe, D., et al. 2002. ARACHNE: a whole-genome shotgun assembler. *Genome Res.* 12, 177–189.

Blum, A., Jiang, T., Li, M., et al. 1994. Linear approximation of shortest superstrings. *J. ACM* 41, 630–647.

Bodlaender, H., Downey, R., Fellows, M., et al. 1995. Parameterized complexity analysis in computational biology. *Comput. Appl. Biosci.* 11, 49–57.

Braga, M., and Meidanis, J. 2002. An algorithms that builds a set of strings given its overlap graph. *Proc. LATIN 2002*.

Butler, J., MacCallum, I., and et al., 2008. ALLPATHS: *de novo* assembly of whole-genome shotgun microreads. *Genome Res.* 133–146.

Dohm, J., Lottaz, C., Borodina, T., et al. 2007. SHARCGS, a fast and highly accurate short-read assembly algorithm for *de novo* genomic sequencing. *Genome Res.* 17, 1697–1706.

Gabow, H.N., Kaplan, H., and Tarjan, R.E. 1999. Unique maximum matching algorithms. *Proc. STOC 1999* 70–78.

Green, P. 1994. PHRAP documentation. *Algorithms*.

Hernandez, D., Francois, P., and et al. 2008. *De novo* bacterial genome sequencing: millions of very short reads assembled on a desktop computer. *Genome Res.* 18, 802–809.

Huson, D.H., Reinert, K., and Myers, E.W. 2001. The greedy path-merging algorithm for sequence assembly. *Proc. RECOMB 2001* 157–163.

Hutchinson, J., & Wilf, H. 1975. On Eulerian circuits and words with prescribed adjacency patterns. *J. Comb. Theory* 18, 80–87.

Jeck, W., Reinhardt, J., and et al. 2007. Extending assembly of short DNA sequences to handle error. *Bioinformatics* 23, 2942–2944.

Kaplan, H., and Shafrir, N. 2005. The greedy algorithm for shortest superstrings. *Inf. Process. Lett.* 93, 13–17.

Kececioglu, J.D., and Myers, E. W. 1995. Combinatorial algorithms for DNA sequence assembly. *Algorithmica* 13, 7–51.

Medvedev, P., and Brudno, M. 2007. Ab initio whole genome shotgun assembly with mated short reads. *Proc. RECOMB 2008* 50–64.

Medvedev, P., Georgiou, K., Myers, G., et al. 2007. Computability of models for sequence assembly. *Proc. WABI 2007* 289–301.

Middendorf, M. 1998. Shortest common superstrings and scheduling with coordinated starting times. *Theor. Comput. Sci.* 191, 205–214.

Myers, E., Sutton, G., et al. 2000. A whole-genome assembly of *Drosophila. Science* 287, 2196–2204.

Myers, E.W. 2005. The fragment assembly string graph. *Bioinformatics* 21, 79–85.
Pevzner, P., Tang, H., and Waterman, M. 2001. An Eulerian path approach to DNA fragment assembly. *Proc. Natl. Acad. Sci. USA* 98, 9748–9753.
Skiena, S., and Sundaram, G. 1993. Reconstructing strings from substrings [extended abstract]. *WADS '93* 565–576.
Sweedyk, Z. 2000. a $2\frac{1}{2}$-approximation algorithm for shortest superstring. *SIAM J. Comput.* 29, 954–986.
Valiant, L. 1979. The complexity of computing the permanent. *Theor. Comput. Sci.* 8, 189–201.
Warren, R., Sutton, G., Jones, S., et al. (2007). Assembling millions of short DNA sequences using SSAKE. *Bioinformatics* 23, 500–501.
Waterman, M.S. 1995. *Introduction to Computational Biology*. Chapman & Hall, New York.
Zerbino, D.R., and Birney, E. 2008. Velvet: algorithms for *de novo* short read assembly using de Bruijn graphs. *Genome Res.* 18, 821–829.

Address correspondence to:
Dr. Mihai Pop
Center for Bioinformatics and Computational Biology
Institute for Advanced Computer Studies
University of Maryland
College Park, MD 20742

E-mail: mpop@umiacs.umd.edu