Minimum Paired-End Interval Cover and Its Application

Liang Ding
Department of Computer Science
University of Georgia
Athens, GA 30602
USA
Email: adamdingliang@gmail.com

Bin Fu
Department of Computer Science
University of Texas-Pan American
Edinburg, TX 78541
USA
Email: binfu@cs.panam.edu

Binhai Zhu
Corresponding author
Department of Computer Science
Montana State University
Bozeman, MT 59717, USA
Email: bhz@cs.montana.edu

***Abstract*—Paired-end sequencing is a very useful method for the whole genome sequencing problem which determines the complete DNA sequence of an organism's genome with the help of laboratory processes. A paired-end (PE) interval for a sequence S is composed of at most two disjoint intervals. Due to the cheap cost to obtain PE-intervals (also known as *short reads*), in practical datasets a letter in some genome to be sequenced can be covered by at least M PE-intervals. In this paper, we consider the m-fold paired-end interval cover problem ($m \leq M$), which can be defined as given a family $\mathbb{F}$ of paired-end intervals on a sequence S such that each letter in S is covered at least M times, find the minimum number of paired-end intervals of $\mathbb{F}$ to cover each letter in S at least m times. We prove that the (1-fold) paired-end interval cover problem is NP-complete. We present a polynomial-time 12-approximation algorithm for the case when $m = 1$, which is based on greedy search. This result also generalizes the set-cover problem (an element in the base set becomes an interval which could contain a number of elements) and a simple fixed parameter tractable algorithm is presented for the related k-bounded c-interval cover problem. We then generalize the algorithm and analysis to the general m-fold PE-interval cover problem, to have a factor of $2c(4m-1)$. Our implementation results show that the practical approximation ratios are mostly bounded by 2.2 for examples constructed from real datasets.**

Keywords: computational genomics, set cover, short reads, approximation algorithms, FPT algorithms, NP-completeness.

I. Introduction

In the past two decades, we have seen a huge progress on genome sequencing. The problem we investigate in this paper, PE-interval cover, is derived from the paired-end sequencing method which was developed based on shotgun sequencing. Shotgun sequencing is a method used for sequencing long DNA strands. The paired-end sequencing method comes a bit after shotgun sequencing, it is also known as double-barrel shotgun sequencing and was first described by Edwards and Caskey in 1991 [7]. In paired-end sequencing, sequences are determined from both ends of random subclones derived from a DNA target. The benefit of this method toward shotgun sequencing is the information obtained by sequencing both ends of a fragment of DNA, commonly known as *short reads*, could be more useful. After the work of Edwards and Caskey, many variants of the strategy have been developed by several groups [8], [9], [10], [11], [12], [13].

Paired-end sequencing has been used successfully to sequence small genomic targets, such as microbial genomes and large-insert subclones of large genomes [14], [15], [16]. In 1999, heuristic algorithms were provided by Anson and Myers to handle whole genome shotgun sequencing [17]. In 2002, Huson *et al.* proved that the problem of sequencing a genome with short reads is NP-complete and presented a heuristic greedy method [24]. This method was improved recently by Gao *et al.* [25].

Recently, we started to investigate this problem in a more formal way. The first step is to identify the most useful (i.e., non-redundant) information from the input, which typically contains errors and redundancy. We formulate this as the *PE-interval cover* problem. It turns out that the results for the PE-interval cover problem can be generalized to the c-interval cover problem which extends the classical set-cover problem when c is arbitrary. We then take the practical condition into account by considering the m-fold PE-interval cover problem.

The set-cover problem is a classical and fundamental problem in computer science with many applications. Given a finite set X of size n and a family $\mathbb{F}$ of subsets of X such that every element of X belongs to at least one subset in $\mathbb{F}$, the problem is to find a minimum-size subset $\mathbb{C} \subseteq \mathbb{F}$ whose members cover X. It is a well-known NP-complete problem showed in Karp's 21 NP-complete problems in 1972 and its solutions give rise to the development of the entire field of approximation algorithms [1]. The well known approximation algorithm for the set-cover problem which has a ratio of $\ln n + O(1)$ was given by Johnson [2] in 1974. Chvatal [3] improved the upper bound on the approximation ratio to $\ln n - \ln\ln n + O(1)$ in 1979. These ratios are tight because of a famous inapproximability result of Feige [4] which states that there is no $(1-\varepsilon)\ln n$-approximation algorithm for the set-cover problem unless there are subexponential $O(n^{polylog(n)})$ time deterministic algorithms for all problems in NP. A lower bound of $c \ln n$ was established recently, where c is a constant, under the weaker assumption that P$\neq$NP [6], [1].

The general set-cover problem has two interesting variants: k-set cover and k-bounded set cover. We derive some results from these variants of set cover. Before presenting these results, we first give the formal definitions of the c-interval cover problem and the paired-end interval cover problem. Let

$|A|$ be the size of set A, we have:

Definition 1: Assume that S is a sequence that has no repetition with its elements, i.e. S is a permutation. Let c be an integer parameter.

- A *c-interval* $I = \{J_1, J_2, \ldots, J_t\}$ of S is a series of disjoint non-empty intervals $J_i (1 \leq i \leq t)$ of S which satisfies that $|I| = t \leq c$.
- The *c-interval cover problem* is defined as given a family $\mathbb{F}$ of c-intervals, find the least number of c-intervals of $\mathbb{F}$ to cover S.
- The *m-fold c-interval cover problem* is defined as given a family $\mathbb{F}$ of c-intervals on S such that each element in S is covered at least M times, find the minimum number of c-intervals of $\mathbb{F}$ to cover S such that each element of S is covered at least m $(m \leq M)$ times.

The paired-end interval cover problem is a special case of the c-interval cover problem for $c = 2$ and $m = 1$. For simplicity, we call the paired-end interval cover problem the PE-interval cover problem. The m-fold paired-end interval cover problem is more practical in PE-based genome sequencing. Due to the low cost of obtaining paired-end reads, a lot of short reads are obtained such that each element is covered at least a certain number (e.g., 200) of times. In this case, it makes sense to have a minimum-size covering which covers each element at least $m \leq 200$ times.

We comment that our results hold even when S is not a permutation. Also, in some real datasets there is an additional information on the distance between two short reads. But this information does not help on the PE-interval cover problem; in fact, it can be easily seen that our NP-completeness proof holds even if this distance information is given. However, this distance information might be useful when we try to sequence a genome (or re-order a set of contigs) later on.

When each interval contains exactly one element, the paired-end interval problem reduces to 2-set cover. In *k-set cover*, every subset of X is of size at most k. When $k = 2$, the 2-set cover can be reduced to the maximum matching problem using the following two steps:

1) Find a maximum matching in a graph constructed according to the given 2-sets: create a vertex for each element, and there is an edge between two vertices if there is a 2-set consisting of this pair of elements.
2) Return all the 2-sets corresponding to the edges of the maximum matching and the 1-sets of the uncovered elements (by the collection of 2-sets which we found).

It is known that the maximum matching problem is solvable in polynomial time. So for PE-interval cover, if each interval is degenerated to a single point, it is equivalent to the 2-set cover and can be solved in polynomial time. However, we prove that the general PE-interval cover problem is NP-complete. Using the fact that the k-set cover problem is APX-hard for $k \geq 3$ [19], we prove, in Section V, that the c-interval cover problem is APX-hard if $c \geq 3$. (It is well known that the 1-interval cover problem is polynomially solvable as it can be formulated as a shortest path problem.) Moreover, a polynomial time $6c$-approximation algorithm is provided to solve the c-interval cover problem for $c \geq 2$.

We define that one element of X occurs once if it appears in exactly one of the subsets of $\mathbb{F}$. Then *k-bounded set cover* satisfies that the number of occurrences of any element of X in $\mathbb{F}$ is bounded by a constant $k \geq 2$ and there exists an element which appears in $\mathbb{F}$ exactly k times. The best approximation algorithm has a factor of $\frac{k}{k-(k-1)\sqrt[k]{1-\varepsilon}} + o(1)$ [5]. In this paper, we show a fixed parameter tractable algorithm for the k-bounded set cover problem, and hence it can be used directly to solve k-bounded c-interval cover.

The rest of the paper is organized as follows: Section II gives fundamentals on approximation algorithms and FPT algorithms. Section III gives an approximation algorithm for the c-interval cover problem using a greedy method, and then we generalize it to the m-fold c-interval cover problem. In Section IV, we show a fixed parameter tractable algorithm for the k-bounded set cover problem and the k-bounded c-interval cover problem. The complexities of the PE-interval cover problem and the c-interval cover problem are presented in Section V. Then, some experimental results are presented in Section VI, for $m = 1$ and $m = 30$ respectively. Section VII concludes the paper.

II. Preliminaries

For completeness, we present some basic definitions regarding approximation algorithms and FPT (fixed-parameter tractable) algorithms.

For a minimization problem Π, a polynomial-time algorithm B is a factor α approximation if for any instance $\Pi(I)$, the solution returned by B, $B(I)$, satisfies $|B(I)| \leq \alpha \cdot O^*(I)$, where $O^*(I)$ is the corresponding optimal solution value. (For maximization problems, this can be defined similarly.)

Definition 2: [20] The *class APX* is the set of optimization problems that allow polynomial time approximation algorithms with approximation ratios bounded by a constant.

- If there is a polynomial time algorithm to solve a problem within any constant factor greater than one, then the problem is said to have a *polynomial time approximation scheme (PTAS)*. Unless P=NP, there are problems that are in APX but not in PTAS; that is, problems that can be approximated within some constant factor, but not every constant factor.
- A problem is said to be *APX-hard* if there is a linear reduction from every problem in APX to that problem, and to be *APX-complete* if the problem is APX-hard and also in APX.

While approximation algorithms provide a way to handle NP-hard problems, in many situations we can handle them with exact algorithms. For a decision problem D with parameter k, an FPT (fixed-parameter tractable) algorithm is one which solves D in time $O(f(k)n^c)$ where $f(-)$ is any function only on k and c is independent of k. The underlying idea of the FPT theory is that if k is fixed (typically as a small constant, like 30) then the corresponding FPT algorithm can solve D practically in polynomial time. For more information, the readers are referred to some standard textbook [23].

III. Greedy Approximation Algorithms for the c-interval Cover Problems

In this section, we derive a $6c$-approximation algorithm for the c-interval cover problem using a greedy method. We then generalize the algorithm and analysis to the m-fold c-interval cover problem, to have a factor of $2c(4m-1)$. Theoretically, these approximation ratios are big, but they have much better performance in practice (at least from our simulation results). We will show our experimental results in Section VI.

A. Approximation Algorithm for the c-interval Cover Problem

Definition 3: Let U be the set which includes all the uncovered elements of S, and let $I = \{J_1, J_2, \ldots, J_t\}$ be a c-interval of $\mathbb{F}$.

- The *improvement* of interval J_i for S is defined as $|J_i \cap U|$.
- The *single improvement* of c-interval I for S is defined as $\max_{1\le i\le t} |J_i \cap U|$.
- The *sum improvement* of c-interval I for S is defined as $\sum_{1\le i\le t} |J_i \cap U|$.

The idea of our greedy algorithm is to select a c-interval I_i in $\mathbb{F}$, at each step i, such that I_i makes the biggest single improvement for S. The detailed algorithm is as follows.

Single Improvement Greedy Algorithm

Input: a genome sequence S and a family $\mathbb{F}$ of c-intervals $I_i (1 \le i \le |\mathbb{F}|)$.

Output: a subset C of $\mathbb{F}$ that covers S.

Steps:

1 Let $U \leftarrow S$, $C \leftarrow \emptyset$.
2 While $U \neq \emptyset$,
3 Select a c-interval I_i from $\mathbb{F}$ such that a single interval J_m of I_i covers the most uncovered elements in S, i.e., $|J_m \cap U|$ is maximized.
4 $U \leftarrow U - \{\cup_{J_j \in I_i} J_j\}$
5 $C \leftarrow C \cup \{I_i\}$
6 Return C.

End of Algorithm

Since the number of iterations of the loop on lines 2–7 is bounded from above by $|\mathbb{F}|$, and the loop body can be implemented to run in time $O(\sum_{I_i\in\mathbb{F}} |I_i|)$, there is an implementation that runs in time $O(|\mathbb{F}| \sum_{I_i\in\mathbb{F}} |I_i|)$. The following theorem shows that it is a constant factor approximation algorithm.

Theorem 1: There is a polynomial time $6c$-approximation algorithm for the c-interval cover problem.

Proof: Let O be an optimal solution for the c-interval cover problem. Also, let A be the approximate solution computed by our algorithm. The optimal solution contains at most $c \times |O|$ intervals and hence at most $2 \times c \times |O|$ endpoints. Let E be the set of endpoints of these intervals of O. The algorithm repeatedly selects a c-interval I_i such that a single interval J_m of I_i covers the most uncovered elements of S; or, in other words, I_i makes a maximum single improvement.

WLOG, every such interval J_m contains at least one endpoint $e \in E$. To see this, take any element $x \in J_m$ and observe that x must be covered by at least one interval J_k in the optimal solution. If J_m does not contain any endpoint of J_k then J_m is completely contained in J_k. Thus, the algorithm would select the c-interval containing J_k instead of the c-interval containing J_m.

On the other hand, every endpoint $e \in E$ can be in at most three intervals J_m that make a maximum single improvement. Namely, there can be one interval that makes a maximum single improvement to both sides of e, one interval that makes a maximum single improvement to the left of e and one interval that makes a maximum single improvement to the right of e. After selecting such three intervals, there can be no more interval containing e that makes a maximum single improvement, or one of the previous selections would not be maximum.

Therefore, because every c-interval in A contains at least one endpoint from E (in the interval making maximum single improvement), and each endpoint from E can be selected (in an interval making a maximum single improvement) at most three times,

$$|A| \le 3 \times |E| \le 3 \times 2 \times c \times |O|.$$

Thus our algorithm is a $6c$-approximation. ■

For the single improvement greedy algorithm, we can make a slight change, i.e., instead of selecting a c-interval that makes the biggest single improvement for S, we select a c-interval that makes the biggest sum improvement for S. This slight change makes the algorithm a special case of the greedy algorithm for the set-cover problem which has a $\ln n + O(1)$ factor [2]. If we call this algorithm *sum-improvement greedy algorithm*, the same as for set-cover, this algorithm has an approximation factor $\ln |S| + O(1)$.

B. Approximation Algorithm for the m-fold c-interval Cover Problem

For the m-fold c-interval cover problem, we can easily generalize the above two greedy algorithms. For the single improvement variation (for the m-fold c-interval cover problem), we can prove that it has an approximation factor of $2c(4m-1)$, which is $O(1)$ when m, c are fixed.

Definition 4: Let $\mathbb{F}$ be a set of c-intervals covering each position (letter) of S at least M times. Let $P \subseteq \mathbb{F}$ and m be a given integer bounded by M. Abusing the terminology a bit, for a c-interval $Q \in P$ with $J \in Q$ (i.e., J is one of the c intervals of Q), we also write $J \in P$ when the context is clear.

- If a position (letter/element) i of S is covered $t < m$ times by intervals in P, and another interval $J \notin P$ covers i (denoted as $i \in J$), we say that the position i incurs a *cover improvement* by J.
- Let p be a position (letter) of S, define $cover(p, P) = |\{J : J \in P \text{ and } p \in J\}|$.
- With respect to a partial solution P, the improvement of an interval $J \notin P$ is the number of positions with cover improvement by J. Formally, we define $improve(J, P) = |\{p : p \in J \text{ and } cover(p, P) < m\}|$.

Greedy Algorithm for m-Fold c-Interval Cover

Input: a genome sequence S and a family $\mathbb{F}$ of c-intervals $H_i(1 \leq i \leq |\mathbb{F}|)$ covering each letter of S at least M times.

Output: a subset P of $\mathbb{F}$ that covers each letter of S at least $m \leq M$ times.

Steps:

1 Let $P \leftarrow \emptyset$.
2 Repeat
3 Select a c-interval H such that for some interval $w(H) \in H$, $improve(w(H), P)$ is the largest.
4 $P \leftarrow P \cup \{H\}$.
5 Until each position in S is covered $\geq m$ times.
6 Return P.

End of Algorithm

Theorem 2: There is a polynomial time factor-$2c(4m-1)$ approximation algorithm for the m-fold c-interval cover problem.

Proof: Let O be an optimal solution for the m-fold c-interval cover problem such that each position of S is covered by at least m c-intervals in $\mathbb{F}$. When a new c-interval H is added to the approximate solution, we have an interval $w(H) \in H$ that makes the largest improvement given the current partial solution P.

When a new c-interval H is added to P, $w(H) \in H$ has a cover improvement at least at position q, which is covered at most $m-1$ times by the c-intervals in P (i.e., $cover(q, P) < m$) before H is added to P. WLOG, assume that there is some c-interval $I \in O$, $I \notin P$, such that I covers q. If $w(H)$ is completely contained in some interval of I, then it violates the greedy choice of the algorithm (as I should be selected instead of H — due to that $I \notin P$, $w(H)$ is contained in I, and I covers q). Let I contain c disjoint intervals $[l_1, r_1], \cdots, [l_c, r_c]$.

Consider a set of intervals $L_i(I)$ in P that covers the left endpoint l_i in the i-th interval $[l_i, r_i]$ in I. Let $H_{i,1}, H_{i,2}, \cdots$ be the c-intervals in $L_i(I)$ according to the order that are added to P. We need to bound the size of $L_i(I)$, $|L_i(I)|$.

Let $H_{i,1}, \cdots, H_{i,2m-1}$ be the first $2m-1$ c-intervals in $L_i(I)$. For any $H_{i,j}$ with $j > 2m-1$ in $L_i(I)$, it is impossible that $H_{i,j}$ can improve the elements to the left and right of l_i. Assume that $H_{i,j}$ improves an element p_L to the left of l_i, and also improves an element p_R to the right of l_i. p_L is covered by at most $m-1$ intervals in $H_{i,1}, \cdots, H_{i,2m-1}$, and p_R is covered by at most $m-1$ intervals among $H_{i,1}, \cdots, H_{i,2m-1}$. Thus, there is an interval among $H_{i,1}, \cdots, H_{i,2m-1}$ that is fully contained in $H_{i,j}$. This contradicts the greedy choice of the algorithm (i.e., if so, $H_{i,j}$ would have been selected earlier).

Let $H_{i,j_1}, H_{i,j_2}, \cdots, H_{i,j_m}$ be m intervals in $L_i(I)$ after the first $2m-1$ intervals are selected to improve the elements to the left of l_i. When we have $H_{i,j_{m+1}}$, it can improve an element p to the left of l_i, and p is covered by at most $m-1$ intervals in $\{H_{i,j_1}, H_{i,j_2}, \cdots, H_{i,j_m}\}$. Thus, at least one interval, say H_{i,j_t}, does not cover p. Therefore, $H_{i,j_{m+1}}$ can have a bigger improvement than H_{i,j_t}. This contradicts the greedy choice. Therefore, after the first $2m-1$ c-intervals $H_{i,1}, \cdots, H_{i,2m-1}$ are selected, there are at most m intervals in $L_i(I)$ to improve the elements to the left of l_i.

Similarly, we can deal with the case of right endpoints improvement for I. Therefore, $L_i(I)$ contains at most $2m-1+m+m = 4m-1$ intervals. The total approximation ratio is bounded by at most $2c(4m-1)$, as there are c intervals in I, each with two endpoints. ■

For the m-fold PE-interval cover problem, the approximation factor becomes $4(4m-1)$, as $c = 2$. It is still open whether some approximation algorithms can be designed for the m-fold c-interval cover problem, with a constant factor independent of m, c. We will report the performance of these two algorithms, for $m = 30$ and $c = 2$, in Section VI.

IV. Fixed Parameter Tractable Result for k-bounded Set-cover

In this section, we show a fixed parameter tractable solution for the k-bounded set cover problem.

Theorem 3: There is an $O(k^t n^{O(1)})$-time algorithm such that given a set X with size n and a family $\mathbb{F}$ of subsets of X, it finds a solution for the k-bounded set cover problem with at most t subsets.

Proof: In order to cover the whole set X, for each element in X that belongs to at most k subsets, we must select one of the k subsets in order to cover this element. The selection has at most k choices. Since a solution needs at most t subsets, the total time is bounded by $O(k^t n^{O(1)})$.

■

Definition 5: The *k-bounded c-interval cover problem* is a variant of the c-interval cover problem which satisfies that the number of occurrences of any elements in each $I_i \in \mathbb{F}$ is bounded by a constant $k \geq 2$ and there exists an element which appears in $\mathbb{F}$ exactly k times.

Since the k-bounded c-interval cover problem is a special case of the k-bounded set cover problem, we have the following corollary.

Corollary 1: Given a permutation S with length n and a set $\mathbb{F}$ of c-intervals, there is an $O(k^t n^{O(1)})$-time algorithm to find a solution for the k-bounded c-interval cover problem with at most t c-intervals.

V. Complexity of the c-Interval Cover Problem

In this section, we explore the hardness of the c-interval cover problem. Theorem 5 shows that the problem is NP-complete when $c = 2$ and by Theorem 4, the c-interval cover problem is APX-hard if $c \geq 3$.

It is well known that k-set cover is APX-hard for $k \geq 3$ [19]. Based on that, we have the following result.

Theorem 4: The c-interval cover problem is APX-hard if $c \geq 3$.

Proof: We have a simple polynomial time reduction from 3-set cover to the 3-interval cover problem by setting each interval to be a single element. The same idea can be used for $c > 3$. ■

In the following part, we show that the PE-interval cover problem is NP-complete. The core idea is that the $(3,3)$-SAT problem is polynomial-time reducible to the PE-interval cover problem. We first give the definitions of 3SAT and $(3,3)$-SAT.

Definition 6: A *3SAT instance* is a conjunctive form $C_1 \wedge C_2 \wedge \ldots \wedge C_m$ such that each C_i is a disjunction of at most three literals. *3SAT* is the language of those 3SAT instances that have satisfiable assignments.

Definition 7: A $(3,3)$-*SAT* instance is an instance G of $3SAT$ such that for each variable x, the total number of occurrences of x and $\bar{x}$ in G is at most 3, and the total number of occurrences of $\bar{x}$ in G is at most 1. $(3,3)$-*SAT* is the language of those $(3,3)$-SAT instances that have satisfiable assignments.

For examples, $(\bar{x}_1 \vee x_2 \vee x_3) \wedge (\bar{x}_1 \vee \bar{x}_2) \wedge (x_1 \vee \bar{x}_3)$ is a 3SAT instance but not a $(3,3)$-SAT instance since $\bar{x}_1$ appears twice. On the other hand, $(x_1 \vee x_2 \vee x_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_3) \wedge (x_1 \vee x_3)$ belongs to both 3SAT and $(3,3)$-SAT. It is well known that 3SAT is an NP-complete problem. The following lemma shows that 3SAT is polynomial-time reducible to $(3,3)$-SAT.

Lemma 1: [18] There is a polynomial time reduction $f(.)$ from 3SAT to $(3,3)$-SAT.

Definition 8: Given a $(3,3)$-SAT instance $\phi = C_1 \wedge C_2 \wedge \ldots \wedge C_m$, a clause C_i of ϕ is called a 3-*positive clause* if C_i contains three positive literals. If a $(3,3)$-SAT instance ϕ has no 3-positive clause, ϕ is called a $n3p$-$(3,3)$-*SAT instance*.

Lemma 2: There is a polynomial time transformation from a $(3,3)$-SAT instance ϕ to a $n3p$-$(3,3)$-SAT instance ϕ' such that ϕ is satisfiable iff ϕ' is satisfiable.

Proof: Let $\phi = C_1 \wedge C_2 \wedge \ldots \wedge C_m$ be an instance of $(3,3)$-SAT. The transformation from ϕ to ϕ' performs the same conversion for each clause $C_i (1 \leq i \leq m)$. If a clause $C_i = (x_1^i \vee x_2^i \vee x_3^i)$ of ϕ is a 3-positive clause, we convert C_i into $C_i' \wedge C_i'' = (x_1^i \vee x_2^i \vee \bar{x}_a^i) \wedge (x_3^i \vee x_a^i)$ by adding a new variable x_a^i. Neither C' nor C'' is a 3-positive clause, after the conversion. Therefore, a $(3,3)$-SAT instance can be transformed into a $n3p$-$(3,3)$-SAT instance in $O(m)$ time. It is easy to verify that C_i is true iff $C_i' \wedge C_i''$ is true, so ϕ is satisfiable iff ϕ' is satisfiable. ∎

For example, $\phi_1 = C_1 \wedge C_2 \wedge C_3$ is an instance of $(3,3)$-SAT, where $C_1 = (x_1 \vee \bar{x}_2)$, $C_2 = (\bar{x}_1 \vee x_2 \vee \bar{x}_3)$, $C_3 = (x_1 \vee x_2 \vee x_3)$. We convert it into an equivalent $n3p$-$(3,3)$-SAT instance $\phi_1' = C_1 \wedge C_2 \wedge C_3' \wedge C_4'$ by adding a (dummy) variable x_4. Here $C_3' = (x_1 \vee x_2 \vee \bar{x}_4)$ and $C_4' = (x_3 \vee x_4)$.

As a decision problem, for a given permutation S with length n, a family $\mathbb{F}$ of paired-end intervals and an integer number k, we ask simply whether there exists a subset $\mathbb{C} \subseteq \mathbb{F}$ such that the permutation formed by the subset $\mathbb{C}$ covers all the elements in S and $|\mathbb{C}| = k$.

Theorem 5: The decision version of the PE-interval cover problem is NP-complete.

Proof: To show that the PE-interval cover problem is in NP, for a given set $\mathbb{F}$ of paired-end intervals, we nondeterministically select a subset $\mathbb{C} \subseteq \mathbb{F}$ as a certificate for $\mathbb{F}$. Let S' be the set of elements in S covered by $\mathbb{C}$. Checking the certificate $\mathbb{C}$ can be accomplished in polynomial time in n ($|S| = n$) by checking, for each element $s \in S$, whether s belongs to S'.

We next prove that $(3,3)$-SAT $\leq_p$ PE-interval cover, which shows that the PE-interval cover problem is NP-hard. We are given a $(3,3)$-SAT instance $\phi = C_1 \wedge C_2 \wedge \ldots \wedge C_m = s(x_1, x_2, \ldots, x_k)$ where m is the number of clauses and k is the number of variables in ϕ. WLOG, we make three simplified assumptions about the formula ϕ. Firstly, no clause contains both a variable and its negation, for such a clause, it is automatically satisfied by any assignment of values to the variables. Secondly, each variable appears in at least one clause, for otherwise it does not matter what value is assigned to the variable. Thirdly, no variable appears three times in ϕ positively, otherwise we can simply assign these variables true to remove them and the clauses which contains these variables without affecting the satisfiability of ϕ.

First, we transform ϕ into a $n3p$-$(3,3)$-SAT instance $\phi' = C_1' \wedge C_2' \wedge \ldots \wedge C_{m'}' = s(x_1', x_2', \ldots, x_{k'}')$ using Lemma 2. Let m' and k' be the clause size and variable size of ϕ' respectively, we have $m \leq m' \leq 2m$ and $k \leq k' \leq k + m$. In our construction, S is initially composed of a sequence of non-empty segments/intervals, each is filled with a permutation of letters such that no two intervals share a common letter. Moreover, $|S| = n$ and there are $m' + k'$ intervals in S — we assume that n is a multiple of $m' + k'$. We then try to construct a set $\mathbb{F}$ which contains $2k'$ paired-end intervals, based on the intervals in S, such that ϕ is satisfiable iff there exists a subset $\mathbb{C} \subseteq \mathbb{F}$ such that the sequence formed by the subset $\mathbb{C}$ covers all the elements of S and $|\mathbb{C}| = k'$.

The set $\mathbb{F}$ can be constructed using the following 4 steps of procedures (see an example in Figure 1):

Step 1: Divide the sequence S into $m' + k'$ intervals $S_1, S_2, \ldots, S_{m'+k'}$ of the same size $\frac{n}{m'+k'}$. We have $S_1 = \langle 1, 2, \ldots, \frac{n}{m'+k'} \rangle$ and $S_i = \langle \frac{n(i-1)}{m'+k'}+1, \frac{n(i-1)}{m'+k'}+2, \ldots, \frac{ni}{m'+k'} \rangle$ for $2 \leq i \leq m' + k'$.

Step 2: Mark all the intervals and all the variables as unused, let $i = 1$.

Step 3: For clause $C_i' = (l_1^i \vee l_2^i \vee l_3^i)$ of ϕ', where $i \leq m'$, we have three cases:

- *Case 1*: If C_i' has exactly one unused positive variable, select the first unused interval S_u. Let the variables l_1^i, l_2^i and l_3^i represent the same interval S_u, then S_u is marked as used. We call the intervals which are marked by clauses as *Clause Intervals*. For that unused positive variable l_j^i $(j \in \{1,2,3\})$ in C_i', which means $l_j^i = x_t'$ $(1 \leq t \leq k')$, select the first unused interval S_v, let l_j^i and $\bar{l}_j^i$ represent the same interval S_v. Then the variable $l_j^i = x_t'$ and the interval S_v are marked as used. We call the intervals which are marked by variables as *Variable Intervals*.
- *Case 2*: If C_i' has two unused positive variables, select the second unused interval S_u, let variables l_1^i, l_2^i and l_3^i represent the same interval S_u. Then the interval S_u is marked as used. For those two positive variables $l_{j_1}^i$ $(j_1 \in \{1,2,3\})$ and $l_{j_2}^i$ $(j_2 \in \{1,2,3\})$ in C_i', select the first and the second unused intervals S_{v_1}, S_{v_2}. Let $l_{j_1}^i$ and $\bar{l}_{j_1}^i$ represent the same interval S_{v_1}. Similarly, let $l_{j_2}^i$ and $\bar{l}_{j_2}^i$ represent the same interval S_{v_2}. Then the variables $l_{j_1}^i$, $l_{j_2}^i$ and the intervals S_{v_1}, S_{v_2} are marked as used.
- *Case 3*: If there does not exist any unused positive variable in C_i', select the first unused interval S_u, let the variables l_1^i, l_2^i and l_3^i represent the same interval S_u. Then the interval S_u is marked as used.

Let $i = i + 1$, repeat step 3.

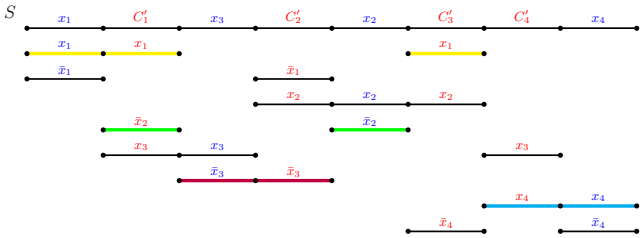

Fig. 1. The reduction from $n3p$-$(3,3)$-SAT instance $\phi'_1 = C'_1 \wedge C'_2 \wedge C'_3 \wedge C'_4$ to an instance of the PE-interval cover problem, where $C'_1 = (x_1 \vee \bar{x}_2 \vee x_3)$, $C'_2 = (\bar{x}_1 \vee x_2 \vee \bar{x}_3)$, $C'_3 = (x_1 \vee x_2 \vee \bar{x}_4)$, $C'_4 = (x_3 \vee x_4)$. Set S is divided into eight intervals. Clause Intervals and Variable Intervals are marked as red and blue respectively. A satisfying assignment of the formula has $x_1 = 1$, $x_3 = 0$, $x_4 = \bar{x}_3 = 1$, and x_2 may be either 0 or 1. This assignment satisfies C'_1 and C'_3 with x_1, it satisfies C'_4 with x_4 and satisfies C_2 with $\bar{x}_3$, corresponding to four paired-end intervals with four different colored bold lines.

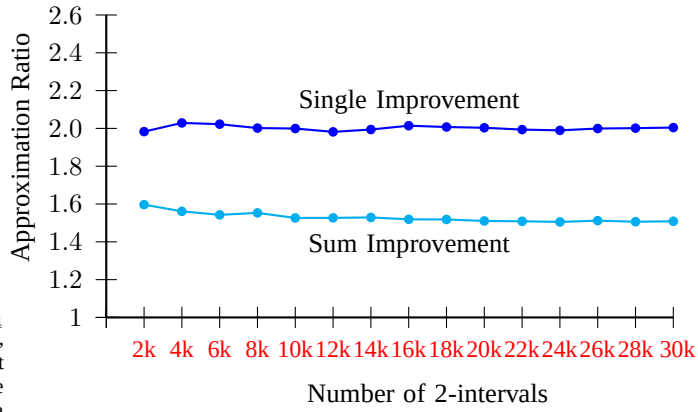

Fig. 2. The comparison between the single-improvement and sum-improvement greedy algorithms. The sequence length is $20,000$, the length of each interval is 100 and each element is covered at least once. The horizontal coordinates represent the number (in k=1000) of given 2-intervals, the vertical coordinates represent the average approximation ratio. The ratios for the two algorithms are shown in the blue curve and cyan curve respectively.

Step 4: If all the variables are marked as used, the construction is done. Otherwise there are unused variables. Then, for each unused variable x_u, select the first unused interval S_u. Let x_u and $\bar{x}_u$ represent the same interval S_u.

It is easy to see that the set $\mathbb{F}$ contains a paired-end interval for both x'_t and $\bar{x}'_t$ $(1 \leq t \leq k')$ in ϕ'. So the total number of paired-end intervals in $\mathbb{F}$ is $2k'$. In addition, the transformation takes $O(m' + k')$ time.

Finally, we must show that the transformation is a reduction. First, suppose that ϕ has a satisfying assignment. By Lemma 2, ϕ' also has a satisfying assignment. Then for each variable x'_t $(1 \leq t \leq k')$, x'_t is assigned either 1 or 0. If it is assigned 1, we select the paired-end interval represented by x'_t; otherwise, we select the paired-end interval represented by $\bar{x}'_t$. We put all the selected intervals into the set $\mathbb{C}$, it is obvious that $\mathbb{C}$ has k' elements. We next show that the union of all the intervals in $\mathbb{C}$ covers the sequence S. Since ϕ' is satisfiable, each clause C'_i contains at least one literal l^i_j which is assigned 1. So for each clause interval, it is covered. For each variable interval, because either x'_t or $\bar{x}'_t$ is selected, it is covered too. The sequence S is made up of the union of all the clause intervals and variable intervals, therefore S is covered by the paired-end intervals in $\mathbb{C}$.

Conversely, suppose that $\mathbb{F}$ is constructed using the above procedures, and $\mathbb{C} \subseteq \mathbb{F}$ is of size k' and it covers all the elements in S. We can assign 1 to all literals which are used to represent the k' paired-end intervals in $\mathbb{C}$. Here we do not need to worry about assigning 1 to both a literal and its complement. Since $\mathbb{C}$ is of size k', if the paired-end intervals represented by x'_t and $\bar{x}'_t$ are both included in $\mathbb{C}$, there must exist a variable x''_t such that neither the paired-end interval represented by x''_t nor by $\bar{x}''_t$ is included in $\mathbb{C}$. According to the construction of $\mathbb{F}$, the union of all intervals in $\mathbb{C}$ cannot cover all the elements of S. After the truth assignment, each clause is satisfied. Then ϕ' is satisfied, and therefore ϕ is satisfied.

■

We comment that this hardness result also holds if the reconstructed object is a sequence instead of a permutation.

VI. Experimental Results

We implement the greedy algorithms described in Section III for the PE-interval cover problem. The experimental results of the single improvement greedy algorithm show a much better approximation ratio than $6c = 12$, which was given in Theorem 1. The sum improvement greedy algorithm is also tested. The experimental results show that it has a general better performance compared with the first one. We also implement two algorithms for the m-fold PE-interval cover problem, focusing on $m = 30$ (and $c = 2$). This conforms more with the practical datasets in PE-based genome sequencing, simply because the cost of obtaining the short reads is cheap, hence it is possible to obtain a lot of short reads such that each element is covered at least $M \geq m$ times.

A. Implementation Details

With the purpose of easily testing the approximation performance, we select two short sequences of length $20,000$. The selections are based on the first sequenced RNA-genome [21] and DNA-genome [22]. The length of each short read is fixed at 100, i.e., each PE-interval contains two disjoint intervals of length 100 each. We use *diSize* to represent the number of given 2-intervals. We first construct an optimal solution which uses the least number of 2-intervals to cover the whole sequence (which is 20000/200=100) and add it into the 2-interval set, and then the remaining 2-intervals are constructed randomly.

B. Results for PE-interval Cover

For the single improvement greedy algorithm, the worst approximation ratio is 2.03. It is much better than the ratio 12 which was presented in Theorem 1. For the sum improvement greedy algorithm, the worst approximation ratio is 1.60. From their average approximation ratios, we discover that the second

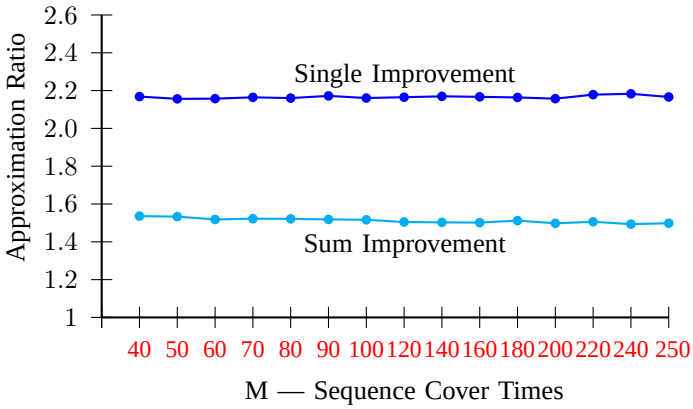

Fig. 3. The comparison between the greedy algorithms Single-1(m) and Sum-1(m) for the m-fold PE-interval cover problem. The sequence length is $20,000$, the length of each interval is set to 100. Both two greedy methods cover each element at least $m = 30$ times layer by layer. The horizontal coordinates represent the least number of times each element of the sequence is covered. The vertical coordinates represent the average approximation ratio. The ratios for the two algorithms are showed in the blue curve and cyan curve respectively.

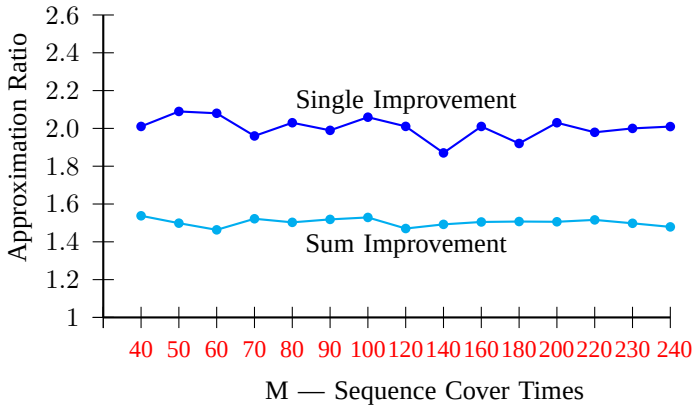

Fig. 4. The comparison between the greedy algorithms Single-2(m) and Sum-2(m) for the m-fold PE-interval cover problem. The sequence length is $20,000$, the length of each interval is set to 100. Both two greedy methods cover each element at least $m = 30$ times. The horizontal coordinates represent the least number of times each element of the sequence is covered. The vertical coordinates represent the approximation ratio. The ratios for the two algorithms are showed in the blue curve and cyan curve respectively.

algorithm has an overall better performance compared with the first one. Though we cannot prove that the second algorithm is a constant factor approximation, the testing results show that it works very effectively in practice. See Figure 2 for a comparison for the two greedy algorithms with their average approximation ratios.

We also found that the approximation ratios for these two greedy algorithms are not relevant to the number of given 2-intervals. This is possibly due to that the intervals are of the same length (which is a common practice in PE-based genome sequencing).

C. Results for m-fold PE-interval Cover

For the m-fold PE-interval cover problem, we implement two versions of algorithms, which are both based on the single-improvement and sum-improvement algorithms for PE-interval cover. The first version, Single-1(m) (resp. Sum-1(m)), is to run the single-improvement (resp. sum-improvement) greedy algorithm m times, i.e., level by level, each level corresponds to a solution for the PE-interval cover problem. We use the number of times an element is leastly covered, M, as another parameter. In the whole process, we fix m=30. See Figure 3 for a comparison for the two greedy algorithms with their average approximation ratios.

The second version, Single-2(m) (resp. Sum-2(m)), is the factor-$2c(4m-1)$ approximation algorithm; i.e., we select a PE-interval I_i which contains the maximum number of elements which have not been covered m times yet in one (resp. both) of the intervals in I. See Figure 4 for a comparison for the two greedy algorithms with their average approximation ratios. While the approximate ratios are similar for the two versions of algorithms, the second version shows a much larger fluctuation. On the other hand, it is much better than the theoretical ratio which is 478 (in the worst case).

VII. Conclusion

In this paper, we investigate the problem of identifying the minimum number of short reads to cover the genes in a genome (to be sequenced) m times. For $m = 1$, we formulate this as the PE-interval cover problem which is related to the famous set-cover problem. We explore the hardness of the PE-interval cover problem and present some approximation algorithm to solve it. The approximation algorithm also generalizes to the case for fixed m, c and is shown to have a factor of $2c(4m-1)$. There are several open problems. (1) For the m-fold c-interval cover problem, is there a constant factor approximation whose factor is independent of m, c? (2) When $m = 1$, the sum improvement greedy algorithm has a better performance in our implementation; however, there is no approximation analysis of it. (3) It seems that the approximation ratio given in Section III can be improved, at least indicated by our empirical results.

Acknowledgments

Bin Fu is supported in part by NSF Early Career Award 0845376. Binhai Zhu is partially supported by NSF under project DMS-0918034, by NSF of China under project 60928006, by the Shanghai Thousand Talents Program, and by the Open Fund of Top Key Discipline of Computer Software and Theory in Zhejiang Provincial Colleges at Zhejiang Normal University. We also thank anonymous reviewers for several constructive comments.

References

[1] Alon N., Moshkovitz D., and Safra S. Algorithmic construction of sets for k-restrictions. *ACM Trans. Algorithms.*, 2(2):153-177, 2006.
[2] Johnson D.S. Approximation algorithms for combinatorial problems. *J. Comput. System Sci.*, 9:256-278, 1974.
[3] Chvatal V. A greedy heuristic for the set-covering problem. *Mathematics of Operations Research*, 4:233-235, 1979.

[4] Feige U. A threshold of ln n for approximating set cover. *J. ACM*, 45(4):634-652, 1998.

[5] Bar-Yehuda R. and Kehat Z. Approximating the dense set-cover problem. *J.Comput. and System Sci.*, 69:547-561, 2004.

[6] Raz R. and Safra S. A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP. In *STOC '97: Proceedings of the twenty-ninth annual ACM Symposium on Theory of Computing*, pp.475-484, 1997.

[7] Edwards A. and Caskey T. Closure strategies for random DNA sequencing. *Methods Comp. Methods Enzymol*, 3(1):41-47, 1991.

[8] Chen E.Y., Schlessinger D. and Kere J. Ordered shotgun sequencing, as strategy for integrating mapping and sequencing of YAC clones. *Genomics*, 17:651-656, 1993.

[9] Smith M.W., Holmsen A., Wei Y.H., Peterson M. and Evans G.A. Genomic sequencing sampling: A strategy for high resolution sequence-based physical mapping of complex genomes. *Nat. Genet.*, 7:40-47, 1994.

[10] Venter J.C., Adams M.D., Sutton G.G., Kerlavage A.R., Smith H.O. and Hunkapiller M. Shotgun sequencing for the human genome. *Science*, 280:1540-1542, 1998.

[11] Venter J.C., Smith H.O. and Hood L. A new strategy for genome sequencing. *Nature*, 381:364-366, 1996.

[12] Weber J.L. and Myers E.W. Human whole-genome shotgun sequencing. *Genome Res.*, 7:401-409, 1997.

[13] Siegel A. F., Engh G. van den, Hood L., Trask B. and Roach J. C. Analysis of sequence-tagged connector (STC) strategies for DNA sequencing. *Genomics*, 68(3):237-246, 2000.

[14] Eward A., Voss H., Rice P., Civitello A., Stegemann J., Schwager C., Zimmerman J., Erfle H., Caskey T. and Ansorge W. Automated DNA sequencing of the human HPRT locus. *Genomics*, 6:593-608, 1990.

[15] Fleischmann R.D., Adams M.D., White O., Clayton R.A., Kirkness E.F., Kerlavage A.R., Bult C.J., Tomb J.F., Dougherty B.A., Merrick J.M., et al. Whole-genome random sequencing and assembly of Haemophilus influenzae Rd. *Science*, 269(5223):496-512, 1995.

[16] Fraser C.M., Gocayne J.D., White O., Adams M.D., Clayton R.A., Fleischmann R.D., Bult C.J., Kerlavage A.R., Sutton G., Kelley J.M., et al. The minimal gene complement of Mycoplasma genitalium. *Science*, 270(5235):397-404, 1995.

[17] Anson E. and Myers E.W. Algorithms for whole genome shotgun sequencing. In *Proceedings of the third annual International Conference on Research in Computational Molecular Biology (RECOMB'99)*, pp.1-9, 1999.

[18] Tovery C.A. A simplified satisfiability problem. *Discrete Applied Mathematics*, 8:85-89, 1984.

[19] Kann V. Maximum bounded 3-dimensional matching is MAX SNP-complete. *Information Processing Letters*, 37:27-35, 1991.

[20] Papadimitriou C. and Yannakakis M. Optimization, approximation and complexity classes. *J. Comput. and System Sci.*, 43:425-440, 1991.

[21] Fiers W., Contreras R., Duerinck F., Haegeman G., Iserentant D., Merregaert J., Min Jou W., Molemans F., Raeymaekers A., Van Den Berghe A., Volckaert G., and Ysebaert M. Complete nucleotide-sequence of bacteriophage MS2-RNA - primary and secondary structure of replicate gene. *Nature*, 260(5551):500-507, 1976.

[22] Sanger F., Air G.M., Barrell B.G., Brown N.L., Coulson A.R., Fiddes C.A., Hutchison C.A., Slocombe P.M., and Smith M. Nucleotide sequence of bacteriophage phi X174 DNA. *Nature*, 265(5596):687-695, 1977.

[23] Downey R. and Fellows M. *Parameterized Complexity*, Springer-Verlag. 1999.

[24] Huson D., Reinert K., and Myers E.W. The greedy path-merging algorithm for contig scaffolding. *J. ACM*, 49(5):603-615, 2002.

[25] Gao S., Nagarajan N., and Sung W-K. Opera: Reconstructing optimal genomic scaffolds with high-throughput paired-end sequences. In *Proceedings of the fifteenth annual International Conference on Research in Computational Molecular Biology (RECOMB'11)*, pp.437-451, 2011.

Liang Ding Liang Ding is currently a PhD student in computer science at the University of Georgia. He obtained a MS degree in Computer Science at University of Texas - Pan American in 2011. His research interests are computational biology and bioinformatics.

Bin Fu Bin Fu is currently an associate professor in computer science at University of Texas - Pan American. He obtained his PhD in Computer Science from Yale University in 1998. He won the NSF Career Award in 2009. His research interests are complexity theory, algorithms, computational biology and bioinformatics. He has published over 100 papers in these areas.

Binhai Zhu Binhai Zhu is currently a professor in computer science at Montana State University, USA. He obtained his Ph.D. in Computer Science from McGill University, Canada, in 1994. He was a post-doctoral research associate at Los Alamos National Laboratory, USA from 1994 to 1996. From 1996 to 2000, he was an assistant professor at City University of Hong Kong. He has been at Montana State University since 2000 (associate professor until 2006, professor since 2006). Professor Zhu's research interests are geometric computing, biological/geometric modeling, bioinformatics and combinatorial optimization. He has published over 120 papers in these areas.