Approaching the One-sided Exemplar Adjacency Number Problem

Letu Qingge
College of Computing and Informatics
University of North Carolina
Charlotte, NC 28223, USA
Email: letu.qingge@uncc.edu

Killian Smith
Gianforte School of Computing
Montana State University
Bozeman, MT 59717, USA
Email: killian.smith@msu.montana.edu

Sean Jungst
Gianforte School of Computing
Montana State University
Bozeman, MT 59717, USA
Email: sean.jungst@msu.montana.edu

Baihui Wang
Martin J. Whitman School of Management
Syracuse University
Syracuse, NY 13244, USA
Email: bwang04@syr.edu

Qing Yang
Dept. of Computer Science and Engineering
University of North Texas
Denton, TX 76207
USA
Email: qing.yang@unt.edu

Binhai Zhu
Corresponding author
Gianforte School of Computing
Montana State University
Bozeman, MT 59717, USA
Email: bhz@montana.edu

***Abstract*—The one-sided Exemplar Adjacency Number (EAN) is a known problem for computing the exemplar similarity between a generic linear genome $\mathcal{G}$ with gene duplications and an exemplar genome H (over the same set of n gene families). In this problem, we need to compute an exemplar genome G, which is a permutation obtained from $\mathcal{G}$, such that the number of common adjacencies between G and H is maximized. Unfortunately, the problem is not only NP-hard but also NP-hard to approximate. In this paper, we approach the problem by relaxing the constraint such that a sub-permutation G^+ obtained from $\mathcal{G}$ does not have to include all the gene families, but still needs to have a length at least k. Hence G^+ is called a *pseudo-exemplar* genome. Then, a slightly more general problem (One-sided EAN+) is defined: compute a pseudo-exemplar genome G^+ from $\mathcal{G}$ such that the number of common adjacencies between H and G^+ is maximized. Certainly One-sided EAN+ contains One-sided EAN as a special case; moreover, it presents some flexibility in designing algorithms. Firstly, we relax and formulate the One-sided EAN+ problem as the maximum independent set (MIS) on a colored interval graph and hence reduce the appearance of each gene to at most two times. We show that this new relaxation is still NP-complete, though a simple factor-2 approximation algorithm can be designed; moreover, we also prove that the problem cannot be approximated within $2-\varepsilon$ by a local search technique. We then show that this relaxed version is fixed-parameter tractable (FPT). Secondly, to ensure that each gene appears in G^+ at most once, we use integer linear programming (ILP) to solve this problem. Finally, we implement our algorithm and compare it with the up-to-date software GREDU, with simulated signed and unsigned genomes. It turns out that our algorithm is more stable and can process genomes of length up to 12,000 for signed genomes (while GREDU can falter on such a large signed genome and it cannot handle unsigned genomes at all).**

Keywords: genome comparison, exemplar adjacency number, maximum independent set, colored interval graph, integer linear programming.

I. Introduction

Computing the distance/similarity of two genomes is essential for building the evolution history of a genome. In the past two decades, there have been three general ways to compute the genomic distance or similarity between general genomes (with gene duplications): (1) computing the exemplar genomic distance (i.e., deleting all but one gene for each gene family) [20], [4], [5], [7], [2], [3], [16], (2) computing the number of common adjacencies between them [2], [14], [15], [17], [18], [22], [13], and (3) computing the minimum common partition number [9]. As our method somehow falls into (1), we review the corresponding methods in more detail.

In 1999, Sankoff formulated the exemplar breakpoint distance problem as follows: select exactly one gene from each gene family from two generic genomes $\mathcal{G}$ and $\mathcal{H}$ over the same set of gene families such that the breakpoint distance between the two resulting exemplar genomes G and H is minimized [20]. From a theoretical point of view, computing the exemplar breakpoint distance is not only NP-hard [4], but also NP-hard to approximate (regardless of the approximation factor) when each gene appears in $\mathcal{G}$ and $\mathcal{H}$ at most three times [5] or twice [3], [16]. This was shown by proving that deciding whether the optimal exemplar breakpoint distance is zero, i.e, whether $G = H$, is NP-complete. Hence, computing any approximation solution is NP-hard.

Nonetheless, several algorithms have been proposed for the problem. Sankoff first proposed a branch-and-bound approach for the exemplar breakpoint distance problem [20]. Nguyen *et al.* developed a more efficient divide-and-conquer algorithm for the problem [19]. Angibaud *et al.* presented an integer linear programming (ILP) method for computing the exact breakpoint distance [1]. More recently, Shao and Moret gave a slightly relaxed formulation for the exemplar breakpoint distance problem where not all gene families need to show up in the final (reduced) genomes, and they then implemented a fast and exact algorithm (named GREDU) using ILP [21].

The exemplar adjacency number problem is simply the complement of the exemplar breakpoint distance problem [6], [8]. Formally, the Exemplar Adjacency Number (EAN) problem can be defined as follows: given two generic linear genomes $\mathcal{G}$ and $\mathcal{H}$ over a set of n gene families, delete duplicated genes to obtain two exemplar genomes G and H of length n, such that the number of (common) adjacencies between G and H is maximized. The One-Sided EAN problem is the special case when $\mathcal{H}(=H)$ is given exemplar.

For the EAN problem, Chen *et al.* first showed that if $\mathcal{H}(=H)$ is exemplar and the other genome $\mathcal{G}$ is 2-repetitive (i.e., a gene from each of the n gene families appears at most twice in $\mathcal{G}$), then this version of One-sided EAN admits neither a polynomial-time factor-$n^{0.5-\varepsilon}$ approximation unless $P = NP$, nor an FPT algorithm unless FPT=W[1] [6], [8]. (FPT stands for *fixed-parameter tractable*, the formal definition is given in Section 4.) A factor-$n^{0.5}$ approximation algorithm was presented for the EAN problem when each gene appears at most twice in both $\mathcal{G}$ and $\mathcal{H}$ [8]. Intuitively, due to the hardness result of the EAN problem (especially that a gene in each gene family must appear in G), it is not convenient to design efficient algorithms.

In this paper, we investigate the One-sided EAN problem whose input is $\mathcal{G}$ and H, over a set of n gene families. We first relax the problem by deleting genes from $\mathcal{G}$ such that the reduced (pseudo-exemplar) genome G^+ is a sub-permutation of length at most n, and the objective is still to maximize the number of (common) adjacencies between G^+ and H. We call this problem One-sided EAN+, which is a generalized version of One-sided EAN and is at least as hard as the latter. (Note that the two problems are not exactly the same. For example, $\mathcal{G} = 315264$ and $H = 123456$, for the latter problem there is no common adjacency while in the former we could still delete 2 to obtain a common adjacency 56.)

On the other hand, One-sided EAN+ does give us some freedom in designing algorithms. We first obtain a new relaxation of the problem as the maximum independent set (MIS) on a colored interval graph — a new variant of interval graph for linear genomes where the intervals corresponding to the potential adjacency ab or $-b-a$ in $\mathcal{G}$ are given the same color. While it is well-known that MIS on interval graphs is polynomially solvable [12], we show in this paper that the MIS problem on colored interval graphs becomes NP-hard. On the other hand, a simple factor-2 approximation, a twist of Gavril's greedy algorithm [12], can be obtained. We show, a bit surprisingly, that a simple local search cannot further improve the 2 factor; moreover, we also show that this MIS problem on colored interval graphs is fixed-parameter tractable (FPT).

The above MIS approximation only gives us a reduced/relaxed instance for the One-sided EAN+ problem. Let $\mathcal{G}'$ be the reduced instance for $\mathcal{G}$, obtained by the approximate MIS solution. In $\mathcal{G}'$, each gene appears at most twice in $\mathcal{G}'$ (and some gene family might not appear in $\mathcal{G}'$ at all). The next step is to use ILP to compute the number of exemplar (common) adjacencies between the pseudo-exemplar genome G^+ (obtained from further reducing $\mathcal{G}'$) and H. (We comment that H is never altered, and this is different from the method by Shao and Moret [21].)

We implemented the algorithm and tested it on simulated genomes, generated in very much the same way as in [21]. (We used both signed and unsigned genomes, though in [21] only signed ones were used.) The comparison with GREDU [21] indicates that our algorithm is more stable and generates comparable number of adjacencies (though the definitions of the adjacencies differ a little, as discussed above). The details are in Section 4.

This paper is organized as follows. In section 2, we make necessary definitions. In section 3, we approach the One-sided EAN+ problem by first reformulating it as the maximum independent set on colored interval graphs, and then proving its NP-hardness. In section 4, we first design a simple 2-approximation algorithm (to obtain a reduced instance). In addition, we show that the problem is FPT (though the high running time makes it impractical). Finally we use ILP to solve the reduced instance exactly. In section 5, we present our empirical results in comparison with GREDU on simulated genomes (with enough duplicated genes). In section 6, we conclude the paper and point out some directions for future work.

II. Preliminaries

As genomes could be signed or unsigned, in this section we focus on the former (the latter is clear form the context, which will be explained in the next section). Given a set $\mathcal{F}$ of n gene families (alphabet), a (signed) genome $\mathcal{G}$ is a sequence of elements of $\mathcal{F}$ such that each element is with a sign (+ or -). Given a (signed) genome with a gene in each family appearing exactly once (which is called *exemplar*) $G = g_1 g_2 \cdots g_n$, we say that the gene g_i immediately precedes g_j, if $j = i+1$. Given two exemplar genomes G,H, if gene a immediately precedes b in G and neither a immediately precedes b nor $-b$ immediately precedes $-a$ in H, then they constitute a *breakpoint* in G. Then the breakpoint distance is simply the number of breakpoints in G or H, denoted as $bd(G, H)$. Similarly, given exemplar genomes G,H, if gene a immediately precedes b in G and either a immediately precedes b or $-b$ immediately precedes $-a$ in H, then they constitute a (common) *adjacency* in G. The adjacency number is the number of (common) adjacencies in G or H, denoted as $an(G, H)$. For exemplar genomes G,H, we have the equation $bd(G, H) + an(G, H) = n - 1$. For example, let $G = 12345, H = -5 - 4312$, then there are two adjacencies and two breakpoints between G and H.

Given two generic genomes (i.e., with gene duplications) $\mathcal{G}$ and $\mathcal{H}$ the exemplar breakpoint distance between them, denoted as $ebd(\mathcal{G},\mathcal{H})$, is the minimum breakpoint distance $bd(G,H)$ among all exemplar genomes G,H obtained from $\mathcal{G}$ and $\mathcal{H}$. Similarly, the exemplar adjacency number between $\mathcal{G}$ and $\mathcal{H}$, denoted as $ean(\mathcal{G},\mathcal{H})$, is the maximum adjacency number $an(G,H)$ among all exemplar genomes G,H obtained from $\mathcal{G}$ and $\mathcal{H}$. Again, we have the equation $ebd(\mathcal{G},\mathcal{H}) + ean(\mathcal{G},\mathcal{H}) = n-1$.

The problem of computing $ean(\mathcal{G},\mathcal{H})$ is formally defined as the *Exemplar Adjacency Number* (*EAN*) problem. When one of $\mathcal{G},\mathcal{H}$ is given exemplar, the corresponding problem is called *One-sided EAN* problem. Throughout this paper, we assume that $\mathcal{H}$ is given exemplar and we use H instead of $\mathcal{H}$ henceforth. A genome G^+ with length at least k ($k \leq n$), obtained from $\mathcal{G}$ by deleting duplicated genes (but each gene appears at most once in G^+), is called a *pseudo-exemplar* genome. (Note that G^+ is really a sub-permutation and the definitions of breakpoints and adjacencies between pseudo-exemplar genomes are the same as for exemplar genomes, except that they do not necessarily sum to $n-1$ anymore.) Given k, the problem of computing G^+ from $\mathcal{G}$ such that the number of adjacencies between G^+ and H is maximized, is hence called the *One-sided EAN+* problem. (Note that One-sided EAN+ contains One-sided EAN as a special case when $k = n$; hence it is at least as hard as the latter.)

Finally, an interval graph $\mathcal{I} = (V,E)$ is a graph whose vertices have an one-to-one correspondence to a set of intervals on a line. An interval $u = (l(u), r(u))$ is represented by its left endpoint $l(u)$ and the right endpoint $r(u)$. There is an edge between two vertices $u, v \in V$ iff the intervals have a non-empty intersection. See Fig. 1 for an example.

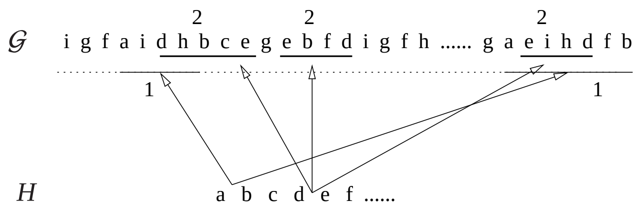
In the next section, we present a new method for the One-sided EAN+ problem. We start with a different relaxation/formulation of the problem.

III. A New Formulation of the Problem

As discussed earlier, the input for us is a generic linear genome $\mathcal{G}$ and an exemplar linear genome H, over the same set of gene families. First of all, we try to identify some disjoint intervals in $\mathcal{G}$, one for each color (each corresponding to a unique 2-substring in H). These intervals are the vertices of the corresponding colored interval graph $\mathcal{I}$. Then we try to identify the maximum number of disjoint intervals in $\mathcal{G}$, each of a different color. Clearly these intervals form a maximum (colored) independent set in $\mathcal{I}$. We show how these intervals are constructed as follows.

For each 2-substring $a_i a_{i+1}$ in H, we list all the minimal intervals $a_i\beta a_{i+1}$ or $-a_{i+1}\beta - a_i$ (for unsigned genomes $a_{i+1}\beta a_i$) in $\mathcal{G}$ such that the contents β could be deleted to have a potential adjacency $a_i a_{i+1}$ or $-a_{i+1} - a_i$ ($a_{i+1}a_i$ for unsigned genomes). Note that a substring $x\beta y$ is *minimal* if the substring β does not contain x, or y, or a subsequence (potential adjacency) $-y-x$. All the minimal intervals in $\mathcal{G}$ corresponding to $a_i a_{i+1}$ in H are given the same color. See Fig. 1 for an example of a colored interval graph where an unsigned genome is used.

We define the Maximum Independent Set (MIS) problem in a Colored Interval Graph (MIS-CIG for short) as follows.



$\mathcal{G}$
i g f a i d h b c e g e b f d i g f h ...... g a e i h d f b
2
2
2
1
1
H
a b c d e f ......

Fig. 1. Formulation of the one-sided EAN problem potentially as MIS in a colored interval graph. Note that in $\mathcal{G}$ the two intervals labeled with color 1 correspond to the adjacency *ab*, while the three intervals labeled with color 2 correspond to the adjacency *de*.

Problem: MIS-CIG
Instance: A set $\mathcal{I}$ of m intervals on a line, each is colored by one of the k ($k < m$) colors, and a parameter $k_1 \leq k$.
Question: Are there k_1 disjoint intervals of different colors?
We show next that MIS-CIG is NP-complete.

A. Hardness of MIS-CIG

Theorem 1: The MIS-CIG problem is NP-complete.

Proof: It is easy to see that MIS-CIG is in NP. To prove its NP-completeness, we reduce the classic 3SAT problem to MIS-CIG. Given a Boolean formula ϕ in 3-conjunctive normal form, $\phi = F_1 \wedge F_2 \wedge \cdots \wedge F_\ell$, where each of the ℓ clauses F_i has three distinct literals from a set of m boolean variables $x_1, x_2, \cdots, x_m$ and their negations, the problem is to decide whether ϕ is satisfiable.

For each literal (variable x_i and its negation $\overline{x_i}$), we construct two copies of interleaving intervals of four different colors $4i-3, 4i-2, 4i-1, 4i$. (See Figure 2.) If x_i is assigned TRUE, we select $4i-3, 4i-1$ at the top-right corner and $4i-2, 4i$ at the bottom-left corner. If x_i is assigned FALSE, we select $4i-3, 4i-1$ at the top-left corner and $4i-2, 4i$ at the bottom-right corner. For each clause F_j, we create three very small intervals of the same color (F_j). We put the interval with color F_j under the interval $4i-3$ of x_i if x_i appears in F_j, and we put the interval with color F_j under the interval $4i-3$ of $\overline{x_i}$ if $\overline{x_j}$ appears in F_j. We give an example to illustrate the construction. Assume we have a 3SAT formula $\phi = F_1 \wedge F_2 \wedge F_3 \wedge F_4$, with $F_1 = (x_1 \vee \overline{x_2} \vee x_3), F_2 = (\overline{x_1} \vee x_2 \vee \overline{x_4}), F_3 = (\overline{x_2} \vee \overline{x_3} \vee x_4)$, and $F_4 = (x_1 \vee \overline{x_3} \vee \overline{x_4})$. In Fig. 2, we show the corresponding construction for ϕ.

We prove next that ϕ is satisfiable if and only if the colored interval graph has an IS of $4m + \ell$ intervals (all with different colors).

"*Only if part*:" Suppose that the 3SAT instance ϕ is satisfiable. If x_i is assigned TRUE, we select the top-right intervals $4i-3, 4i-1$, the bottom-left intervals $4i-2, 4i$ and all other F_j intervals in the bottom row under the unselected interval $4i-3$ as part of the MIS. (If multiple F_j intervals are selected, we just arbitrarily keep one of them.) If x_i is assigned FALSE, we select the top-left intervals $4i-3, 4i-1$, the bottom-right intervals $4i-2, 4i$ and all other F_j intervals in the bottom row under the unselected interval $4i-3$ as part of the MIS. Consequently, we obtain an MIS with $4m+\ell$ intervals (all with different colors).

"*If part*:" Suppose that the colored interval graph contains an independent set of $4m + \ell$ intervals, all with different colors. As we have exactly $4m + \ell$ different colors, all colors must appear in the independent set. For the two groups of interleaving intervals $4i-3, 4i-2, 4i-1$ and $4i$ (corresponding to x_i and $\overline{x_i}$ respectively), clearly we could only select them in two different ways. We could either select the top-right $4i-3, 4i-1$ and the bottom-left $4i-2, 4i$ intervals, or we could select the top-left $4i-3, 4i-1$ and the bottom-right $4i-2, 4i$ intervals. In the former case, we assign x_i=TRUE, and then the F_j intervals containing x_i will be in the independent set as the top-left interval $4i-3$ (which intersects all the F_j intervals containing x_i) is not selected. In this case, F_j evaluates to TRUE and is satisfied. Similarly, we could show F_j containing $\overline{x_i}$ is also satisfied if we select the top-left $4i-3, 4i-1$ and the bottom-right $4i-2, 4i$ intervals, which corresponds to assigning x_i=FALSE. As all intervals of color F_j must appear in the independent set once (hence all clauses F_j's are satisfied in ϕ), we have a truth assignment for all the variables to satisfy ϕ.

The reduction obviously takes $O(m + \ell)$ time. Hence the theorem is proven. ■

In Fig. 3, we show the selected intervals for x_1=TRUE, x_2=FALSE, x_3=TRUE, and x_4=FALSE.

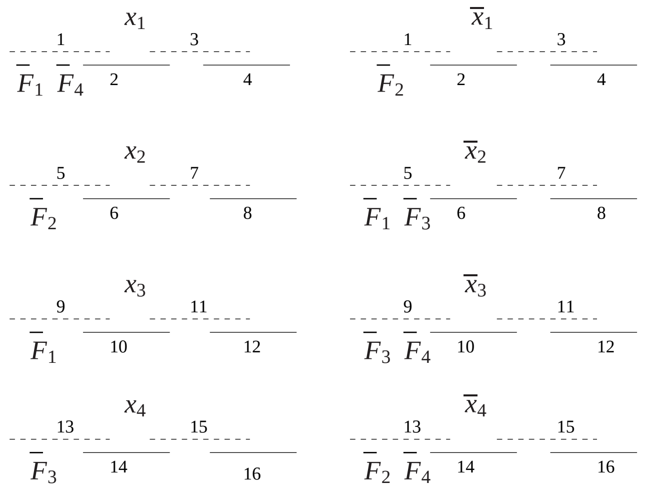


Fig. 2. Illustration for the reduction from the 3SAT instance $\phi = F_1 \wedge F_2 \wedge F_3 \wedge F_4$, where $F_1 = (x_1 \vee \overline{x_2} \vee x_3), F_2 = (\overline{x_1} \vee x_2 \vee \overline{x_4}), F_3 = (\overline{x_2} \vee \overline{x_3} \vee x_4)$, and $F_4 = (x_1 \vee \overline{x_3} \vee \overline{x_4})$.

IV. Algorithms

In this section, we design a factor-2 approximation algorithm for the MIS-CIG problem. (Moreover, we prove that the approximation factor cannot be improved to be less than 2 by some simple local search technique.) This gives us a reduced instance $\mathcal{G}'$ where each gene appears at most twice in it. We then show that MIS-CIG is FPT, though the results is only of theoretical meaning. Finally, we use ILP to efficiently delete extra gene duplications in $\mathcal{G}'$ while computing the maximum number of (common) adjacencies.

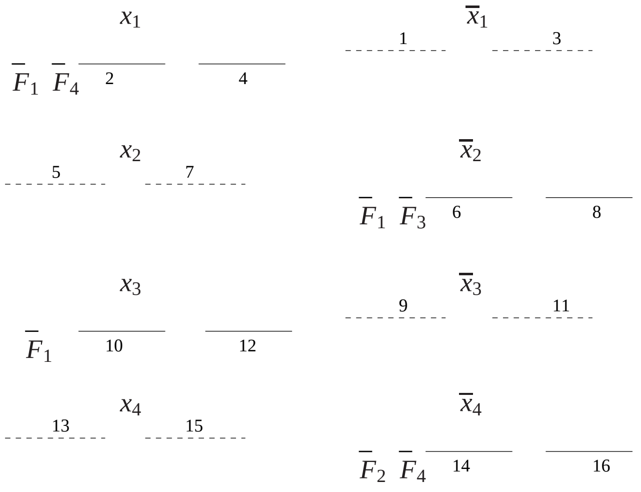


Fig. 3. The colored intervals selected based on the truth assignment $x_1 = x_3 =$ TRUE, $x_2 = x_4 =$ FALSE. Note that for intervals labelled with F_1 and F_4 we can select one arbitrarily.

A. A factor 2-approximation

A factor-2 approximation for MIS-CIG can be obtained using a twist of the well-known greedy method. Firstly, the intervals are sorted according to their right endpoints (from left to right). Secondly, we scan the intervals from left to right, according to the sorted order, and put the first interval I_1 in the solution. Then we delete all the intervals of the same color with I_1 or intersecting I_1, and repeat this process until no more interval can be put in the solution.

Algorithm 1 Greedy algorithm

```
1)  $m \leftarrow$ size of $\mathcal{I}$.
2)  $S \leftarrow \emptyset$.
3)  Sort all intervals in $\mathcal{I}$ according to their right endpoints
    as $I_1, ..., I_m$.
4)  Delete from $\mathcal{I}$ all the intervals of the same color with
    $I_1$ or intersecting $I_1$.
5)  Delete $I_1$ from $\mathcal{I}$.
6)  $S \leftarrow S \cup \{I_1\}$.
7)  while $\mathcal{I} \neq \emptyset$
8)      $I_t \leftarrow$ the first interval in $\mathcal{I}$.
9)      $S \leftarrow S \cup \{I_t\}$.
10)     Delete all the intervals of the same color with $I_t$ or
        intersecting $I_t$.
11)     Delete $I_t$ from $\mathcal{I}$.
12) Return $S$.
```

It is straightforward to see that the algorithm returns a set S of intervals. We show below that S is a factor-2 approximation for MIS-CIG. Let $I_i \in S$ and let I_j be some interval which intersects $r(I_i)$ (the right endpoint of I_i). As the algorithm scans intervals from left to right, any optimal solution not containing I_i must either contain an interval of the same color with I_i, or contain an interval of the same color with I_j (inclusive of I_j), or both. Hence the approximation algorithm would return at least a half of the optimal solution associated with $r(I_i)$. Applying this argument inductively, we could conclude that S is a factor-2 approximation for MIS-CIG.

B. A local search improvement?

With the greedy algorithm, it is natural to try to improve S using local search. Let $\mathcal{I}$ be the input intervals for the MIS-CIG problem. We search for a subset of c intervals in S, S_c, such that putting S_c back to $\mathcal{I} \backslash S$ enables us to find locally a subset S'_c of more than c independent intervals with different colors and all the colors in S'_c do not appear in $S \backslash S_c$. Then $(S \backslash S_c) \cup S'_c$ would give us a better solution than S and we update $S \leftarrow (S \backslash S_c) \cup S'_c$. This process will continue until no S_c can be found. For a constant c, we call the corresponding local search procedure *c-local search*. Unfortunately, for $c = 1, 2$, this local search method cannot improve the result in the worst case. We summarize the result as follows.

Theorem 2: Algorithm 1 returns a 2-approximation for MIS-CIG. For $c = 1, 2$ and some constant ε, the solution obtained by Algorithm 1 cannot be improved to have an approximation factor smaller than $2 - \varepsilon$ using c-local search.

Proof: We only show the second part, as the first part has just been proved. Assume to the contrary that with c-local search, $c = 1, 2$, we could obtain a $2 - \varepsilon$ approximation for the MIS-CIG problem. We construct an instance with $3n + 2$ intervals, with $2n$ colors $\{0, 1, ..., 2n - 1\}$. The greedy algorithm would select the intervals at the bottom row with colors $0, 1, 2, ..., n$. The optimal solution is to select all the intervals in the top row (except the last interval with color 0), with colors $0, 1, ..., 2n - 2, 2n - 1$. When context is clear, we just call an interval with color i interval i. See Fig. 4.

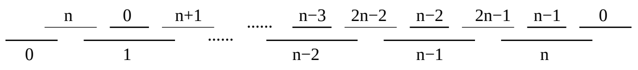


Fig. 4. Illustration for the proof of Theorem 2.

To see that c-local search does not incur any improvement for $c = 1, 2$, it is important to notice that any interval i ($i > 0$) at the bottom row contains an interval $i - 1$ at the top row; moreover, the two neighboring intervals of interval $i - 1$ at the top row, $n + i - 1$ and $(n + i) \mod 2n$, are both intersecting with the corresponding neighboring intervals of i at the bottom row. (For interval 0 at the bottom row, it intersects interval n at the top row.) Hence, for 1-local search, it is impossible to obtain a larger independent set by swapping interval i at the bottom row with two intervals in new colors at the top row. Similarly, for 2-local search, it is impossible to obtain a larger independent set by swapping two intervals i, j at the bottom row with three intervals in new colors at the top row.

For this instance, the approximation factor of the greey method is

$$\frac{2n}{n+1} = 2 - \frac{2}{n+1} > 2 - \varepsilon,$$

when $n > \frac{2-\varepsilon}{\varepsilon}$ (i.e., when n is sufficiently large). Hence, the approximation factor of the greedy algorithm is tight, whether or not c-local search, $c = 1, 2$, are applied. ■

Note that when $c \geq 3$, the above argument does not work anymore. On the other hand, a local search with $c \geq 3$ could incur a high cost. Hence, after a reduced genome $\mathcal{G}'$ is obtained through the MIS approximation, we would use integer linear programming (ILP) instead. In the next subsection, for theoretical purpose, we describe a simple FPT algorithm for the problem.

C. An FPT algorithm

We show briefly an FPT (fixed-parameter tractable) algorithm for this problem. (An FPT algorithm for a decision problem with parameter k is one which runs in $O(f(k)n^c) = O^*(f(k))$ time, where $f(-)$ is any computable function, n is the input size and c is a constant not related to n and k [10], [11].) The algorithm is very similar to the above greedy algorithm. Note that here the question is to decide whether there are k non-intersection segments spanning all the k colors.

We first take all possible permutations of the k colors, i.e., $k!$ of them. For each permutation $\pi_i = \langle i_1, i_2, ..., i_k \rangle$, we check whether there is a solution of k non-intersecting segments all with different colors such that reading their colors from left to right, we get exactly π_i. It is straightforward to see that if such a solution exists, then there must be a solution whose leftmost segment/interval is the leftmost segment/interval with color i_1. Then we could repeatedly apply this greedy idea to obtain the leftmost segment of color i_{j+1} (provided that it does not intersect the previously selected segment with color i_j), until we obtain the segment with color i_k. Since we have to try over all possible permutations of the k colors, this algorithm obviously takes $O^*(k!)$ time.

The algorithm can clearly be improved. We could use a standard dynamic programming algorithm. Let $s_1, s_2, \cdots, s_n$ be the list of intervals sorted by their right endpoints and let c_i be the color of s_i for $i = 1..n$. Assume that K is the set of k given colors and $C \subseteq K$, we define $U[C, s_i]$ as the binary table deciding whether there is a set of $|C|$ disjoint intervals in the list $s_1, s_2, ..., s_i$ such that they are all of different colors and the union of their colors is exactly C. Let $S[C, s_i]$ be the corresponding set of these $|C|$ intervals. To be more precise, $U[C, s_i]$ =TRUE if such a solution $S[C, s_i]$ exists; otherwise, $U[C, s_i]$ =FALSE. Initially, $U[\{c_i\}, s_i]$ =TRUE for all i. Given s_i, let $s_{i'}$ be the rightmost interval to the left of s_i that does not intersect s_i, i.e., i' is the largest index j satisfying $j < i$ and $s_j \cap s_i = \emptyset$. Let s_ℓ be any interval to the left of s_i, i.e., $\ell < i$. If s_ℓ does not intersect any segment in a set of segments S, we denote $s_\ell \cap S = \emptyset$.

We show how $U[C, s_i]$ is updated as follows. The update for $S[C, s_i]$ is trivial, hence omitted.

$$\begin{aligned} U[C, s_i] &= (U[C - \{c_\ell\}, s_i] \wedge (s_\ell \cap S[C - \{c_\ell\}, s_i] = \emptyset)) \quad (1) \\ &\vee\ U[C, s_{i-1}] \quad (2) \\ &\vee\ U[C - \{c_i\}, s_{i'}] \quad (3) \end{aligned}$$

We explain the three updates in more detail.

1) $U[C, s_i]$ is updated through an interval s_ℓ to the left of s_i where s_ℓ does not intersect any interval in the current solution $S[C - \{c_\ell\}, s_i]$ and its color is new.
2) $U[C, s_i]$ is updated by not including s_i in the solution.
3) $U[C, s_i]$ is updated by including s_i in the solution.

Then, there is a colored independent set spanning all the k colors iff $U[K, s_n]$ is TRUE. The actual solution can be found in $S[K, s_n]$. The running time of this dynamic programming algorithm is obviously $O^*(2^k)$ as we have 2^k subsets for k colors.

Theorem 3: MIS-CIG can be solved with an FPT algorithm in $O^*(2^k)$ time.

Unfortunately, for our application, k is usually large. (In fact, k could be as large as a couple of thousands.) Hence, this FPT algorithm is only of some theoretical meaning. To handle the practical problem, we will use integer linear programming (ILP), which will be described next.

D. ILP formulation on the reduced instance

As we use ILP in this subsection, we just use integers to represent genes. For the ease of description, we focus on unsigned genes here. (In our implementation we in fact cover both signed and unsigned genomes.)

In the reduced genome $\mathcal{G}'$ each gene appears at most twice after using the MIS approximation in a colored interval graph on the original (generic) genome $\mathcal{G}$ — where some gene might not appear in $\mathcal{G}'$ at all. This is due to that each gene i could appear in two intervals, e.g., $(i-1, i)$ and $(i, i+1)$, which could be a long distance away. Hence from $\mathcal{G}'$ we could only hope to compute a pseudo-exemplar genome (i.e., each gene appears at most once, some might never appear). In the the following, we use integer linear programming (ILP) on the reduced genome $\mathcal{G}'$ to obtain a pseudo-exemplar genome with the input being pairs of genes appearing in the form $(i, i+1)$ or $(i+1, i)$. We keep the order of genes in the intervals resulting from the greedy method. For example, $\mathcal{G}' = \boxed{78}\boxed{56}\boxed{43}\boxed{67}\boxed{23}\boxed{12}$.

Let x^k_{ij} denote the k-th interval with content (or simply color) ij with $|j-i| = 1$. As i, j each could appear twice in $\mathcal{G}'$ we have $k \le 4$, i.e., we could have at most 4 different ways to form the adjacency (i, j). Note that $j = i-1$ or $j = i+1$. We comment that we never alter H, while using the algorithm in [21] H could be altered. An example is as follows, $\mathcal{G} = 128456573$, $\mathcal{H} = 12345678$. With the algorithm in [21], the optimal solution is $G = H = 124567$, with five adjacencies. With our problem $\mathcal{H}$ is already exemplar so we cannot alter it, and the optimal solution is $G = 12845673$ and $H = 12345678$, with only four adjacencies. Moreover, our optimal solution might not be unique and it could be a pseudo-exemplar genome, e.g., $G' = 1245673$ or $G' = 124567$ (while $H = 12345678$).

In $\mathcal{G}'$, there could be two genes from each gene family. Let y_{ij} $(j = 1, 2)$ represent each gene from its gene family y_i. Two adjacent pairs $(i, i+1), (i+1, i+2)$ are called *neighboring* pairs. and two adjacent pairs $(i, i+1), (i+2, i+3)$ are called *consecutive* pairs. (Note that one or both of them could be in reverse order.) If these pairs are all sorted, then we call them *sorted-neighboring* and *sorted-consecutive* respectively. If two adjacent pairs have other pairs separating them, say $\boxed{45}\boxed{89}\boxed{67}$, we define them as *disjoint-neighboring* pairs or *disjoint-consecutive* pairs — depending on whether they share a gene or not. A *block* is composed of a sequence of neighboring or consecutive pairs. For example, $\boxed{12}\boxed{23}\boxed{45}\boxed{56}\boxed{76}\boxed{89}$. Our objective function is to maximize the number of adjacencies

$$\max \sum_{i,j} x^k_{ij}.$$

Our idea on applying ILP to the reduced instance is as follows. We identify all the blocks and we then solve them locally using ILP. Then these local solutions are put together to have a final solution. We give more details as follows. First, we perform some preprocessing by computing maximal common substrings in $\mathcal{G}'$ (with respect to H), keep the longest ones (when there is a conflict) and delete all the corresponding duplicated genes. Second, identify all the blocks and apply five groups of ILP constraints: overlapping, same-color, adjacency, non-adjacency and exemplar constraints.

Preprocessing:

1) We compute all maximal common substrings (in either a normal or reversed order) between $\mathcal{G}'$ and H and keep those with no conflict (i.e., no common intersections). For instance, with $H = 1234567$, a substring of $\mathcal{G}'$, $\boxed{45}\boxed{67}$, would be kept. If two maximal common substrings have a conflict, we would keep the longer one and delete all the duplicated genes in the shorter substring. For example, let $\mathcal{G}' = \boxed{7654321234}$ and $H = 1234567$. Then we have two maximal common substrings: $C_1 = \boxed{7654321}$ and $C_2 = \boxed{1234}$. The two substrings 4321 and 1234 cause a conflict, but $\{2, 3, 4\}$ is the set of duplicated genes. If we delete $\boxed{432}$ from C_1, we get $\boxed{7651234}$. But the optimal solution is $\boxed{7654321}$ by deleting $\boxed{234}$ in C_2.
2) If there is a pair between two disjoint-consecutive pairs, we keep the consecutive pairs and delete the pair in the middle. For example, we keep $\boxed{45}\boxed{67}$ from $\boxed{45}\boxed{89}\boxed{67}$, while $\boxed{89}$ is deleted.
3) For two sorted-neighboring pairs, we delete one gene which appears in the middle. For example in $\boxed{45}\boxed{56}$, we delete $\boxed{5}$.

The following lemma can be proved.

Lemma 1: The preprocessing procedure cannot reduce the number of adjacencies in the optimal solution.

We next illustrate all the five groups of constraints.

Adjacency constraints: Identify blocks of neighboring and consecutive pairs of genes and the number of distinct genes involved. In each block, we apply some adjacency constraint. For example, $\boxed{56}\boxed{76}\boxed{87}$ with four distinct genes. In this case, it can form at most 3 adjacencies using the following adjacency constraint.

$$x^{i_1}_{56} + x^{i_2}_{56} + x^{j_1}_{67} + x^{j_2}_{67} + x^{j_3}_{67} + x^{k_1}_{78} + x^{k_2}_{78} \le 3 \qquad (4)$$

Non-adjacency constraints: For every pair $(i, i+1)$, find the disjoint-neighboring pair $(i+1, i+2)$ in other blocks such that there are more than two genes between them. In this case, one of them could be selected. Likewise, for two disjoint-consecutive pairs $(i, i+1), (i+2, i+3)$ in different blocks with more than two genes between them, both of them could be selected. The corresponding constraints are as follows. Here we have $1 \le p, q, r, s \le 4$.

$$x^p_{i,i+1} + x^q_{i+1,i+2} \leq 1 \tag{5}$$
$$x^r_{i,i+1} + x^s_{i+2,i+3} \leq 2 \tag{6}$$

Overlapping constraints: For a set of overlapping intervals, at most one could be selected. For example, for a substring of $\mathcal{G}'$, 457689, we have three overlapping intervals and the constraint is as follows.

$$x^p_{56} + x^q_{67} + x^r_{78} \leq 1 \tag{7}$$

Same-color constraints: Given a set of identical intervals, only one of them could be selected. Note that these intervals cannot all come from the MIS solution. For example, the optimal solution in $\boxed{67}\boxed{45}\boxed{65}\boxed{78}$ is $\boxed{45678}$. We use the same color constraint to delete the first interval $\boxed{67}$:

$$x^p_{67} + x^q_{67} + x^r_{67} + x^s_{67} \leq 1 \tag{8}$$

Exemplar constraints: It is required that at most one gene from each family could appear in the final solution.

$$y_{11} + y_{12} \leq 1 \tag{9}$$
$$y_{21} + y_{22} \leq 1 \tag{10}$$
$$\vdots \tag{11}$$
$$y_{n1} + y_{n2} \leq 1 \tag{12}$$

Finally, if the variable x^k_{ij} which represents an interval (formed by a pair of genes) is selected, we also require that the two variables y_{i,l_1} and y_{j,l_2}, which are the elements of x^k_{ij}, be selected as well. If the gene y_{i,l_1} forming the adjacency x^k_{ij} is not selected, then x^k_{ij} should also be discarded. The corresponding constraints are as follows.

$$y_{i,l_1} \geq x^k_{ij} \tag{13}$$
$$y_{j,l_2} \geq x^k_{ij} \tag{14}$$

Of course, x^k_{ij} and y_{ij} must all be binary variables, representing every pair of genes and every single gene in the reduced genome.

$$x^k_{ij} \in \{0,1\}\ (i = 1,2,\cdots,n;\ j = 1,2,\cdots,m) \tag{15}$$
$$y_{ij} \in \{0,1\}\ (i = 1,2,\cdots,n;\ j = 1,2) \tag{16}$$

V. Simulation Results

Our simulation is composed of two parts. In the first part, we perform data generation and greedy exemplar selection. This part was coded in Java. The second part applies the ILP implementation to compute the pseudo-exemplar genome, which was coded in Matlab (using CPLEX for ILP). (On the other hand, the GREDU software was coded in C++ which is usually much faster; moreover, instead of CPLEX, the ILP package GUROBI was used in GREDU.) We ran our Greedy-ILP algorithm as well as GREDU on a PC with 2.5 GHz Intel Core processor and 4 GB of memory.

Our simulated data are generated in a similar way as in [21]. The dataset generator first constructs an exemplar genome H of size n, comprising of an integer sequence $[1..n]$. Each number in $[1..n]$ represents a unique gene. Then, it constructs some generic genome $\mathcal{G}$ by mutating H in m rounds, where m is a user-defined integer that roughly corresponds to the number of generations between H and $\mathcal{G}$. A mutation cycle on a genome is performed by traversing each gene g_k in the genome, and mutating the substring around g_k with some probability. Our generator can perform up to nine different mutations:

- Unit Reversal : Given a gene, g_k, switch the locations of g_k and g_{k+l} with a probability of p_1.
- Unit Insertion : Insert an arbitrary gene at location k with a probability of p_2.
- Unit Deletion : The gene g_k is removed with a probability of p_3.
- Unit Duplication : The gene g_k is copied and then inserted at location $k+1$ in the genome with a probability of p_4.
- Segment Reversal : Given some length l and a gene g_k, the ordering of the genome between genes g_k and g_{k+l} is reversed with a probability of p_5.
- Tandem Duplication : Given some length l and a gene g_k, the genes between g_k and g_{k+l} are copied and placed at location $k+l+1$ with a probability of p_6.
- Segment Deletion : Given some length l and a gene g_k, the genes between g_k and g_{k+l} are removed with a probability of p_7.
- Segment Duplication : Given some length l and a gene g_k, the sequence of the genome between g_k and g_{k+l} is copied and then inserted at a random location out of the interval $[k, k+l-1]$ in the genome with a probability of p_8.
- Transposition: Given some length l and a gene g_k, a transposition operation is performed on a segment of length l starting at g_k with a probability of p_9.

Every mutation cycle is performed on the mutated genome from the previous cycle, until m cycles have completed. Both the exemplar genome H and the genome $\mathcal{G}$ are then saved. These $(H,\mathcal{G})$ pairs are then used as test datasets for our Greedy-ILP algorithm, and also for the GREDU software that we compare against.

Throughout our simulations we use four different settings: P_1, P_2, P_3 and P_4, which are further defined as follows.

- $P_1 = \{p_1 = 0.05, p_2 = 0.10, p_3 = 0.05, p_4 = 0.05, p_5 = 0.03, p_6 = 0.06, p_7 = 0.03, p_8 = 0.10, p_9 = 0.07, l = 5\}$
- $P_2 = \{p_1 = 0.20, p_2 = 0.15, p_3 = 0.15, p_4 = 0.10, p_5 = 0.05, p_6 = 0.08, p_7 = 0.04, p_8 = 0.12, p_9 = 0.10, l = 1\}$
- $P_3 = \{p_1 = 0.20, p_2 = 0.18, p_3 = 0.10, p_4 = 0.10, p_5 = 0.05, p_6 = 0.09, p_7 = 0.05, p_8 = 0.10, p_9 = 0.00, l = 10\}$
- $P_4 = \{p_1 = 0.20, p_2 = 0.18, p_3 = 0.10, p_4 = 0.10, p_5 = 0.05, p_6 = 0.09, p_7 = 0.05, p_8 = 0.10, p_9 = 0.00, l = 5\}$

While these cases are not exhaustive with regard to the coverage of genome generations, they do provide the three unique cases of genomes that we are interested in for our comparison of our algorithm and GREDU. The first case, P_1 is designed to not mutate aggressively, and is designed to change slowly. This means that most of the gene families will still be in the same ordering as the exemplar genome, and many aligned pairs should be found. The second case, P_2, is designed for rapid mutations. Higher mutation rates coupled with a small l value mean that the genome generated will likely have a vastly different relative ordering of gene families compared to the exemplar genome, and few adjecencies should be found compared with P_1. The last two cases, P_3 and P_4, are somewhat of a mixture of P_1 and P_2. The mutation rates are still set relatively high, but the value of l is varied to 10 in P_3, and 5 in P_4. This means that not only the genome is changing rapidly, large sections of it will also be moved, copied and deleted. We do not expect too many adjacencies to be computed compared with P_2. In fact, the corresponding empirical results are slightly worse than that from P_2.

Even though the platforms (together with the adjacency definitions) between our implementation and GREDU are quite different, we compare them using the same simulated data in Table 1. All the numbers are averaged over 10 tries. Although GREDU is much faster as it is written in C++, our implementation, which is based on Java and Matlab, does not need more than 20 minutes for any case we tested (which should be fine with this kind of application). However, as can be seen in Table 1, for many P_3 datasets GREDU cannot run to completion. Our implementation is much stable and the number of adjacencies computed do not differ too much between the two (even though the definitions of adjacencies differ a bit).

We also obtained similar results for unsigned genomes, the details are listed in Table 2. Note that GREDU only handles signed genomes, hence no comparison is available.

VI. Conclusion

In this paper, we approach the One-sided EAN problem by considering a generalized version, i.e., One-sided EAN+. We handle the One-sided EAN+ problem by first relaxing it as the maximum independent set (MIS) in a colored interval graph, which open a new research direction to deal with the exemplar genomic distance problems for linear genomes. Although this new version of MIS in a colored interval graph is NP-hard, it admits a 2-approximation with the standard greedy method. We subsequently designed a $greedy+ILP$ algorithm for One-sided EAN+. Simulation results indicate that our algorithm can handle large scale (singed and unsigned) genomic data with deep evolution depth.

For the future work, it is natural to approach the EAN problem by formulating the problem as MIS in a colored 2-interval graph for two generic input genomes. When the input genomes are circular, we can also define similarly MIS in a colored circular-arc graph. However, in these cases, the greedy algorithm cannot produce a 2-approximation in these settings. Hence, a different method needs to be designed.

Acknowledgments

This research is supported by NSF (CNS-1644348 and CNS-1761641) and also by NSF of China (No. 61628027),

References

[1] S. Angibaud, G. Fertin, I. Rusu, and S. Vialette. A pseudo-boolean framework for computing rearrangement distances between genomes with duplicates. *Journal of Computational Biology*, **14**(4): 379-393, 2007.

[2] S. Angibaud, G. Fertin, I. Rusu, A. Thevenin, and S. Vialette. On the approximability of comparing genomes with duplicates. *J. Graph Algorithms and Applications*, 13(1):19-53, 2009.

[3] G. Blin, G. Fertin, F. Sikora, and S. Vialette. The exemplar breakpoint distance for non-trivial genomes cannot be approximated. In *Proc. 3nd Workshop on Algorithm and Computation (WALCOM'2009)*, LNCS 5431, pp. 357-368, 2009.

[4] D. Bryant. The complexity of calculating exemplar distances. *In D. Sankoff and J. Nadeau, editors, Comparative Genomics: Empirical and Analytical Approaches to Gene Order Dynamics, Map Alignment, and the Evolution of Gene Families*, pp. 207-212. Kluwer Acad. Pub., 2000.

[5] Z. Chen, B. Fu, and B. Zhu. The approximability of the exemplar breakpoint distance problem. In *Proc. 2nd Intl. Conf. on Algorithmic Aspects in Information and Management (AAIM'06)*, LNCS 4041, pp. 291-302, 2006.

[6] Z. Chen, B. Fu, B. Yang, J. Xu, Z. Zhao, and B. Zhu. Non-breaking similarity of genomes with gene repetitions. In *Proceedings of the 18th Annual Symposium on Combinatorial Pattern Matching (CPM'07)*, LNCS 4580, pp. 119–130, 2007.

[7] Z. Chen, B. Fu, R. Fowler, and B. Zhu. On the inapproximability of the exemplar conserved interval distance problem of genomes. *J. Combinatorial Optimization*, **15**(2):201-221, 2008.

[8] Z. Chen, B. Fu, R. Goebel, G. Lin, W. Tong, J. Xu, B. Yang, Z. Zhao, and B. Zhu. On the approximability of the exemplar adjacency number problem of genomes with gene repetitions. *Theoretical Computer Science*, **550**:59-65, 2014.

[9] G. Cormode and S. Muthukrishnan. The string edit distance matching problem with moves. *Proc. 13th ACM-SIAM Symp. on Discrete Algorithms (SODA'02)*, pp. 667-676, 2002.

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

[11] J. Flum and M. Grohe. *Parameterized Complexity Theory*, Springer-Verlag. 2006.

[12] F. Gavril. Algorithms for minimum coloring, maximum clique, minimum covering by cliques, and maximum independent set of a chordal graph. *SIAM J. Comput.*, **1**:180–187, 1972.

[13] H. Jiang, L. Qingge, D. Zhu, and B. Zhu. A 2-approximation algorithm for the contig-based scaffold filling problem. *J. Bioinformatics and Computational Biology*, **16**(6):1-15, Dec, 2018.

[14] H. Jiang, F. Zhong, and B. Zhu. Filling scaffolds with gene repetitions: maximizing the number of adjacencies. *Proc. 22nd Annual Combinatorial Pattern Matching Symposium (CPM'11)*, LNCS 6661, pp. 55-64, 2011.

[15] H. Jiang, C. Zheng, D. Sankoff, and B. Zhu. Scaffold filling under the breakpoint and related distances. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 9(4):1220-1229, July/August, 2012.

[16] M. Jiang. The zero exemplar distance problem. *Journal of computational biology*, **18**(9): 1077-1086, 2011.

[17] N. Liu, H. Jiang, D. Zhu, and B. Zhu. An improved approximation algorithm for scaffold filling to maximize the common adjacencies. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 10(4):905-913, July/August, 2013.

[18] N. Liu, D. Zhu, H. Jiang, and B. Zhu. A 1.5-approximation algorithm for two-sided scaffold filling. *Algorithmica*, 74(1):91-116, 2016.

[19] C.T. Nguyen, Y.C. Tay, and L. Zhang. Divide-and-conquer approach for the exemplar breakpoint distance. *Bioinformatics*, **21**(10):2171-2176, 2005.

[20] D. Sankoff. Genome rearrangement with gene families. *Bioinformatics*, **15**(11): 909-917, 1999.

[21] M. Shao and B. Moret. A fast and exact algorithm for the exemplar breakpoint distance. *Journal of Computational Biology*, **23**(5): 337-346, 2016.

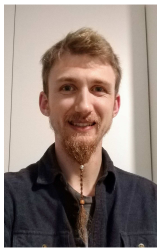
[22] B. Zhu. Genomic scaffold filling: A progress report. *Proc. 10th Intl. Frontiers of Algorithmics Workshop (FAW'16)*, LNCS 9711, pp. 8-16, 2016.

m	n	P_1		P_2		P_3		P_4	
		n_1	n_2	n_1	n_2	n_1	n_2	n_1	n_2
	1	247	242	216	224	204	198	178	182
500	3	87	41	34	34	19	20	36	33
	5	17	12	23	8	8	2	27	22
	1	512	481	391	406	301	297	354	365
1000	3	106	96	66	61	54	47	73	64
	5	24	23	33	29	23	18	30	29
	1	1552	1442	1231	1244	703	756	1022	1075
3000	3	360	330	232	222	171	164	228	218
	5	80	78	97	73	64	50	101	92
	1	2441	2307	2044	2067	1532	—	1742	1830
5000	3	658	568	439	397	204	196	310	311
	5	163	159	272	197	110	91	148	130
	1	3471	3433	2495	2525	2010	—	2400	2450
7000	3	823	740	630	512	515	409	585	486
	5	223	208	249	139	151	128	179	174
	1	4563	4296	3507	3588	3355	—	3129	3306
9000	3	1103	969	933	828	583	—	724	619
	5	257	249	276	154	177	152	253	208
	1	6183	5756	5217	5322	3423	—	4987	5021
12000	3	1383	1287	1247	1112	279	237	928	816
	5	365	332	379	239	257	205	342	318

TABLE I

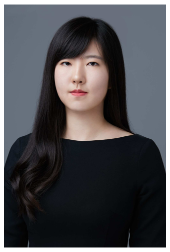
COMPARISON RESULTS (SIGNED GENOME) BETWEEN THE NUMBER OF ADJACENCIES n_1 FROM OUR ALGORITHMS AND THE NUMBER OF ADJACENCIES n_2 FROM GREDU. THE GAP — INDICATES THAT WE GET A CORE DUMPED WARNING FROM GREDU AND CAN NOT OBTAIN THE RESULTS AFTER 10 TIMES OF TRIES.

Letu Qingge Letu Qingge is currently a postdoc at College of Computing and Informatics, University of North Carolina-Charlotte, NC, USA. His obtained his PhD in Computer Science in 2018 from Montana State University, Bozeman, MT, USA. His research interests are computational biology and algorithms.

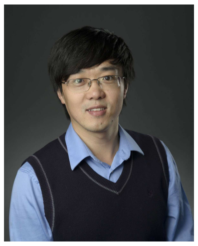


Killian Smith Killian Smith is currently an MS student in computer science at Gianforte School of Computing, Montana State University, Bozeman, MT, USA.

Sean Jungst Sean Jungst is currently an undergraduate student in computer science at Gianforte School of Computing, Montana State University, Bozeman, MT, USA.



Baihui Wang Baihui Wang is currently a MS student at Martin J. Whitman School of Management, Syracuse University, USA. Her research interests are algorithms and statistical methods.



Qing Yang Qing Yang is currently an assistant professor in computer science and engineering at University of North Texas, USA. He obtained his Ph.D. in Computer Science from Auburn University, USA in 2011. From 2011 to 2017, he was an assistant professor at Montana State University, Bozeman, MT, USA. His research interests are Internet of Things, autonomous vehicular system, network security and privacy. He has published over 40 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 190 papers in these areas.

m	n	P_1		P_2		P_3		P_4	
		t_1	n_1	t_1	n_1	t_1	n_1	t_1	n_1
	1	0	237	0	223	0	207	0	210
500	3	0	86	0	96	0	82	0	85
	5	0	20	0	43	0	27	0	39
	1	0	497	0	425	0	377	0	380
1000	3	0	116	0	165	0	133	0	135
	5	0	41	1	82	0	49	0	51
	1	29	1450	30	1281	22	1189	25	1204
3000	3	6	398	27	467	2.5	434	2.5	431
	5	3	127	33	207	0	181	0	196
	1	127	2214	142	2181	88	2025	111	2076
5000	3	30	662	105	726	15	431	21	469
	5	8	159	145	328	2	178	3.5	211
	1	368	3317	398	3097	260	2694	312	2723
7000	3	70	768	326	998	30	614	43	741
	5	26	263	435	453	10	235	21	279
	1	765	4368	868	4179	446	3650	722	3876
9000	3	207	985	799	1245	80	857	112	911
	5	64	310	879	632	25	345	41	434
	1	1691	5789	1877	5564	1006	4822	1552	5107
12000	3	409	1321	1521	1532	158	1017	233	1112
	5	120	465	1945	913	58	397	77	498

TABLE II
THE RUNNING TIME t_1 (SECONDS) AND THE NUMBER OF ADJACENCIES n_1 FOR UNSIGNED GENOMES, AVERAGED OVER 10 TRIES. NO COMPARISON IS DONE WITH GREDU AS IT ONLY HANDLES SIGNED GENOMES.