Genomic Scaffold Filling: A Progress Report

Binhai Zhu(✉)

Department of Computer Science,
Montana State University, Bozeman, MT 59717-3880, USA
bhz@montana.edu

Abstract. The genomic scaffold filling problem has attracted a lot of attention since 2010. The general problem is on filling an incomplete sequence (sequence scaffold) I into I', with respect to a complete reference genome G, such that the number of adjacencies between G and I' is maximized. The problem is NP-complete and APX-hard, and admits a 1.2-approximation. In this survey paper, we will first review the progress being made for this setting.

However, the sequence input I is not quite practical and does not fit most of the real datasets (where a scaffold is more often given as a list of contigs). Then, we will review the most recent progress on this new version of the genomic scaffold filling problem where, (1) a scaffold S is given, the missing genes $X = c(G) - c(S)$ can only be inserted in between the contigs, and the objective is to maximize the number of adjacencies between G and the filled S', and (2) a scaffold S is given, a subset of the missing genes $X' \subset X = c(G) - c(S)$ can only be inserted in between the contigs, and the objective is still to maximize the number of adjacencies between G and the filled S''. Some open problems will be posed for further research.

1 Introduction

Since 2001, the cost of sequencing a genome has been reduced significantly, with the current cost being around \$1 k. This results in a lot of genomes being sequenced, usually not completely finished (they are typically called *draft* genomes). On the other hand, the cost to finish these genomes completely has not been decreased as much [6]. The result is that we are having more and more draft genomes. Nonetheless, for many tools analyzing the genomic data we do need complete genomes. For instance, to compute the reversal distance between two genomes we do need two complete genomes. Hence, there is a need to turn a draft genome into a complete one.

To make the result biologically interesting, Munoz *et al.* first proposed the following *scaffold filling* problem (on multichromosomal genomes with no gene repetition) as follows [24]. Given a complete (permutation) genome R and an incomplete scaffold S, fill the missing genes in $R - S$ into S to have S' such that the genomic distance (or DCJ distance [26]) between R and S' is minimized. It was shown that this problem can be solved in polynomial time. In [16], Jiang *et al.* considered the case for singleton genomes without gene repetition

D. Zhu and S. Bereg (Eds.): FAW 2016, LNCS 9711, pp. 8–16, 2016.
DOI: 10.1007/978-3-319-39817-4_2

(i.e., permutations), using the simplest *breakpoint* distance as the similarity measure. It was not surprising that this problem was shown to be polynomially solvable; in fact, even for the two-sided case when both the input scaffolds, being a reference to each other, are incomplete permutations.

When the genomes and scaffolds contain gene repetitions, the problem becomes harder. (That should not be considered as a surprise as even computing certain similarity measure between two complete genomes is NP-complete, for instance, with the exemplar breakpoint distance [2,4,7,9,19], exemplar adjacency number [8,10], or the minimum common string partition [11].) The similarity measure adopted for the scaffold filling problem is the *number of common (string) adjacencies*, which can be computed in polynomial time [2,15,16]. In [15,16], it was shown by Jiang *et al.* that filling a scaffold to maximize the number of common string adjacencies (SF-MNSA) is NP-hard. (Formally, the problem is to fill an incomplete sequence scaffold I into I', with respect to a complete reference genome G, such that the missing letters in $G-I$ are inserted back to I and the number of common adjacencies between G and I' is maximized.) A factor-1.33 approximation was designed in [15,16], and this bound has been improved to 1.25 [21], and to 1.20 [17]. For the corresponding two-sided case, i.e., when two sequence scaffolds are references to each other, the problem admits a factor-1.5 approximation with the number of common adjacencies between the filled scaffolds being maximized [22]. Using the number of common adjacencies as a parameter, it was shown that this problem is also fixed-parameter tractable (FPT) — this only handles that case when G and I' are not very similar so it is only of a theoretical meaning [5].

Recently, a practical factor is seriously considered [18]. Firstly, the 'scaffold' used in most of these papers is an incomplete sequence, i.e., a missing gene can be inserted anywhere in such a 'scaffold'. In practice, most of the real datasets are not in this format; in fact, a scaffold in a real dataset is usually composed of a sequence of contigs, where a contig is usually computed with mature tools like Celera Assembler [1], hence should not be arbitrarily altered. This case was only briefly considered a few years ago [16,24]. Secondly, take a complete reference genome G and a scaffold S, there is no guarantee that the filled scaffold S' is of the same length as that of G; in fact, sometimes we could know roughly the length of the target genome S^* (S' should be as close to S^* as possible). Then, we might only need to insert a subset of letters in $G-S$ into S (to obtain S').

Formally, the above two problems are called One-sided Scaffold Filling (One-sided-SF-max), and One-sided Subset Scaffold Filling (One-sided-SF-max($\subset$)) respectively. (For the important practical case when a gene can only appear at most d times in G, we call the corresponding problems One-sided-SF-max(d) and One-sided-SF-max($\subset, d$) respectively.) The objective function in both cases are to maximize the number of common adjacencies between the reference and the filled scaffold.

The paper is organized as follows. In Sect. 2, we give the preliminaries. In Sect. 3, we review the results in three categories: when genomes have no gene duplications, approximation results for the one-sided cases, and FPT results for the one-sided cases. We conclude the paper in Sect. 4 by giving directions for further research.

2 Preliminaries

Throughout this paper we focus only on singleton genomes (i.e., each is a sequence). But the results can be easily generalized to multichromosomal or circular genomes, with minor changes.

At first, we review some necessary definitions, which are also defined in [16, 27]. We assume that all genes and genomes are unsigned, and it is straightforward to generalize the result to signed genomes. Given a gene set Σ, a string P is called *permutation* if each element in Σ appears exactly once in P. We use *c(P)* to denote the set of elements in permutation P. A string A is called *sequence* if some genes appear more than once in A, and $c(A)$ denotes genes of A, which is a multi-set of elements in Σ. For example, $\Sigma = \{a,\ b,\ c,\ d\}$, $A = abcdacd$, *c(A)* $= \{a,\ a,\ b,\ c,\ c,\ d,\ d\}$. A *sequence scaffold* is an incomplete sequence, typically obtained by some sequencing and assembling process. A substring with m genes (in a sequence) is called an *m-substring*, and a 2-substring is also called a *pair*; as the genes are unsigned, the relative order of the two genes of a pair does not matter, i.e., the pair xy is equal to the pair yx. Given an incomplete sequence (or sequence scaffold) $A = a_1 a_2 a_3 \cdots a_n$, let $P_A = \{a_1a_2, a_2a_3, \ldots, a_{n-1}a_n\}$ be the set of pairs in A.

Definition 1. *Given two sequence scaffolds $A = a_1a_2 \cdots a_n$ and $B = b_1b_2 \cdots b_m$, if $a_ia_{i+1} = b_jb_{j+1}$ (or $a_ia_{i+1} = b_{j+1}b_j$), where $a_ia_{i+1} \in P_A$ and $b_jb_{j+1} \in P_B$, we say that a_ia_{i+1} and b_jb_{j+1} are matched to each other. In a maximum matching of pairs in P_A and P_B, a matched pair is called an* ***adjacency****, and an unmatched pair is called a* ***breakpoint*** *in A and B respectively.*

It follows from the definition that sequence scaffolds A and B contain the same set of adjacencies but distinct breakpoints. The maximum matched pairs in B (or equally, in A) form the *(common) adjacency set* between A and B, denoted as $a(A, B)$. We use $b_A(A, B)$ and $b_B(A, B)$ to denote the set of breakpoints in A and B respectively. We illustrate the above definitions in Fig. 1.

For a sequence A and a multi-set of elements X, let $A + X$ be the set of all possible resulting sequences after filling all the elements in X into A. We define a contig as a string over a gene set Σ whose contents should not be altered. A *scaffold S* is simply a sequence of contigs $\langle C_1, ..., C_m\rangle$. We define $c(S) = c(C_1) \cup \cdots \cup c(C_m)$. Now, we define the problems on scaffolds formally.

Given two incomplete sequences (or sequence scaffolds) $A = a_1a_2 \cdots a_n$ and $B = b_1b_2 \cdots b_m$, as we can see, each gene except the four ending ones is involved in two adjacencies or two breakpoints or one adjacency and one breakpoint. To get rid of this imbalance, we add "#" to both ends of A and B.

Definition 2. *Scaffold Filling to Maximize the Number of (String) Adjacencies (SF-MNSA).*

Input*: two sequence scaffolds A and B over a gene set Σ and two multi-sets of elements X and Y, where $X = c(B) - c(A)$ and $Y = c(A) - c(B)$.*

Question*: Find $A^* \in A + X$ and $B^* \in B + Y$ such that $|a(A^*, B^*)|$ is maximized.*

$$\begin{aligned} sequence\ scaffold\ A &= \langle c\ b\ c\ e\ d\ a\ b\ a\ \rangle \\ sequence\ scaffold\ B &= \langle a\ b\ a\ b\ d\ c \rangle \\ P_A &= \{cb, bc, ce, ed, da, ab, ba\} \\ P_B &= \{ab, ba, ab, bd, dc\} \\ matched\ \ pairs\ &:\ (ab \leftrightarrow ba), (ba \leftrightarrow ab) \\ a(A, B) &= \{ab, ba\} \\ b_A(A, B) &= \{cb, bc, ce, ed, da\} \\ b_B(A, B) &= \{ab, bd, dc\} \end{aligned}$$

Fig. 1. An example for adjacency and breakpoint definitions.

The one-sided SF-MNSA problem is a special instance of the SF-MNSA problem where one of X and Y is empty. We formally define it as follows.

Definition 3. *One-sided SF-MNSA.*

Input*: a complete sequence G and an incomplete sequence scaffold I over a gene set Σ, a multi-set $X = c(G) - c(I) \neq \emptyset$ with $c(I) - c(G) = \emptyset$.*

Question*: Find $I^* \in I + X$ such that $|a(I^*, G)|$ is maximized.*

Note that while the two-sided SF-MNSA problem is more general and more difficult, the One-Sided SF-MNSA problem is more practical as a lot of genome analysis are based on some reference genome [24]. We next consider the usual scaffolds composed of sequences of contigs.

Definition 4. *One-Sided-SF*-max*.*

Input*: a complete genome G and a scaffold $S = \langle C_1, C_2, ..., C_m \rangle$ where G and the contig C_i's are over a gene set Σ, a multiset $X = c(G) - c(S) \neq \emptyset$.*

Question*: Find $S^* \in S + X$ such that $|a(S^*, G)|$ is maximized.*

One-Sided-SF-max($\subset$) is exactly the same as One-Sided-SF-max except that only a subset $X' \subset X$ need to be inserted into S. When a gene can appear at most d times in G, the two versions of problems are abbreviated as One-Sided-SF-max(d) and One-Sided-SF-max($\subset, d$) respectively.

3 Current Status

We review the current status for the research on the genomic scaffold filling problems.

3.1 Results on Filling Permutation Scaffolds

When the genomes contain no duplicated genes, the genomic scaffold filling problems are all known to be polynomially solvable. For the one-sided case, when the distance measure is DCJ (double-cut-and-join) and the scaffolds are composed of contigs, Munoz *et al.* used the *breakpoint graph* to obtain a polynomial-time solution (in fact, a linear-time solution after the breakpoint graph is constructed). Later, the result was generalized to the two-sided case by Jiang *et al.* [16]. When the distance measure is the breakpoint distance, and when the scaffolds are incomplete permutations (meaning missing genes can be inserted anywhere), Jiang *et al.* showed that both of the one-sided and two-sided problems can be solved in $O(n^2)$ time, where n is the total number of genes in the input [16].

Recently, Liu et al. [23] studied the one-sided scaffold filling problem when the scaffold S is given as a set of contigs, i.e., $S = \langle C_1, C_2, \cdots, C_m \rangle$. Let the complete reference genome (permutation) be R and let $X = c(R) - c(S)$. Let $\alpha(C_i), \beta(C_i)$ be the first and last letter of C_i respectively. Then $\langle \beta_i, \alpha_{i+1} \rangle$ (or simply $\beta_i \alpha_{i+1}$) constitutes a *slot* where missing genes can inserted between β_i and α_{i+1}. We write $\beta_0 = -\infty$ and $\alpha_{m+1} = +\infty$, where $\langle -\infty, \alpha_1 \rangle$ and $\langle \alpha_m, +\infty \rangle$ are the leftmost and rightmost (open) slot respectively.

Define a type-1 (resp. type-2) substring s of length $\ell \geq 1$, over X, as one which can be inserted in the slot $\langle \beta_i, \alpha_{i+1} \rangle$, for some i, to increase the total number of adjacencies by $\ell + 1$ (resp. ℓ).

Note that if $\beta_i \alpha_{i+1}$ is already an adjacency with respect to R, then in general it is possible that s is inserted in the slot to generate $|s| + 1$ adjacencies (while destroying the adjacency $\beta_i \alpha_{i+1}$). However, when the genome contains no gene duplication it can be shown that there exists an optimal solution which always preserves such an existing adjacency $\beta_i \alpha_{i+1}$.

Similarly, define a type-3 substring s of length $\ell \geq 1$, over X, as one which can be inserted in the slot $\langle \beta_i, \alpha_{i+1} \rangle$, for some i, to increase the number of adjacencies by $\ell - 1$. Note that a type-3 substring can only form adjacencies internally, hence it does not matter where we insert s — provided that it does not destroy any existing adjacency.

Clearly, due to that there is no gene duplication in R, a type-i substring with length ℓ must be a substring (or the reversal of a substring) of R. (This property does not hold when there are gene duplications.) We show an example as follows:

$$R = \langle 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15 \rangle,$$

$$S = \langle \boxed{1,3}, \boxed{5,6}, \boxed{15,2}, \boxed{14,13,10,9} \rangle.$$

We have $\alpha_1 = 1, \beta_1 = 3$, $\alpha_2 = 5, \beta_2 = 6$, $\alpha_3 = 15, \beta_3 = 2$, $\alpha_4 = 14, \beta_4 = 9$. Then, $X = \{4, 7, 8, 11, 12\}$ are missing from S. The optimal solution is

$$S^* = \langle \boxed{1,3}, 4, \boxed{5,6}, 7, 8, \boxed{15,2}, 11, 12, \boxed{14,13,10,9} \rangle.$$

In this case, 4 is type-1, $\langle 7, 8 \rangle$ is type-2, and $\langle 11, 12 \rangle$ is type-3.

Then, Liu et al. [23] tried to first identify type-i substrings, and then fill them in the other of $i = 1, 2, 3$. The running time is dominated by the computation of a maximum matching in a bipartite graph, which takes $O(n^{2.5})$ time. We summarize the results in following Table 1.

Table 1. Results on scaffold filling when the genomes contain no gene duplications. In the first column, 'contigs' means a scaffold is composed of a list of contigs, 'permutation' means a scaffold is an incomplete permutation.

Problem	Distance/Similarity measure	Status
One-sided, singleton, contigs	breakpoint/adjacency number	P [23]
One-sided, multichromosome, contigs	DCJ	P [24]
two-sided, singleton, permutation	breakpoint/adjacency number	P [16]
two-sided, multichromosome, contigs	DCJ	P [16]

3.2 Results on One-Sided Scaffold Filling

When the genomes contain duplicated genes, the genomic scaffold filling problems become NP-hard. The past effort has been mainly focused on the one-sided case. (We comment that the two-sided SF-MNSA can be approximated with a factor 1.5 [22].) So we will focus on One-sided SF-MNSA and One-sided-SF-max, representing that the scaffold is an incomplete sequence and a list of contigs respectively.

For One-sided SF-MNSA, Jiang *et al.* first showed that it is NP-hard by a reduction from Exact Cover by 3-Sets (X3C) [15,16]. Subsequently, a factor-1.33 approximation was given [15,16]. The main idea is that when a scaffold is a sequence (meaning missing genes can be inserted anywhere), there is no type-3 substrings. Then, the idea is to maximize the inserted type-1 substrings of length one and two, using a greedy method. Finally, for the remaining genes, it can be done so that each inserted one contributes at least one adjacency. (This last step is in fact not trivial, the details were filled later [21,27].)

In [21], the approximation factor was improved to 1.25. The idea was to insert i-type-1, $i = 1, 2, 3$, substrings using a mixture of greedy search, maximum matching and local search. Then, for the remaining genes, it can be done so that each inserted one contributes at least one adjacency. Recently, the approximation factor for One-sided SF-MNSA was improved to 1.2 and the problem was shown to be APX-hard [17]. The method was based on non-oblivious local search [20]. (As of this writing, it seems there was a small bug in the proof and a paper was devoted in this proceeding to fix that.)

While the research on the One-sided SF-MNSA has been fruitful, it does not help much on solving the practical problem. The main reason is that a scaffold in reality is usually computed with mature tools, like Celera Assembler [1]. Hence, in real datasets a scaffold is usually given as a list of contigs, each should not be altered arbitrarily. Recently, research on this version, formally called One-sided-SF-max, has been started.

In [18], a simple reduction from Hamiltonian Path to One-sided-SF-max was constructed. Thus, the problem is NP-hard. Then, a factor-2 approximation was presented using greedy search on type-1 substrings of length one and two, also on type-2 substrings of length one. Finally, a maximum matching method was used to make sure that for each pair *useful* genes, at least one adjacency can be

Table 2. Approximation results on One-sided SF-MNSA, Two-sided SF-MNSA, and One-sided-SF-max.

Problem	Similarity measure	Approximation ratio
One-sided SF-MNSA	adjacency number	1.33 [15,16]
One-sided SF-MNSA	adjacency number	1.25 [21]
One-sided SF-MNSA	adjacency number	1.20 [17]
Two-sided SF-MNSA	adjacency number	1.50 [22]
One-sided-SF-max	adjacency number	2.00 [18]

computed. (Formally, a gene is useful if it can contribute some adjacency in an optimal solution.)

We summarize the approximation results for One-sided SF-MNSA, Two-sided SF-MNSA, and One-sided-SF-max in following Table 2.

3.3 Parameterized Results on Scaffold Filling

In this subsection, we briefly review the FPT results on scaffold filling. (Readers are referred to [12,13,25] for standard FPT concepts and definitions.) In fact, as scaffold filling is a maximization problem, any FPT algorithm parameterized on the solution size is only of theoretical meaning.

In [5], Bulteau *et al.* showed that SF-MNSA is FPT, and the running times are $O^*(2^{O(k)})$ for the one-sided case and $O^*(2^{O(k \log k)})$ for the two-sided case respectively. The technique is based on color-coding [3] and subset enumeration.

In [18], it was shown that One-sided-SF-max(d) is FPT and the running time is $O^*((2d)^{O(k)})$ and when d is unbounded whether it is FPT is still open. On the other hand, it was shown that One-sided-SF-max$(\subset)$, parameterized by the number of genes inserted, is W[1]-hard. The reduction is from the Partial Vertex Cover (PVC) problem (via the standard FPT-reduction from Independent Set to PVC) [14]. Again, we list the corresponding results in following Table 3.

4 Future Research Directions

In this paper, we have thoroughly reviewed the current research on scaffold filling problems. While many theoretically interesting results have been obtained, a lot

Table 3. FPT results/status on SF-MNSA, One-sided-SF-max and One-sided-SF-max$(\subset)$.

Problem	Parameter k	FPT status
SF-MNSA	adjacency number	FPT [5]
One-sided-SF-max	adjacency number	open [18]
One-sided-SF-max$(\subset)$	number of genes inserted	W[1]-hard [18]

still need to be done to make practical impact on the problem. We list some of these problems as follows.

1. For One-sided-SF-max, is it possible to design an approximation algorithm with a factor less 2? less than 1.5?
2. For One-sided-SF-max, is it possible to design an FPT algorithm parameterized by the solution size? As a positive answer will not solve the practical problem, how about using the number of breakpoints as a parameter? Is the problem FPT then?
3. For One-sided-SF-max($\subset$), is it possible to design an FPT approximation with a factor less than 2?

Acknowledgments. I would like to thank my collaborators for this series of research: Chenglin Fan, Haitao Jiang, Nan Liu, David Sankoff, Boting Yang, Chunfang Zheng, Farong Zhong, Daming Zhu and Peng Zou.

References

1. http://wgs-assembler.sourceforge.net/
2. Angibaud, S., Fertin, G., Rusu, I., Thevenin, A., Vialette, S.: On the approximability of comparing genomes with duplicates. J. Graph Algorithms Appl. **13**(1), 19–53 (2009)
3. Alon, N., Yuster, R., Zwick, U.: Color-coding. J. ACM **42**(4), 844–856 (1995)
4. Blin, G., Fertin, G., Sikora, F., Vialette, S.: The exemplar breakpoint distance for non-trivial genomes cannot be approximated. In: Das, S., Uehara, R. (eds.) WALCOM 2009. LNCS, vol. 5431, pp. 357–368. Springer, Heidelberg (2009)
5. Bulteau, L., Carrieri, A.P., Dondi, R.: Fixed-parameter algorithms for scaffold filling. Theoret. Comput. Sci. **568**, 72–83 (2015)
6. Chain, P.S., Grafham, D.V., Fulton, R.S., et al.: Genome project standards in a new era of sequencing. Science **326**, 236–237 (2009)
7. Chen, Z., Fu, B., Zhu, B.: The approximability of the exemplar breakpoint distance problem. In: Cheng, S.-W., Poon, C.K. (eds.) AAIM 2006. LNCS, vol. 4041, pp. 291–302. Springer, Heidelberg (2006)
8. Chen, Z., Fu, B., Xu, J., Yang, B., Zhao, Z., Zhu, B.: Non-breaking similarity of genomes with gene repetitions. In: Ma, B., Zhang, K. (eds.) CPM 2007. LNCS, vol. 4580, pp. 119–130. Springer, Heidelberg (2007)
9. Chen, Z., Fu, B., Fowler, R., Zhu, B.: On the inapproximability of the exemplar conserved interval distance problem of genomes. J. Comb. Optim. **15**(2), 201–221 (2008)
10. Chen, Z., Fu, B., Goebel, R., Lin, G., Tong, W., Xu, J., Yang, B., Zhao, Z., Zhu, B.: On the approximability of the exemplar adjacency number problem of genomes with gene repetitions. Theoret. Comput. Sci. **550**, 59–65 (2014)
11. Cormode, G., Muthukrishnan, S.: The string edit distance matching problem with moves. In: Proceedings of the 13th ACM-SIAM Symposium on Discrete Algorithms (SODA 2002), pp. 667–676 (2002)
12. Downey, R., Fellows, M.: Parameterized Complexity. Springer, New York (1999)
13. Flum, J., Grohe, M.: Parameterized Complexity Theory. Springer, Heidelberg (2006)

14. Guo, J., Niedermeier, R., Wernicke, S.: Parameterized complexity of vertex cover variants. Theor. Comput. Syst. **41**(3), 501–520 (2007)
15. Jiang, H., Zhong, F., Zhu, B.: Filling scaffolds with gene repetitions: maximizing the number of adjacencies. In: Giancarlo, R., Manzini, G. (eds.) CPM 2011. LNCS, vol. 6661, pp. 55–64. Springer, Heidelberg (2011)
16. Jiang, H., Zheng, C., Sankoff, D., Zhu, B.: Scaffold filling under the breakpoint, related distances. IEEE/ACM Trans. Comput. Biol. Bioinform. **9**(4), 1220–1229 (2012)
17. Jiang, H., Ma, J., Luan, J., Zhu, D.: Approximation and nonapproximability for the one-sided scaffold filling problem. In: Xu, D., Du, D., Du, D. (eds.) COCOON 2015. LNCS, vol. 9198, pp. 251–263. Springer, Heidelberg (2015)
18. Jiang, H., Fan, C., Yang, B., Zhong, F., Zhu, D., Zhu, B.: Genomic scaffold filling revisited. In: Proceedings of the 27th Annual Symposium on Combinatorial Pattern Matching (CPM 2016), Tel Aviv, Israel, 27–29 June 2016 (2016)
19. Jiang, M.: The zero exemplar distance problem. In: Tannier, E. (ed.) RECOMB-CG 2010. LNCS, vol. 6398, pp. 74–82. Springer, Heidelberg (2010)
20. Khanna, S., Motwani, R., Sudan, M., Vazirani, U.: On syntactic versus computational views of approximability. SIAM J. Comput. **28**(1), 164–191 (1998)
21. Liu, N., Jiang, H., Zhu, D., Zhu, B.: An improved approximation algorithm for scaffold filling to maximize the common adjacencies. IEEE/ACM Trans. Comput. Biol. Bioinform. **10**(4), 905–913 (2013)
22. Liu, N., Zhu, D., Jiang, H., Zhu, B.: A 1.5-approximation algorithm for two-sided scaffold filling. Algorithmica **74**(1), 91–116 (2016)
23. Liu, N., Zou, P., Zhu, B.: A polynomial time solution for permutation scaffold filling. Technical report, Department of Computer Science, Montana State University (2016)
24. Muñoz, A., Zheng, C., Zhu, Q., Albert, V., Rounsley, S., Sankoff, D.: Scaffold filling, contig fusion and gene order comparison. BMC Bioinform. **11**, 304 (2010)
25. Niedermeier, R.: Invitation to Fixed-Parameter Algorithms. Oxford University Press, New York (2006)
26. Yancopoulos, S., Attie, O., Friedberg, R.: Efficient sorting of genomic permutations by translocation, inversion and block interchange. Bioinformatics **21**, 3340–3346 (2005)
27. Zhu, B.: A retrospective on genomic preprocessing for comparative genomics. In: Chauve, C., et al. (eds.) Models and Algorithms for Genome Evolution, pp. 183–206. Springer, London (2013)

http://www.springer.com/978-3-319-39816-7

Frontiers in Algorithmics
10th International Workshop, FAW 2016, Qingdao,
China, June 30- July 2, 2016, Proceedings
Zhu, D.; Bereg, S. (Eds.)
2016, XVII, 290 p. 73 illus., Softcover
ISBN: 978-3-319-39816-7