We are IntechOpen, the world's leading publisher of Open Access books
Built by scientists, for scientists

5,000
Open access books available

125,000
International authors and editors

140M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS
BOOK CITATION INDEX
INDEXED

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index
in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

Chapter

Scaffolding Contigs Using Multiple Reference Genomes

Yi-Kung Shieh, Shu-Cheng Liu and Chin Lung Lu

Abstract

Scaffolding is an important step of the genome assembly and its function is to order and orient the contigs in the assembly of a draft genome into larger scaffolds. Several single reference-based scaffolders have currently been proposed. However, a single reference genome may not be sufficient alone for a scaffolder to correctly scaffold a target draft genome, especially when the target genome and the reference genome have distant evolutionary relationship or some rearrangements. This motivates researchers to develop the so-called multiple reference-based scaffolders that can utilize multiple reference genomes, which may provide different but complementary types of scaffolding information, to scaffold the target draft genome. In this chapter, we will review some of the state-of-the-art multiple reference-based scaffolders, such as Ragout, MeDuSa and Multi-CAR, and give a complete introduction to Multi-CSAR, an improved extension of Multi-CAR.

Keywords: bioinformatics, sequencing, contig, scaffolding, multiple reference genomes

1. Introduction

Due to recent advances in next-generation sequencing (NGS) technologies, more and more genomes of organisms can be sequenced quickly at a moderate cost [1]. However, assembling a large number of reads generated from current NGS sequencing platforms into a complete genome still is a challenging job [2]. Largely because of repetitive sequences, whose lengths are often larger than those of the reads, most of assembled sequences are just *draft* genomes that usually consists of several hundreds or even thousands of *contigs* (contiguous sequences). The availability of complete genomes actually is significant to the downstream analysis and interpretation of their sequences in many biological applications [3]. To further obtain more complete sequences of draft genomes, therefore, the contigs of the draft genomes usually are required to be ordered and oriented into *scaffolds*, which actually are larger gap-containing sequences whose gaps between the scaffolded contigs can be closed later in the gap-filling process [4].

The scaffolding process utilizes a genomic sequence available from a related organism to serve as a *reference* to scaffold the contigs of a draft genome. So far, many such reference-based scaffolders have been proposed [5–14]. The algorithms used to develop all these scaffolders can be classified into two main categories: the *alignment-based* algorithms [5–10] and the *rearrangement-based* algorithms [11–14]. The alignment-based scaffolding algorithms first align contigs in a target draft genome against a reference sequence and then scaffold the contigs according to the

positions of their matches in the reference. On the other hand, the rearrangement-based scaffolding algorithms utilize the concept of genome rearrangements to scaffold the contigs of the target draft genome such that the sequence markers (or genes) shared between the scaffolded target and reference genomes have similar order and orientation as much as possible.

In some cases, it may be insufficient for a scaffolder to utilize only one single genome as the reference for correctly computing the scaffolds of a target draft genome, in particular when the target and reference genomes have a distant phylogenetic relationship or they have undergone some kinds of rearrangements, such as reversals, transpositions, block-interchanges and translocations. This situation inspires the requirement for developing multiple reference-based scaffolders, expecting that they can refer to several different but complementary genomes to order and orient the contigs of the target genome.

2. State-of-the-art multiple reference-based scaffolders

Below, we review three state-of-the-art multiple reference-based scaffolders: Ragout [15], MeDuSa [16] and Multi-CAR [17].

2.1 Ragout

Ragout (Reference-Assisted Genome Ordering UTility) is a rearrangement-based scaffolder for ordering and orienting the contigs of a draft genome using multiple reference genomes [15]. The input of Ragout includes a target draft genome, multiple reference genomes, and a phylogenetic tree between them. Ragout uses different colors to display the target and reference genomes and further represents all of these genomes as sequences of *synteny blocks*. Ragout then creates a so-called *incomplete multi-color breakpoint graph*, in which vertices represent the ends of synteny blocks and edges denote adjacencies of two synteny blocks occurring in the target and reference genomes. For the purpose of distinction, the edges are also colored by Ragout using the colors of the corresponding genomes. Because the target genome is already fragmented into contigs, some adjacencies of synteny blocks in the target genome are missing. Ragout tries to recover these missing adjacencies by using other existing adjacencies from the reference genomes. In the recovery process, Ragout computes the parsimony costs of all possible missing adjacencies by solving a so-called *half-breakpoint state parsimony problem* on the given phylogenetic tree, which actually is an NP-hard (non-deterministic polynomial time-hard) problem, meaning that it is hard to compute its optimal solution in polynomial time. Therefore, a heuristic approach is applied by Ragout to calculate the approximate parsimony costs of all the missing adjacencies. A perfect matching with minimum cost is then computed by Ragout on a graph created by using the missing adjacencies and is further used to scaffold the contigs of the target genome. Actually, the above procedure is repeated by Ragout multiple times with using different sizes of synteny blocks and moreover the scaffolding results obtained from all these iterations are then combined into a single set of scaffolds. Finally, a refinement is performed by Ragout to insert a number of small but repetitive contigs back to the resulting scaffolds.

2.2 MeDuSa

MeDuSa (Multi-Draft based Scaffolder) is a multiple reference-based scaffolder that does not require a given phylogenetic tree for the target and references

genomes [16]. From the given target and reference genomes, MeDuSa constructs a so-called *scaffolding graph*, which denotes by vertices the contigs of the target genome and by edges the adjacencies between any two contigs when they can be mapped to the reference genomes. Moreover, each edge in the scaffolding graph is associated with a *weight* to represent the number of reference genomes supporting the existence of the edge. As a result, it is not hard to see that a *path cover*, which is a set vertex-disjoint paths covering all the vertices of the scaffolding graph, denotes a set of scaffolds in the target genome. Unfortunately, however, finding a path cover of maximum weight in a graph is already known as an NP-hard problem. Therefore, MeDuSa utilizes a 2-approximation algorithm to find an approximate path cover from the scaffolding graph. Finally, MeDuSa applies a majority rule to determine the orientations of contigs on each path of the approximate path cover.

2.3 Multi-CAR

Multi-CAR (Multiple reference-based Contig Assembly using Rearrangements) is multiple-reference version of CAR (Contig Assembly using Rearrangements) [17]. CAR actually is a single reference-based scaffolder that utilizes a complete reference genome to scaffold the contigs of a target draft genome [13]. Like MeDuSa, Multi-CAR does not require prior knowledge concerning phylogenetic relationships among target and reference genomes. However, in contrast to Ragout and MeDuSa, both attempting to solve an NP-hard problem in their scaffolding processes, the algorithm behind Multi-CAR involves only polynomially solvable problems, as described as follows. First, Multi-CAR utilizes CAR to compute a single reference-derived scaffolding result for a target draft genome based on each of multiple reference genomes. Second, Multi-CAR uses all single reference-derived scaffolds to build an edge-weighted *contig adjacency graph*. In this contig adjacency graph, the vertices denote extremities of contigs (i.e., each contig is represented by two vertices) and the edges represent whether two contigs are ordered consecutively in a scaffold returned by CAR based on a single reference genome (if so, the adjacent extremities of these two contigs are connected by an edge). In addition, if there are multiple reference genomes to *support* an edge connection, then this edge will be assigned a weight that equals to the sum of the weights of the supporting reference genomes. The weight of each reference genome is given by the users in advance; otherwise, it is defaulted to one. Third, Multi-CAR continues to find a maximum weighted perfect matching from the contig adjacency graph. Finally, Multi-CAR constructs a multiple reference-derived scaffold for the target draft genome according to the maximum weighted perfect matching.

3. A recent multiple reference-based scaffolder

In this section, we give a detailed introduction to a recent multiple reference-based scaffolder, called Multi-CSAR (Multiple reference-based Contig Scaffolder using Algebraic Rearrangements), which is an improved extension of Multi-CAR [18]. Unlike Ragout and MeDuSa, Multi-CAR actually can not accept incomplete genomes as references, which greatly limits the widespread adoption of Multi-CAR because complete reference genomes are not always available for a target draft genome in practical usage [19]. In addition, the weight of all reference genomes used by Multi-CAR must be assigned by the users; otherwise, they are defaulted to one. However, it is usually not easy for the ordinary users to correctly determine these weights. Therefore, Multi-CSAR has been developed to further overcome these limitations of Multi-CAR. In principle, the main steps of the algorithm in

Multi-CSAR is the same as those in Multi-CAR, except that Multi-CSAR utilizes CSAR [14], instead of CAR [13], to compute the single reference-derived scaffolding result for the target draft genome, and also designs a *sequence identity-based weighting scheme* to automatically derive the weights of all the reference genomes. CSAR actually is an improved version of CAR and their main difference in usage is that the reference genome used by CAR needs to be complete, but the one used by CSAR can be incomplete.

3.1 Algorithm of multi-CSAR

Suppose that T denotes a target draft genome with n contigs $c_1, c_2, \dots, c_n$ and $R_1, R_2, \dots, R_k$ denote k reference genomes with weights $w_1, w_2, \dots, w_k$, respectively. Contigs actually are fragmented linear DNA sequences with two *extremities*, called *head* and *tail*, respectively. Multi-CSAR performs the following steps to scaffold the contigs in the target genome T using the multiple reference genomes $R_1, R_2, \dots, R_k$. First, Multi-CSAR utilizes CSAR to obtain a single reference-derived scaffold S_i of T based on each R_i, where $1 \leq i \leq k$. Second, Multi-CSAR constructs a *contig adjacency graph* $G = (V, E)$ such that there are two vertices c_j^h and c_j^t for representing the head and tail of each contig c_j, respectively, and there also is an edge for linking any two vertices if they are the extremities coming from the different contigs. An edge in E is said to be *supported* by a reference genome R_i if its two vertices are adjacent extremities from two distinct but continuous contigs in scaffold S_i. If an edge in E is supported by several reference genomes at the same time, then this edge receives a weight equal to the sum of the weights of all these supporting reference genomes. However, if an edge in E is not supported by any reference genome, then it has a weight of zero. Third, Multi-CSAR utilizes the Blossom V [20] to find a maximum weighted perfect matching M in G, where a subset of edges in G is called a *perfect matching* if every vertex in G is incident to exactly one edge in this subset. Let $C = \left\{ \left(c_j^t, c_j^h \right) \mid 1 \leq j \leq n \right\}$ and M' denote a subset of M (i.e., $M' \subseteq M$) with the minimum weight such that there is no cycle in $M' \cup C$. Finally, Multi-CSAR makes use of the edge connections in M' to scaffold the contigs of T. **Figure 1** displays an example for illustrating how the algorithm of Multi-CSAR works.

Note that CSAR was developed based on on a near-linear time algorithm [21] and Blossom V based on an $\mathcal{O}(n^4)$-time algorithm [20], where n is the number of vertices in a graph. Therefore, all the steps in the Multi-CSAR algorithm described previously can be implemented in polynomial time. In addition, Multi-CSAR utilizes the following *sequence identity-based weighting scheme* to automatically compute the weights $w_1, w_2, \dots, w_k$ of the k reference genomes. First, Multi-CSAR applies either NUCmer or PROmer for identifying those *sequence markers* that actually are aligned regions between the target genome T and each reference genome R_i, where $1 \leq i \leq k$. Note that both NUCmer and PROmer come from the MUMmer package [22]. The main difference between NUCmer and PROmer is that the former finds the sequence markers directly on input DNA sequences, while the latter recognizes them on the six-frame protein translation of the input DNA sequences. Suppose that there are τ sequence markers, say $m_1, m_2, \dots, m_\tau$, between T and R_i, and $L(m_j)$ and $I(m_j)$ are used to denote the alignment length of each m_j and its percent identity, respectively. Next, Multi-CSAR calculates the *weight* of each reference genome R_i by the formula $w_i = \sum_{j=1}^{\tau} L(m_j) \times I(m_j)$. The principle of the sequence identity-based weighting scheme is that the more similar the reference genome R_i is to the target genome T, the more weight R_i receives.
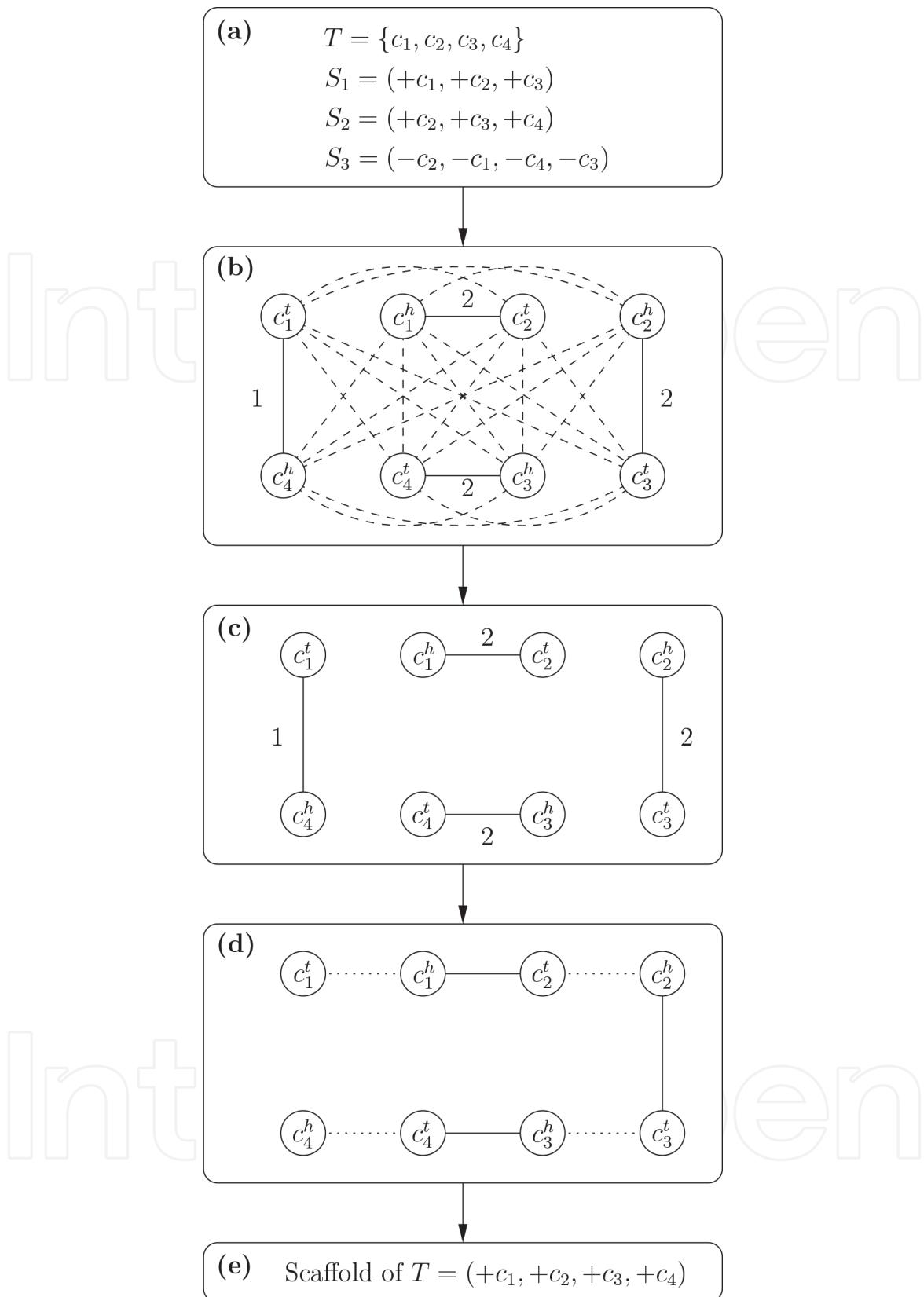
Figure 1.
Schematic workflow of multi-CSAR: (a) a target genome $T = \{c_1, c_2, c_3, c_4\}$ and three single reference-derived scaffolds $S_1 = (+c_1, +c_2, +c_3)$, $S_2 = (+c_2, +c_3, +c_4)$ and $S_3 = (-c_2, -c_1, -c_4, -c_3)$ that are supposed to be computed by applying CSAR on three reference genomes R_1, R_2 and R_3, respectively, with $w_1 = w_2 = w_3 = 1$. (b) The contig adjacency graph G constructed by using S_1, S_2 and S_3, where zero-weighted edges are denoted by dashed lines. (c) A perfect matching with maximum weight $M = \left\{ \left(c_1^h, c_2^t\right), \left(c_2^h, c_3^t\right), \left(c_3^h, c_4^t\right), \left(c_4^h, c_1^t\right) \right\}$ derived by applying Blossom V on G. (d) $M' = \left\{ \left(c_1^h, c_2^t\right), \left(c_2^h, c_3^t\right), \left(c_3^h, c_4^t\right) \right\}$ is obtained by removing edge $\left(c_4^h, c_1^t\right)$ with minimum weight from M such that $M' \cup C$ contains no cycles, where the dotted lines denote the edges in C. (e) The final scaffold $(+c_1, +c_2, +c_3, +c_4)$ of T constructed based on the edge connections in M'.

3.2 Usage of multi-CSAR

Currently, Multi-CSAR offers a web server[1] with an easy-to-operate interface (see **Figure 2**) to the users. To run Multi-CSAR, the users first need to upload a target genome and one or more reference genomes in multi-FASTA format. If needed, the users can click the "plus" (respectively, "minus") button to add (respectively, remove) a reference genome field. Second, the users can determine whether or not to utilize the sequence identity-based weighting scheme provided by Multi-CSAR for automatically calculating the weights of reference genomes. If the weighting scheme is not used, then the weights of all the reference genomes are defaulted to one. Third, the users can choose either NUCmer or PROmer to identify sequence markers between the target genome and each of the reference genomes. Fourth, the users can enter an email address, which is optional, if they would like to run Multi-CSAR in a batch way. When running Multi-CSAR in this batch way, the users will be notified of the scaffolding result via email when the submitted job is finished by the web server of Multi-CSAR.

Multi-CSAR outputs its scaffolding results in four tab pages: (a) input data & parameters, (b) Circos plot validation, (c) dotplot validation, and (d) scaffolds of target. In the "Input data & parameters" page (see **Figure 3** for an example), Multi-CSAR simply shows the information of the input target and reference genomes, the user-specified program (either NUCmer or PROmer) for identifying their sequence markers, and whether the weighting scheme of reference genomes is used or not. By clicking on the links of the target and reference genomes in this page, Multi-CSAR

Figure 2.
Interface of multi-CSAR web server.

---

[1] The web server of Multi-CSAR is available at http://genome.cs.nthu.edu.tw/Multi-CSAR/.

Figure 3.
A display of the "Input data & parameters" tab page.

will also display their input DNA sequences. By clicking on the link "Dotplot against target genome" on the reference genomes, Multi-CSAR will display a *dotplot* that allows the users to visually inspect sequence markers shared between un-scaffolded target genome and a reference genome. In the dotplot (see **Figure 4** for an instance), the un-scaffolded target genome and a selected reference genome are represented on the *y* and *x* axes, respectively. Note that the contigs and scaffolds in the dotplot are separated by horizontal and vertical dashed lines. Moreover, each forward (respectively, reverse) sequence marker is shown by a red (respectively, blue) line and its begin and end are represented by two unfilled circles. The users can sort the contigs of the input target genome based on their sizes by clicking on
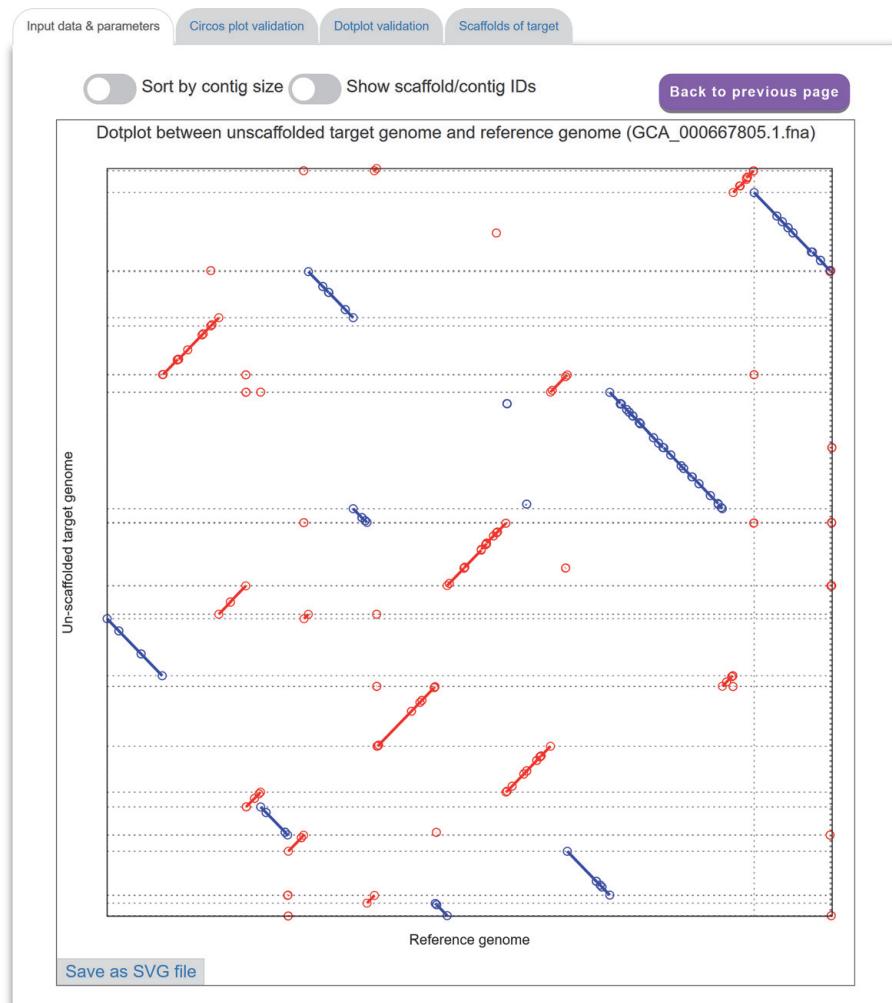
Figure 4.
A display of a dotplot between un-scaffolded target genome and a reference genome.

the toggle switch "Sort by contig size". The users also can show or hide the IDs of contigs and scaffolds used in Multi-CSAR by using the toggle switch "Show scaffold/contig IDs." The format of contig (respectively, scaffold) IDs begins with three-letter prefix CTG (respectively, SCF) followed by an underscore (_) and at least one digit (e.g., CTG_1 and SCF_1). In addition, the users can click the "Save as SVG file" button to download a copy of the dotplot in scalable vector graphics (SVG) format.

In the "Circos plot validation" page, (see **Figure 5** for an example), Multi-CSAR displays its total running time, as well as its scaffolding result by a Circos plot between scaffolded target genome and all reference genomes. In the initial Circos plot, the scaffolds of target genome (displayed in purple) and all the reference genomes (displayed in other colors) are arranged in a circle with the inner links connecting corresponding sequence markers between the target genome and each of reference genomes. The color of an inner link comes from the reference genome it connects. In the Circos plot, the number of crossing inner links can be viewed as a
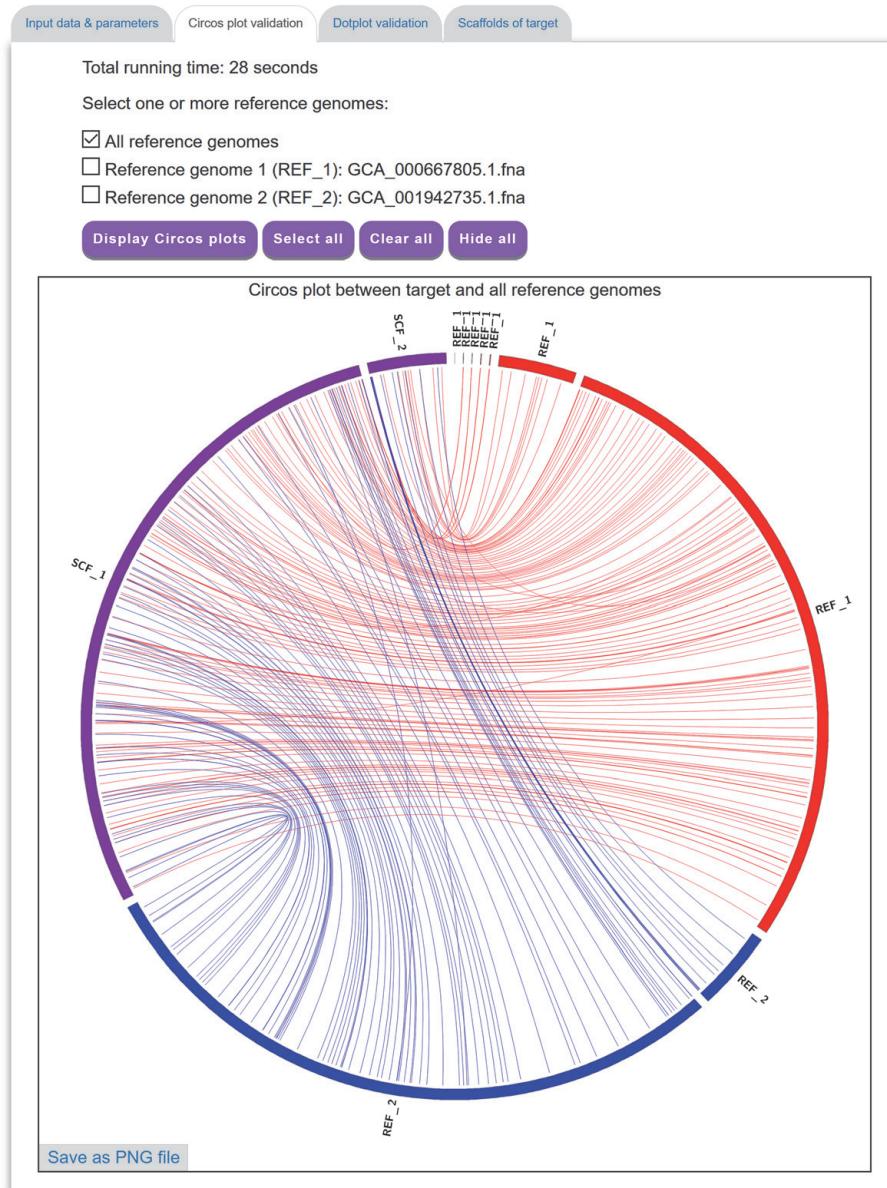
Figure 5.
*A display of a Circos plot between scaffolded target genome and all reference genomes.*
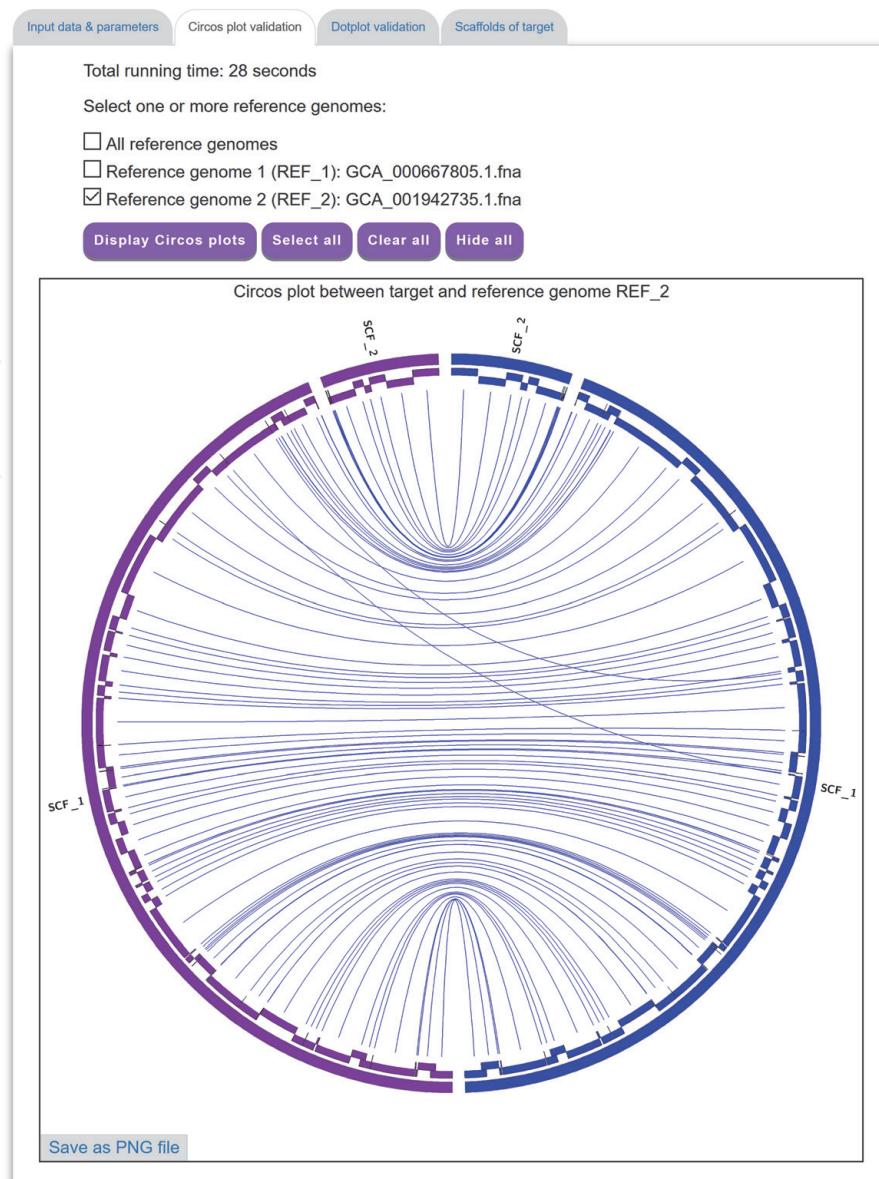
Figure 6.
A display of a Circos plot between scaffolded target genome and a selected reference genome, where the sequence markers are arranged in alternating layers along the two-layer inner circle.

accuracy measure for a scaffolding result. That is, if the contigs of the target genome are scaffolded well according to a reference genome, the number of crossing inner links between them should be low. For this purpose, Multi-CSAR allows the users to select any reference genome (by clicking the checkbox next to it) from the top of the tab page to display (by clicking the "Display Circos plot" button) its Circos plot against the scaffolded target genome (see **Figure 6** for an instance). In this Circos plot, the inner circle displays the sequence markers shared between the target genome and the selected reference genome. As demonstrated in **Figure 6**, the Circos plots of the scaffolding result are convenient and helpful for the users to visually validate whether the contigs of the target genome are properly scaffolded according to the reference genomes, as well as to visually identify whether there are any genome rearrangements between the scaffolded target and reference genomes. In addition, Multi-CSAR allows the users to the Circos plots of the scaffolds in portable network graphics (PNG) format by clicking the "Save as PNG file" button.

In the "Dotplot validation" page (see **Figure** 7 for an example), Multi-CSAR displays its its scaffolding result by a dotplot between the scaffolded target genome and a selected reference genome (the default is the first reference genome). In fact, the matched sequence regions of sequence markers should be displayed from the bottom left to the top right in the dotplot (as shown in **Figure** 7) or from the top left to the bottom right, if the contigs from the target genome are scaffolded perfectly based on the selected reference genome. Showing the scaffolding result in the dotplot display is another way to conveniently help the users to visually verify whether the contigs of the target genome are scaffolded properly based on the reference genomes or not. The users can click the "Save as PNG file" button to download the dotplot of a scaffold in portable network graphics (PNG) format.

In the "Scaffolds of target" page (see **Figure 8** for an instance), Multi-CSAR displays its scaffolding result in tabular format for the purpose of allowing the users to view the scaffolds of the target genome in detail. The scaffolds in the table are sorted according to their sizes, which equals to the sum of contig sizes. In each scaffold, the ordered contigs, as well as their orientations (forward orientation denoted by 0 and reverse orientation by 1), sequences and lengths, are listed in a table. The users can click on the "Download scaffolds (.txt)" and "Download scaffolds (.csv)" buttons to download the scaffolds of the target genome in the tab-delimited text format and comma-delimited CSV format, respectively. In addition, the users can click on the "Download sequences" button to download the scaffold sequences in the text format, in which the sequences of contigs are separated by 100 Ns if they belong to the same scaffold.
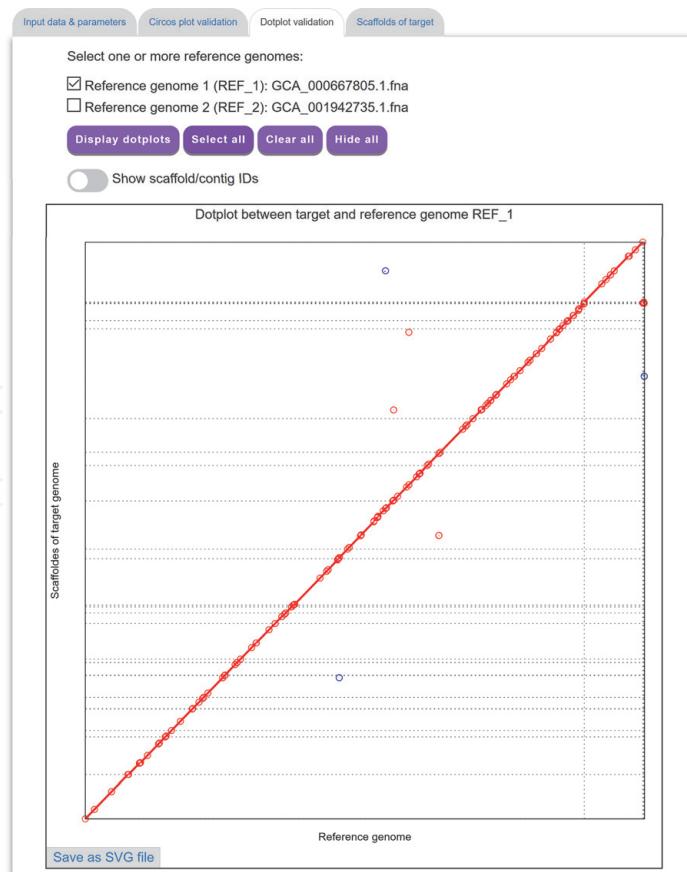

Figure 7.
A display of the "Dotplot validation" tab page.
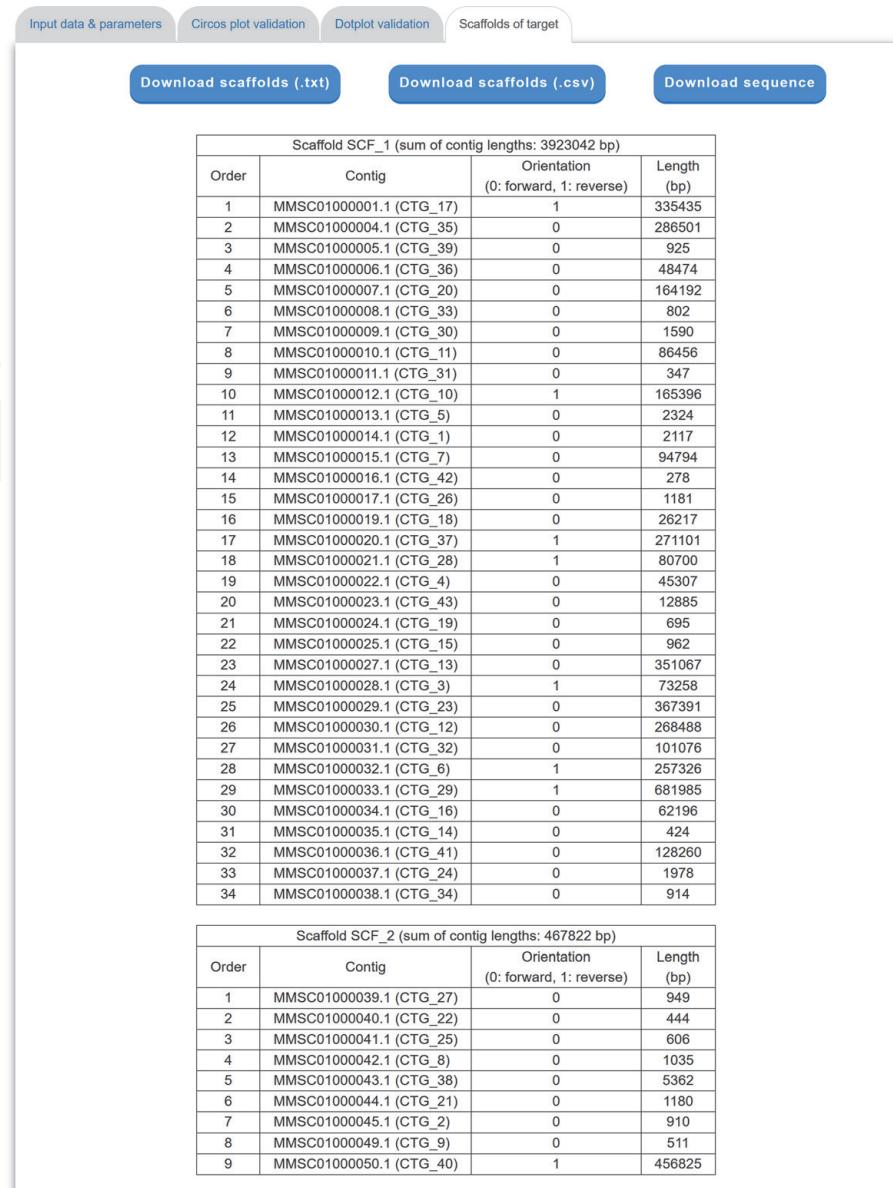
Figure 8.
A display of the "Scaffolds of target" tab page.

4. Results and discussion

4.1 Testing datasets

The three multiple reference-based scaffolders Multi-CSAR, Ragout (version 1.0) and MeDuSa (version 1.6), we introduced in this chapter, were tested on a benchmark of five real bacterial datasets as shown in **Table 1**. In fact, these five testing datasets were originally prepared by Bosi et al. when they studied MeDuSa [16]. Basically, each testing dataset consists of a target draft genome to be scaffolded and two or more reference genomes that can be either complete or incomplete.

4.2 Evaluation metrics

For each testing dataset, Bosi et al. [16] also provided a *reference order* for the contigs of the target genome that can be used a truth standard to evaluate the

Organism	No. of replicons	No. of contigs	No. of references	Genome size (Mbp)	GC %
B. cenocepacia j2315	4	1223	4	8.05	65.9
E. coli K12	1	451	25	4.64	50.8
M. tuberculosis	1	116	13	4.41	65.6
R. sphaeroides 2.4.1	7	564	2	4.60	67.4
S. aureus	3	170	35	2.90	32.0

Table 1.
Summary of the five testing datasets.

multiple reference-based scaffolders. The evaluation metrics of the scaffolders include sensitivity, precision, *F-score*, genome coverage, NGA50, scaffold number and running time. Basically, sensitivity, precision and *F-score* are used to estimate the scaffold accuracy, genome coverage to estimate the scaffold coverage, and NGA50 and scaffold number to estimate the scaffold contiguity. Below, we introduced their detailed definitions.

Note that if any two contigs in a scaffold appear in continuous order and correct orientation in the reference order, then they are viewed as a *correct* join. Let S denote the result obtained by applying a scaffolder on a target genome T and P denote the number of all contig joins in the reference order. The number of the correct contig joins in S is then called as *true positive* (TP) and the number of the others (i.e., incorrect joins) as *false positive* (FP). In addition, the *sensitivity* of S is defined as TP/P, its *precision* as TP/(TP + FP), and its *F-score* as $(2 \times \text{sensitivity} \times \text{precision})/(\text{sensitivity} + \text{precision})$. Actually, *F-score* is a balanced measure between sensitivity and precision and *F-score* is high only when both sensitivity and precision are high.

Suppose that the target genome T contains only circular DNAs and C is a contig in S. If the both sides of C are joined correctly with two contigs, then the whole length of C will be counted in the genome coverage that will be defined later. If exactly one side of C is joined correctly with one contig, then half of the whole length of C will be counted. If the both sides of C are joined incorrectly with two contigs, then the whole length of C will be ignored. Based on the above discussion, the *genome coverage* of S is defined to be the ratio of the sum of the contig lengths counted according to the above-mentioned rules to the sum of all contig lengths. On the other hands, suppose that there are linear DNAs in the target genome T. Then in the reference order of each linear DNA, the first and last contigs have just one neighbor contig and thus only half of their lengths will be counted in the calculation of the genome coverage if these two contigs are correctly joined with neighbor contigs.

The NGA50 value of S is computed as follows [23]. First, the scaffolds of S are aligned with the complete sequence of the target genome T to find the mis-assembly breakpoints. Second, the scaffolds of S are broken at the mis-assembly breakpoints and their unaligned regions are also removed. Finally, the NGA50 value is equal to the NG50 value of the resulting scaffolds, which is the size of the shortest scaffold with longer and equal length scaffolds covering at least 50% of the target genome.

4.3 Comparison of multiple reference-based scaffolding results

All the three evaluated scaffolders Multi-CSAR, Ragout (version 1.0) and MeDuSa (version 1.6) were all run with their default parameters, except that a star

Scaffolder	Sen	Pre	*F*-score	Cov	NGA50	#Scaf	Time
Ragout	79.0	**92.5**	84.4	87.4	992,966	84	24.8
MeDuSa	78.2	81.9	80.0	83.3	671,001	26	3.8
Multi-CSAR (PROmer)	89.3	90.4	89.8	92.5	1,016,308	7	6.3
Multi-CSAR (NUCmer)	**89.6**	90.8	**90.2**	93.2	1,038,257	9	**1.7**

Table 2.
Average performance of the evaluated multiple reference-based scaffolders on the five testing datasets.

Scaffolder	Sen	Pre	*F*-score	Cov	NGA50	#Scaf	Time
Multi-CSAR (PROmer)	89.4	90.5	89.9	92.8	1,045,489	7	6.3
Multi-CSAR (NUCmer)	**89.9**	**91.3**	**90.6**	**93.5**	1,046,288	10	**1.7**

Table 3.
Average performance of multi-CSAR on the five testing datasets when using the sequence identity-based weighting scheme.

tree was used in Ragout to serve as the phylogenetic tree for each testing dataset because reliable phylogenetic trees were still unknown. **Table 2** displays their average performance results over the five bacterial datasets, by showing the values of sensitivity (Sen), precision (Pre), *F-score* and genome coverage (Cov) in percentage (%) and the size of NGA50 in base pairs (bp). In addition, **Table 2** shows the numbers of scaffolds computed by all evaluated scaffolders in the column '#Scaf' and their running times in minutes in the column 'Time'. The best result in each column of **Table 2** is shown in bold.

As shown in **Table 2**, Multi-CSAR running with NUCmer achieves the best sensitivity, *F-score*, genome coverage, NGA50 and running time, and the second best precision and scaffold number. On the other hands, Multi-CSAR running with PROmer has the best result in terms of scaffold number and the second best results in terms of sensitivity, *F-score*, genome coverage and NGA50. From the precision point of view, the performance of Ragout is the best among all the tested multiple reference-based scaffolders. However, the sensitivity of Ragout is substantially inferior to that of Multi-CSAR when either running with NUCmer or PROmer. This negative result also leads to that Ragout is much inferior to Multi-CSAR in the performance of *F-score*. Moreover, Ragout yields the worst results in terms of both scaffold number and running time. Compared Multi-CSAR and Ragout, MeDuSa gives the worst performance in sensitivity, precision, *F-score*, genome coverage and NGA50, although it has the second best performance in running time.

Table 3 shows the average performance results of Multi-CSAR on the five bacterial datasets when using the sequence identity-based weighting scheme, where the best performance in each column is also displayed in bold. As compared to the results of Multi-CSAR as shown in **Table 2**, several performance measures of Multi-CSAR can be further improved if it is run with the sequence identity-based weighting scheme of reference genomes, such as sensitivity, precision, *F-score*, genome coverage and NGA50.

5. Conclusions

Scaffolders are useful tools for sequencing projects to obtain more complete sequences of genomes being sequenced. In this chapter, we mainly introduced some state-of-the-art multiple reference-based scaffolders, such as Ragout, MeDuSa and

Multi-CSAR (improved extension of Multi-CAR), that can efficiently produce more accurate scaffolds of a target draft genome by referring to multiple complete and/or incomplete genomes of related organisms. By testing on five real prokaryotic datasets, Multi-CSAR outperforms Ragout and MeDuSa in terms of average sensitivity, precision, *F-score*, genome coverage, NGA50, scaffold number and running time. Currently, Multi-CSAR provides the users with a web interface that is intuitive and easy to operate. In addition, it displays its scaffolding result in a graphical mode that allows the users to visually validate the correctness of scaffolded contigs and in a tabular mode that allows the users to view the details of scaffolds.

Acknowledgements

Conflict of interest

The authors declare no conflict of interest.

Author details

Yi-Kung Shieh†, Shu-Cheng Liu† and Chin Lung Lu*
Department of Computer Science, National Tsing Hua University, Hsinchu, Taiwan

*Address all correspondence to: cllu@cs.nthu.edu.tw

† These authors are contributed equally.

IntechOpen

References

[1] Goodwin S, McPherson JD, McCombie WR. Coming of age: Ten years of next-generation sequencing technologies. Nature Reviews. Genetics. 2016;**17**:333-351

[2] Pop M. Genome assembly reborn: Recent computational challenges. Briefings in Bioinformatics. 2009;**10**: 354-366

[3] Mardis E, McPherson J, Martienssen R, Wilson RK, McCombie WR. What is finished, and why does it matter. Genome Research. 2002;**12**:669-671

[4] Nagarajan N, Cook C, Di Bonaventura M, Ge H, Richards A, Bishop-Lilly KA, et al. Finishing genomes with limited resources: Lessons from an ensemble of microbial genomes. BMC Genomics. 2010;**11**:242

[5] van Hijum SA, Zomer AL, Kuipers OP, Kok J. Projector 2: Contig mapping for efficient gap-closure of prokaryotic genome sequence assemblies. Nucleic Acids Research. 2005;**33**:W560-W566

[6] Richter DC, Schuster SC, Huson DH. OSLay: Optimal syntenic layout of unfinished assemblies. Bioinformatics. 2007;**23**:1573-1579

[7] Assefa S, Keane TM, Otto TD, Newbold C, Berriman M. ABACAS: Algorithm-based automatic contiguation of assembled sequences. Bioinformatics. 2009;**25**:1968-1969

[8] Rissman AI, Mau B, Biehl BS, Darling AE, Glasner JD, Perna NT. Reordering contigs of draft genomes using the mauve aligner. Bioinformatics. 2009;**25**:2071-2073

[9] Husemann P, Stoye J. r2cat: Synteny plots and comparative assembly. Bioinformatics. 2010;**26**:570-571

[10] Galardini M, Biondi EG, Bazzicalupo M, Mengoni A. CONTIGuator: A bacterial genomes finishing tool for structural insights on draft genomes. Source Code for Biology and Medicine. 2011;**6**:11

[11] Munoz A, Zheng C, Zhu Q, Albert VA, Rounsley S, Sankoff D. Scaffold filling, contig fusion and comparative gene order inference. BMC Bioinformatics. 2010;**11**:304

[12] Dias Z, Dias U, Setubal JC. SIS: A program to generate draft genome sequence scaffolds for prokaryotes. BMC Bioinformatics. 2012;**13**:96

[13] Lu CL, Chen KT, Huang SY, Chiu HT. CAR: Contig assembly of prokaryotic draft genomes using rearrangements. BMC Bioinformatics. 2014;**15**:381

[14] Chen KT, Liu CL, Huang SH, Shen HT, Shieh YK, Chiu HT, et al. CSAR: A contig scaffolding tool using algebraic rearrangements. Bioinformatics. 2018;**34**:109-111

[15] Kolmogorov M, Raney B, Paten B, Pham S. Ragout: A reference-assisted assembly tool for bacterial genomes. Bioinformatics. 2014;**30**:i302-i309

[16] Bosi E, Donati B, Galardini M, Brunetti S, Sagot MF, Lio P, et al. MeDuSa: A multi-draft based scaffolder. Bioinformatics. 2015;**31**:2443-2451

[17] Chen KT, Chen CJ, Shen HT, Liu CL, Huang SH, Lu CL. Multi-CAR: A tool of contig scaffolding using multiple references. BMC Bioinformatics. 2016; **17**:469

[18] Chen KT, Shen HT, Lu CL. Multi-CSAR: A multiple reference-based contig scaffolder using algebraic rearrangements. BMC Systems Biology. 2018;**12**:139

[19] Pagani I, Liolios K, Jansson J, Chen IMA, Smirnova T, Nosrat B, et al. The genomes OnLine database (GOLD) v.4: Status of genomic and metagenomic projects and their associated metadata. Nucleic Acids Research. 2012;**40**:D571-D579

[20] Kolmogorov V. Blossom V: A new implementation of a minimum cost perfect matching algorithm. Mathematical Programming Computation. 2009;**1**:43-67

[21] Lu CL. An efficient algorithm for the contig ordering problem under algebraic rearrangement distance. Journal of Computational Biology. 2015;**22**:975-987

[22] Kurtz S, Phillippy A, Delcher AL, Smoot M, Shumway M, Antonescu C, et al. Versatile and open software for comparing large genomes. Genome Biology. 2004;**5**. Available from: https://genomebiology.biomedcentral.com/articles/10.1186/gb-2004-5-2-r12

[23] Gurevich A, Saveliev V, Vyahhi N, Tesler G. QUAST: Quality assessment tool for genome assemblies. Bioinformatics. 2013;**29**:1072-1075