Phylogenetic Comparative Assembly

Peter Husemann[1,2] and Jens Stoye[1]

[1] AG Genominformatik, Technische Fakultät, Bielefeld University, Germany
[2] International NRW Graduate School in Bioinformatics and Genome Research
Bielefeld University, Germany
{phuseman,stoye}@techfak.uni-bielefeld.de

Abstract. Recent high throughput sequencing technologies are capable of generating a huge amount of data for bacterial genome sequencing projects. Although current sequence assemblers successfully merge the overlapping reads, often several contigs remain which cannot be assembled any further. It is still costly and time consuming to close all the gaps in order to acquire the whole genomic sequence. Here we propose an algorithm that takes several related genomes and their phylogenetic relationships into account to create a contig adjacency graph. From this a layout graph can be computed which indicates putative adjacencies of the contigs in order to aid biologists in finishing the complete genomic sequence.

1 Introduction

Today the nucleotide sequences of many genomes are known. For the first available genomic sequences, the process of obtaining the sequence was costly and tedious. The most common approach for de-novo genome sequencing is *whole genome shotgun sequencing* [1, 2]. Here, the genome is fragmented randomly into small parts. Each of these fragments is sequenced, for example, with recent high throughput methods. In the next step, overlapping reads are merged with an *assembler* software into a contiguous sequence. However, instead of the desired one sequence of the whole genome, often many *contigs* remain, separated by gaps. The main reasons for these gaps are lost fragments in the fragmentation phase and repeating sequences in the genome. In a process called *scaffolding*, the relative order of the contigs as well as the size of the gaps between them is estimated. In a subsequent *finishing* phase the gaps between the contigs are closed with a procedure called *primer walking*. For the ends of two estimated adjacent contigs, specific primer sequences have to be designed that function as start points for two polymerase chain reactions (PCRs) for Sanger sequencing. These PCRs ideally run towards each other until the sequences overlap. To close a gap completely, new primer pairs have to be generated again and again since the maximum read length for Sanger sequencing is restricted. This makes the process expensive and work intensive. It is thus advisable to reduce the pairs of contigs that have to be considered. If no further information about the order of the contigs is given there are $\mathcal{O}(n^2)$ possibilities for n contigs to apply primer

walking. If the order is known, it suffices to do $\mathcal{O}(n)$ primer walks to fill the gaps.

An algorithm that estimates a reasonable order for the contigs is thus a good help for sequencing projects. The estimation is usually based on the sequences of closely related species that are assumed to have a high degree of synteny. A few tools have been developed which use one or several related reference genomes to devise an ordering of the contigs: OSLay [3] for example takes a set of BLAST [4] matches between the contigs and a reference sequence and computes from this a layout for the contigs. The algorithm minimizes the height differences of so-called local diagonal extensions, which are basically matches from the border of a contig to the reference sequence. The program Projector2 [5] is a web service that maps contigs on a reference genome, based on experimentally validated rules, and automatically designs suitable primer sequences. Zhao et al. [6] developed an algorithm to integrate the information of multiple reference genomes. For the ordering, the enhanced ant colony optimization algorithm PGA (pheromone trail-based genetic algorithm) is used.

Our work commenced when analyzing data from inhouse sequencing projects for different species of the *Corynebacteria* genus, where we observed several aspects making it hard to find an ordering of the contigs. Zhao et al. show that poor sequence coverage can be overcome by using multiple reference genomes, but problematic for this approach are major rearrangements in the genomic sequences of more distantly related species. Another challenge are repeating regions in the sequence of the newly sequenced genome. We developed an algorithm that uses the information of all similar regions between a set of contigs and several related reference genomes to estimate an ordering of the contigs. The novel idea is to incorporate the phylogenetic distance of the species in order to alleviate the impact of rearrangements to the ordering. While generating one 'optimal' order of the contigs is the predominant approach to aid the closure of gaps, we propose a more flexible format to describe contig adjacencies that is also capable of dealing with repeating contigs.

The algorithm we present here is based on a simple data structure, the *contig adjacency graph* that is introduced in Sect. 2. There we also give an optimal solution for finding a linear ordering of the contigs using this graph. However, a linear ordering is not sufficient to reflect all relations of real contig data. Therefore we propose a heuristic in Sect. 3 by which the most promising, but not necessarily unique, adjacencies are revealed in a *layout graph*. Sect. 4 shows the results of applying our method to real sequencing data and compares these with a recent approach from the literature.

2 Phylogeny Based Contig Ordering Algorithm

A natural strategy to devise an 'optimal' linear ordering of the contigs, based on one or several related reference genomes, works in three steps: At first, all similar regions between the contigs and each reference genome are determined. Then a graph is created, containing edge weights that reflect the neighborhood

of the contigs. In the last step a weight maximizing path through the graph is calculated, which defines the desired order of the contigs. In the following, we describe these three steps in more detail, in particular in Sect. 2.2, we define a novel edge weight function that incorporates the phylogenetic distance of the involved species.

2.1 Matching Contigs Against a Reference

Let $\Sigma = \{A, C, G, T\}$ be the alphabet of nucleotides. We denote by Σ^* the set of all finite strings over Σ, by $|s| := \ell$ the *length* of string $s = s_1 \dots s_\ell$, $s \in \Sigma^\ell$, and by $s[i, j] := s_i \dots s_j$ with $1 \leq i \leq j \leq \ell$ the *substring* of s that starts at position i and ends at position j. Suppose we are given a set of contigs $\mathcal{C} = \{c_1, \dots, c_n\}$, $c_i \in \Sigma^*$ and a set of already finished reference genomes $\mathcal{R} = \{g_1, \dots, g_m\}$, $g_r \in \Sigma^*$. The relation of the reference genomes is given by a phylogenetic tree $\mathcal{T}$ which contains the evolutionary distances of the species. Note that the tree can be generated even if some genomes are not completely assembled yet, for example from 16S-rRNAs.

To infer information about the order and orientation of the contigs, these are mapped onto each reference genome by calculating local alignments. Let $s = c[s_b, s_e]$ be a substring of contig c and $t = g[t_b, t_e]$ be a substring of reference genome g. The tuple $m = \big((s_b, s_e), (t_b, t_e)\big)$ is called a *matching region* or simply *match* if s and t share sufficient sequence similarity. The *length* of a match, $|m| := t_e - t_b + 1$, is defined as the length of the covered substring in the reference genome. Sufficient sequence similarity can be defined, for example, by a BLAST hit of significance above a user-defined threshold. We, on the contrary, employ in our implementation the *swift* algorithm [7] for matching. It utilizes a q-gram index and provides for each match m the number of exactly matching q-grams, denoted as $\mathrm{qhits}(m)$, which can be used as a quality estimation for that match. Note that each contig can have several matches on a reference genome. For $s_b > s_e$ we define $c[s_b, s_e]$ to be the reverse complement of $c[s_e, s_b]$ and call m a *reverse match*. Further we assume w.l.o.g. that $t_b < t_e$ for all $g[t_b, t_e]$, otherwise we can replace the involved contig sequence by its reverse complement. For brevity of notation, m_i^r denotes a match between contig $c_i \in \mathcal{C}$ and reference genome $g_r \in \mathcal{R}$, and $\mathcal{M}_i^r = \{m_{i,1}^r, \dots, m_{i,s}^r\}$ denotes the (possibly empty) set of all such matches.

Each match $m_i^r = \big((s_b, s_e), (t_b, t_e)\big)$ implies a projection of the contig c_i onto the reference genome g_r. The *projected contig* $\pi(m_i^r) = \big((t_b - s_b),\ (t_e + |c_i| - s_e)\big)$ refers to the implied pair of index positions on g_r. For reverse complement matches, the projection can be defined similarly. Fig. 1 shows an example of two projected contigs as well as their distance, which will be defined next.

The *distance* of two projected contigs $\pi(m) = (t_b,\ t_e)$ and $\pi(m') = (t'_b,\ t'_e)$ is defined as follows:

$$d\big(\pi(m),\ \pi(m')\big) = \begin{cases} t'_b - t_e & \text{if } t_b < t'_b \\ t_b - t'_e & \text{if } t_b > t'_b \\ -\min\{|m|,\ |m'|\} & \text{if } t_b = t'_b \end{cases} .$$

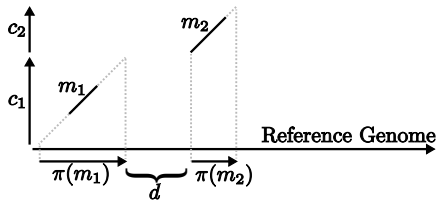


Fig. 1. Illustration of the projected contigs $\pi(m_1)$ and $\pi(m_2)$, which are based on their matches m_1 and m_2.

If the matches refer to different reference genomes, the distance of their projections is undefined. Note that the term distance is used here in the sense of displacement, d is *not* a metric in the mathematical sense. For example, d is negative if the projected contigs overlap.

2.2 Contig Adjacency Graph

In the following we define the edge-weighted *contig adjacency graph* $G_{\mathcal{C},\mathcal{R},\mathcal{T}} = (V, E)$ that contains for each contig $c_i \in \mathcal{C}$ two vertices: l_i as the *left connector* and r_i as the *right connector* of contig c_i. The set of vertices V is then defined as $V = L \cup R$, where $L = \{l_1, \ldots, l_n\}$ and $R = \{r_1, \ldots, r_n\}$.

The graph G is fully connected: $E = \binom{V}{2}$. We split these edges into two subsets: the *intra contig edges* $I = \{\{l_1, r_1\}, \ldots, \{l_n, r_n\}\}$ which connect for each contig its left and its right connector; and the set of *adjacency edges* $A = E \setminus I$ that connect the contigs among each other.

Now we define a weight function for the edges. For each intra contig edge $e \in I$ we set the weight $w(e) = 0$. For the remaining edges let $e = \{v_i, v_j\} \in A$ with $v_i \in \{l_i, r_i\}$ and $v_j \in \{l_j, r_j\}$ be an adjacency between contigs c_i and c_j. Then the weight of this adjacency edge is defined as

$$w(e) = \sum_{g_r \in \mathcal{R}} w_r(v_i, v_j)$$

where the function $w_r(v_i, v_j)$ defines a likelihood score for contigs c_i and c_j being adjacent, based on the matches to reference g_r. Moreover, the phylogenetic distance $d_{\mathcal{T}}$ between the contig's species and the reference genome's species is employed as a weight factor. Thus we define

$$w_r(v_i, v_j) = \sum_{m_i^r \in \mathcal{M}_i^r,\, m_j^r \in \mathcal{M}_j^r} s\Big(d\big(\pi(m_i^r),\, \pi(m_j^r)\big), d_{\mathcal{T}}\Big) \cdot \mathrm{qhits}(m_i^r) \cdot \mathrm{qhits}(m_j^r)$$

where d is the distance between the projected contigs and $s(d, d_{\mathcal{T}})$ is a suitably defined scoring function.

In order to define s we will give some further biological motivations. The scoring function s models the likelihood that two contigs are adjacent based on the distance d of their projected contigs. Projected contigs which are not adjacent should have a high distance and thus a low score. Adjacent contigs should have a distance close to zero and a high score. But on the one hand, the distance of the two projected contigs reaches positive values due to insertions in the reference's genome. On the other hand, the distances are negative if the projections are overlapping which is the case if there are insertions in the contigs' genome. Both cases can be seen in Fig. 2. Note that an insertion in the one genome corresponds to a deletion in the other.

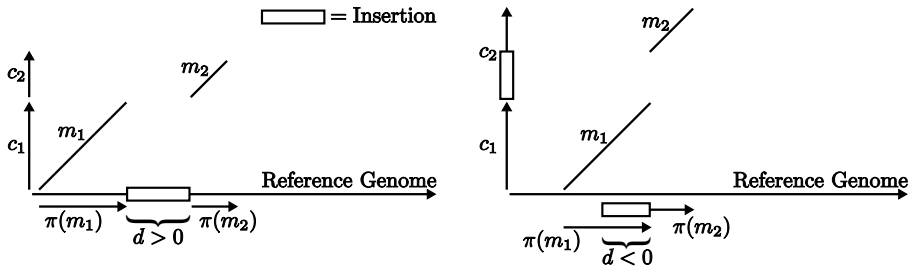


Fig. 2. An insertion in the reference genome leads to a positive distance (left side), whereas an insertion in a contig leads to a negative distance (right side).

A second important aspect that is included in our model are rearrangements between the related species, which can lead to misleading adjacencies of projected contigs. Assuming that between closer related species less rearrangements have taken place, we use the phylogenetic tree distance $d_{\mathcal{T}}$ to weight the match information. To model the mentioned two considerations, we use a Gaussian distribution with an expected value of zero:

$$s(d, d_{\mathcal{T}}) := \frac{1}{d_{\mathcal{T}} \cdot \sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{d}{d_{\mathcal{T}} \cdot \sigma})^2}$$

where σ is the standard deviation for the size of deletions or insertions. A higher tree distance $d_{\mathcal{T}}$ allows larger insertions and deletions, but scores the reliability of the matches to more distantly related genomes to a lesser degree.

However, this model neglects the fact that in the fragmentation phase, for example in parallel pyrosequencing, often fragments disappear such that there are no reads for this fragment. If a fragment is not sequenced, the same situation arises as if there is an insertion into the reference genome, which causes positive distances. To include this detail we use two superimposed Gaussian distributions for the scoring. The first distribution models insertions into the contigs and into the reference genome, the second models lost fragments during sequence

assembly. The influence of each model is determined by a weighting factor φ:

$$s(d, d_{\mathcal{T}}) := \frac{(1-\varphi)}{d_{\mathcal{T}} \cdot \sigma_1 \sqrt{2\pi}} e^{-\frac{1}{2}(\frac{d}{d_{\mathcal{T}} \cdot \sigma_1})^2} + \frac{\varphi}{\sigma_2 \sqrt{2\pi}} e^{-\frac{1}{2}(\frac{d-\mu}{\sigma_2})^2} \quad . \tag{1}$$

The expected value μ of the second Gaussian distribution is equal to the average size of the sequence fragments. The standard deviations σ_1 and σ_2 can be estimated from sequencing projects.

2.3 Finding a Tour Through the Graph

The contig adjacency graph with the described edge weights can be used to find a linear ordering of the contigs. This can be achieved by computing a tour through the graph that incorporates all contigs and maximizes the total weight. With minor enhancements of the graph, this becomes equivalent to finding a shortest Hamiltonian cycle.

The modifications are as follows: At first all edge weights have to be converted to distances. This is done by replacing each edge weight w by $m - w$ where m is the maximum weight in the graph. To ensure that each contig is incorporated exactly once, and only in one direction, we add an intermediate node between the left and the right connector of each contig. The modified graph is then defined as $G'_{\mathcal{C},\mathcal{R},\mathcal{T}} = (V', E')$ with $V' = V \cup \{v_i \mid 1 \leq i \leq n\}$ and $E' = A \cup \{\{l_i, v_i\}, \{v_i, r_i\} \mid 1 \leq i \leq n\}$. The distance of all edges that lead to an intermediate node v_i is set to 0. It is easy to see that a shortest Hamiltonian cycle in the modified graph defines an ordering as well as the orientation of all contigs, and thus any TSP algorithm can be used to find an optimal linear layout of the contigs.

3 Fast Adjacency Discovery Algorithm

As described in the previous section, a linear ordering of the contigs, which is optimal with respect to the adjacency edge weights, can be computed using a suitable optimization algorithm. However, our results on real data in Sect. 4 show that a linear order of the contigs is not necessarily possible, mainly due to arbitrary placement of repeated or rearranged regions. A method that provides a unique layout where possible, but also points out alternative solutions where necessary, may be more useful in practice. We present an approach following this overall strategy in this section.

The basis of our algorithm is a greedy heuristic for the TSP, known as the *multi-fragment heuristic* [8], that proceeds as follows: First the edges of the graph are sorted by increasing distance and then added in this order into an initially empty set of path fragments. Whenever an involved node would exceed the maximal degree of two, or if a path fragment would create a cycle, the edge is skipped. The only exception to the latter is a cycle of length n which constitutes the final Hamiltonian path.

This *best connection first* procedure creates multiple low distance path fragments which are merged sooner or later. We chose this approach because it

seems natural to incorporate those adjacencies first into an ordering that are most promising to be investigated for gap closure.

As already indicated, repeating or rearranged regions may prohibit an unambiguous linear ordering of the contigs. Repeating contigs create cycles in a possible path, and rearrangements can lead to conflicting adjacencies of a contig. To model both, we relax the constraints of the multi-fragment heuristic: First, we do not check for cycles, which permits repeating contigs to be incorporated adequately. Secondly, when inserting an edge, we allow one of the incident nodes, but not both, to exceed the degree of two, which allows to also include conflicting information into our layout. The result of this modified heuristic is a subgraph of the contig adjacency graph $L \subset G$ that we call the *layout graph*. The procedure to generate the layout graph is formally described in Algorithm 1. Note that the resulting layout graph is not necessarily connected.

Algorithm 1: Contig adjacency discovery algorithm

Input: set of contigs $\mathcal{C}$, set of related genomes $\mathcal{R}$, phylogenetic tree $\mathcal{T}$
Output: layout graph L of the contigs

create contig adjacency graph $G_{\mathcal{C},\mathcal{R},\mathcal{T}} = (V, I \cup A)$
create empty layout graph $L = (V_L, E_L)$ with $V_L = \emptyset$ and $E_L = \emptyset$
foreach *adjacency edge* $e = \{v_i, v_j\} \in A$, *sorted by decreasing weight* $w(e)$ **do**
    if $|V_L \cap \{v_i, v_j\}| < 2$ **then**
        $V_L = V_L \cup \{v_i, v_j\}$
        $E_L = E_L \cup e$
    end
end
$E_L = E_L \cup I$

The layout graph can be analyzed to make assumptions about repeating contigs and rearrangements. Conflicting edges can give hints about these two problems. The information about unambiguously incorporated contigs can be used to generate primer pairs for gap closure. Displaying also the ambiguities allows to investigate the conflicting connections further. Instead of pinning the result down to a single, possibly wrong, order of the contigs we prefer to output the best possibilities. Nonetheless it should be kept in mind that rearrangements can cause seemingly good adjacencies that do not belong to a correct layout.

4 Results

We tested our algorithm on real contig data of the *Corynebacteria* genus. For the genomic sequences of *C. aurimucosum* (unpublished), *C. urealyticum* [9], and *C. kroppenstedtii* [10] we obtained the finished genomic sequences as well as the underlying contig data from sequencing projects that were conducted at Bielefeld University. Additionally we chose four more publicly available *Corynebacteria*

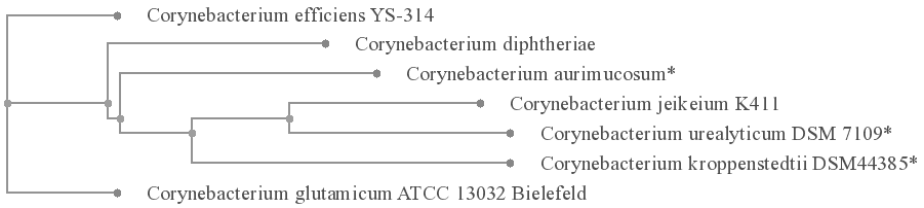
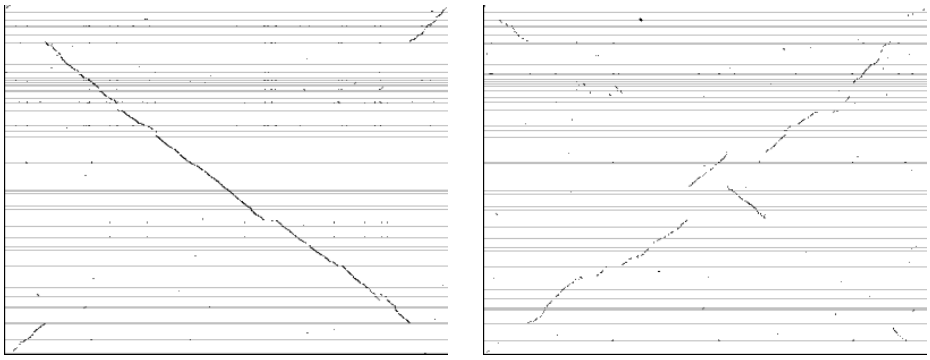


Fig. 3. Phylogenetic tree of the employed Corynebacteria. Contig data was available for the species marked with an asterisk (*). Tree calculated with EDGAR [13], image generated with PHY.FI [15]



(a) *C. jeikeium* (b) *C. aurimucosum*

Fig. 4. Synteny plots of the above species and the contigs of *C. urealyticum*. The latter are stacked on the vertical axis in reference order, separated by horizontal lines.

genomes, *C. diphtheriae*, *C. efficiens*, *C. glutamicum ATCC 13032 Bielefeld*, and *C. jeikeium* that we downloaded from NCBI [11, 12]. Figure 3 shows the evolutionary tree of all employed genomes. The tree was generated with the EDGAR framework [13] applying Neighbor Joining [14] to a set of *core* genes. As an illustration of the major rearrangements that happened between the employed species, Fig. 4 shows two example synteny plots. It is clearly observable that due to rearrangements a mapping of the contigs on the displayed related genomes would provide incorrect adjacencies of some contigs.

All finished sequences, except the genome of the contigs to be layouted, served as references to find a layout for the contig sets. As a standard of truth we computed for each set of contigs a *reference order* by mapping them onto the corresponding finished genome. We would like to note here that this reference order is not unique since many contigs contain or even consist of repeating regions which map non-uniquely to different locations.

We implemented our proposed algorithm in Java. The software *tree-cat* (tree based contig arrangement tool) contains a re-implementation of the fast local

alignment algorithm *swift* [7], the contig adjacency graph creation, a branch and bound exact TSP algorithm, and the fast layout graph heuristic described in Sect. 3. The software is open source (GPL) and available upon request.

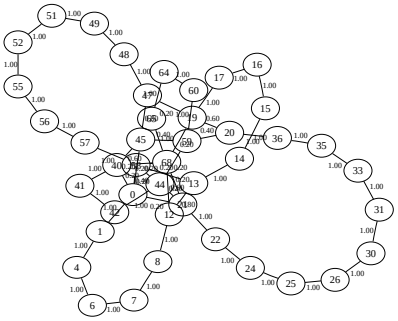
Input to *tree-cat* are the FASTA sequences of the contigs and of the related references as well as a phylogenetic tree in NEWICK format. Each reference can consist of several sequences, for example several chromosomes. When the algorithm is run, first all matches from the contigs to each reference are computed. For the following results, matches were considered to have a minimal length of 64 bases and a maximum error rate of 8%. The matches are cached which allows a visualization like in Fig. 4 and avoids a new computation if subsequent steps are re-run with different parameters. As the second step, after the matching, the contig adjacency graph is constructed as defined in Sect. 2.2. The following (empirically estimated) parameters were used for the scoring function (1) to compute the results: The standard deviation of the insertion/deletion size was set to $\sigma_1 = 10\,000$ bases and the expected lost fragment size to $\mu = 2\,000$ bases with a standard deviation of $\sigma_2 = 1\,000$ bases. The lost fragment weighting factor φ was set to 0.1. In the last step, the computed adjacency graph is used to devise the contig layout graph which can then be visualized with the open source software package GraphViz [16].

We compared the output of our method with the results of applying PGA [6] to the same data sets. OSLay [3] and Projector2 [5] were not included in the comparison since they use only one single genome as reference. Due to space restrictions we give in Fig. 5 only the results for the set of *C. urealyticum* contigs, but the other results are comparable. The data set consists of 69 contigs with a total size of 2.3 Mb and six reference genomes which have a total size of 16.6 Mb. To compute the results, a sparcv9 processor operating at 750 MHz under Solaris was used. Figures 5(a) and 5(b) show two PGA results and Fig. 5(c) contains our resulting layout graph. The node labels of all graphs are the rank of the corresponding contig with respect to the reference order of the contigs. The correct path should therefore be $0, 1, \ldots, 68$.

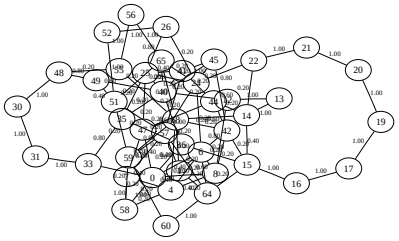
PGA uses BLAST to match the contigs on each genome. After that it computes five paths for the contigs that optimize a *fitness matrix* which is comparable to our contig adjacency graph. For that purpose a genetic algorithm is used, possibly giving different connections with each run. The connections of all five paths are included into the result and the edge weights give the percentage how often a connection occurred. Some nodes are missing in the first two graphs since PGA filters all contigs of length less than 3 500 bases.

In Fig. 5(a) PGA was applied with only one reference, the already finished genome of *C. urealyticum*, which ought to provide the 'perfect' information. The resulting graph shows that it is impossible to find a unique linear ordering of the contigs, even if the best possible reference sequence is available.

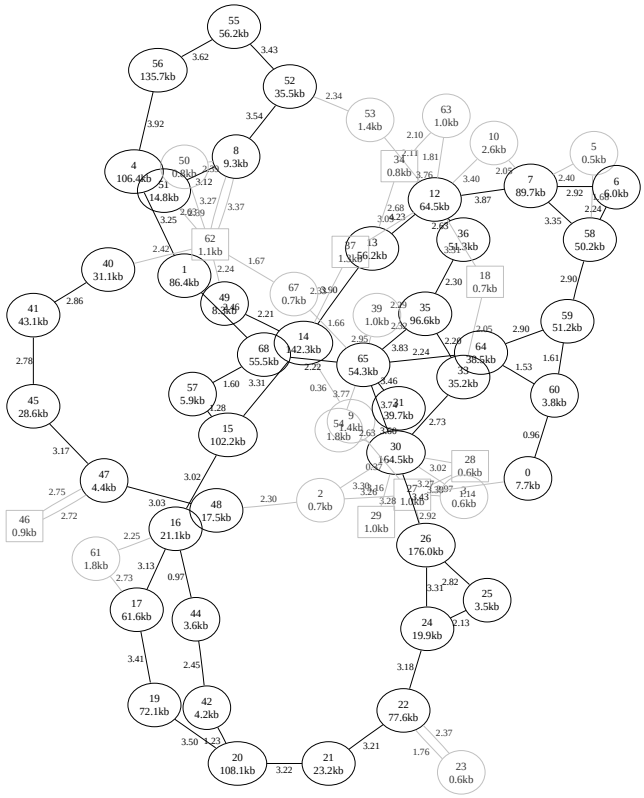
A comparison between PGA and *tree-cat* is given by the graphs in Figs. 5(b) and 5(c). Both are created with all other related genomes of our data set as reference genomes. For the graph in Fig. 5(b) the matching using BLAST needed 457 seconds and after that PGA took 161 seconds to compute its result, using



(a) PGA with perfect information.



(b) PGA using multiple references.



(c) *tree-cat* using multiple references. Contigs smaller than 3.5 kb have gray nodes, repeating contigs for which at least 95% of the sequence occurs more than once on a reference genome have rectangular nodes. Edge weights are given in logarithmic scale.

Fig. 5. *C. urealyticum* contig connections generated by PGA and *tree-cat*. The contig nodes are numbered in reference order.

the standard parameters given in [6]. Our algorithm, within *tree-cat*, required 95 seconds for the matching and about two seconds for the creation of the contig adjacency graph as well as the calculation of the layout graph, shown in Fig. 5(c). Although the reference order is not completely reliable, we used it as a rough estimation for the quality of the results. For each edge in the graphs we tested whether it is present in the reference order. PGAs graph contains 19 true positive connections and 75 false positive. *tree-cat* computed 27 correct and 70 false connections. This shows that our method achieved a better sensitivity (0.39 to 0.28) as well as a better precision (0.28 to 0.2), while being much faster. Additionally, PGA's result contains connections that are obviously incorrect, like, for example, placing contig 26 next to contig 52. Our approach does not show this connection. Manual inspection shows that this is due to the evolutionary distances that we incorporate for the edge scoring since phylogenetically closer genomes do not contain this adjacency. This is further supported by the fact that the connection is also not present when PGA uses the 'perfect' reference, see Fig. 5(a).

5 Conclusion

The contribution of our paper is twofold. On the one hand our results demonstrate that the common approach of searching one linear optimal ordering of the contigs is not feasible for real world data, due to repeating regions and rearrangements between the species. Therefore, we propose a more flexible output for the ordering of contigs and give an algorithm which generates such results.

On the other hand we developed a novel scoring function for the contig adjacency estimation based on multiple reference genomes that is biologically motivated in two ways: Firstly, it contains a sophisticated weighting scheme for the distances of projected contigs, and secondly it integrates the phylogenetic distances of the species to alleviate the effects caused by rearrangements.

A first evaluation of our algorithm shows that its implementation *tree-cat* is considerably faster than a recent approach from the literature while it is at the same time generating better results. We believe that with our approach of including phylogenetic information into the problem of contig layouting, we have gone one step further in using all available information for this important task within genome finishing.

Nevertheless, in sequencing projects, often additional information emerges which is not yet included in our approach. For example, information derived from mate pairs, fosmid libraries or radiation hybrid maps might give valuable hints on the orientation and the distance of contigs while not being biased by evolutionary events. Concerning the phylogenetic tree, rearrangements between the genomes were not predicted by the methods presented in this paper. This leads to ambiguous information for the ordering of contigs and thus to weak or misleading adjacency scores which need to be curated manually. A strategy for the discovery of rearrangements is thus desired in future work. Furthermore, due to horizontal gene transfer some regions of a genome can have different

evolutionary histories than others. Detecting such regions and treating them in a special way might be advisable in an even more sophisticated approach.

Acknowledgments

The authors wish to thank Christian Rückert, Susanne Schneiker-Bekel, and Andreas Tauch for the sequence data, Jochen Blom and Burkhard Linke for the phylogenetic tree, and Travis Gagie and Roland Wittler for helpful discussions.

References

1. Staden, R.: A strategy of DNA sequencing employing computer programs. Nucleic Acids Res. **6**(7) (1979) 2601–2610
2. Anderson, S.: Shotgun DNA sequencing using cloned DNase I-generated fragments. Nucleic Acids Res. **9**(13) (1981) 3015–3027
3. Richter, D.C., Schuster, S.C., Huson, D.H.: OSLay: optimal syntenic layout of unfinished assemblies. Bioinformatics **23**(13) (2007) 1573–1579
4. Altschul, S., Gish, W., Miller, W., Myers, E., Lipman, D.: Basic local alignment search tool. J. Mol. Biol. **215** (1990) 403–410
5. van Hijum, S.A.F.T., Zomer, A.L., Kuipers, O.P., Kok, J.: Projector 2: contig mapping for efficient gap-closure of prokaryotic genome sequence assemblies. Nucleic Acids Res. **33** (2005) W560–W566
6. Zhao, F., Zhao, F., Li, T., Bryant, D.A.: A new pheromone trail-based genetic algorithm for comparative genome assembly. Nucleic Acids Res. **36**(10) (2008) 3455–3462
7. Rasmussen, K.R., Stoye, J., Myers, E.W.: Efficient q-gram filters for finding all epsilon-matches over a given length. J. Comp. Biol. **13**(2) (2006) 296–308
8. Bentley, J.J.: Fast algorithms for Geometric Traveling Salesman Problems. Informs. J. Comp. **4**(4) (1992) 387–411
9. Tauch, A., et al.: The lifestyle of *Corynebacterium urealyticum* derived from its complete genome sequence established by pyrosequencing. J. Biotechnol. **136**(1-2) (2008) 11–21
10. Tauch, A., et al.: Ultrafast pyrosequencing of *Corynebacterium kroppenstedtii DSM44385* revealed insights into the physiology of a lipophilic corynebacterium that lacks mycolic acids. J. Biotechnol. **136**(1-2) (2008) 22–30
11. Wheeler, D.L., Chappey, C., Lash, A.E., Leipe, D.D., Madden, T.L., Schuler, G., Tatusova, T.A., Rapp, B.A.: Database resources of the national center for biotechnology information. Nucleic Acids Res. **28**(1) (2000) 10–14
12. Benson, D.A., Karsch-Mizrachi, I., Lipman, D.J., Ostell, J., Rapp, B.A., Wheeler, D.L.: Genbank. Nucleic Acids Res. **28**(1) (2000) 15–18
13. Blom, J., Albaum, S.P., Doppmeier, D., Pühler, A., Vorhölter, F.J., Goesmann, A.: EDGAR: A software framework for the comparative analysis of microbial genomes. BMC Bioinformatics (2009) to appear.
14. Saitou, N., Nei, M.: The neighbor-joining method: a new method for reconstructing phylogenetic trees. Mol. Biol. Evol. **4**(4) (1987) 406–425
15. Fredslund, J.: PHY.FI: fast and easy online creation and manipulation of phylogeny color figures. BMC Bioinformatics **7** (2006) 315
16. Gansner, E.R., North, S.C.: An open graph visualization system and its applications to software engineering. SPE **30** (1999) 1203–1233