Syntenic Layout of Two Assemblies of Related Genomes

Olaf Delgado Friedrichs [*] Aaron L. Halpern [†] Ross Lippert [‡]
Christian Rausch [§] Stephan C. Schuster [¶] Daniel H. Huson [‖]

Abstract: To facilitate research in comparative genomics, sequencing projects are increasingly aimed at assembling the genomes of closely related organisms. Given two incomplete assemblies of two related genomes, the question arises how to use the similarity of the two sequences to obtain a better ordering and orientation of both assemblies. In this paper, we formalize this question as the Optimal Syntenic Layout problem, show that it is in general NP-hard, but that it can be solved well in practice using an algorithm based on maximal graph matching. We illustrate the problem using different assemblies of two strains of *Bdellovibrio bacteriovorus*.

1 Introduction

To facilitate comparative studies, increasingly, genome sequences are assembled for closely related species or strains, often in parallel sequencing projects [KPEL03, EBR$^+$04]. Given two incomplete assemblies A and B of two related genomes G and H, the question arises how to use the similarity of the two sequences to obtain a (partial) ordering and orientation of each of the two assemblies. This *Optimal Syntenic Layout* problem is related to, but different from, the question of *aligning* two collections of sequences. In an alignment [NW70, SW81], the goal is to find a layout so that the matches between the two sequences form a "diagonal". However, we can expect that rearrangements of parts of G and H have occurred during evolution, and thus any such diagonal alignment would attempt to "undo" and thus mask these evolutionary events, which is undesirable.

In this paper we propose a formulation and algorithm for the *Optimal Syntenic Layout* problem that aims at finding a "syntenic layout" of two assemblies that maximize the number of pairs of extended "local diagonals". Our solution is based on the weighted graph matching problem. Our approach also addresses the important special case in which one of the two genomes is available as a complete sequence, and is used to "syntenically layout" the contigs of the other genome.

[*] `delgado@informatik.uni-tuebingen.de`, ZBIT, Tübingen University, Germany
[†] `ahalpern@tcag.org`, The Center for Advancement of Genomics, Rockville MD, USA
[‡] `lippert@math.mit.edu`, Department of Mathematics, M.I.T., Cambridge MA, USA
[§] `rausch@informatik.uni-tuebingen.de`, ZBIT, Tübingen University, Germany
[¶] `stephan.schuster@tuebingen.mpg.de`, MPI for Developmental Biology, Tübingen, Germany
[‖] `huson@informatik.uni-tuebingen.de`, ZBIT, Tübingen University, Germany

We illustrate the problem and our solution using data from two different sequencing projects, one on *Bdellovibrio bacteriovorus HD100* [RJR$^+$04] and the other on *Bdellovibrio bacterivorus HI100* [REB$^+$04], two different strains of the same species. We compare assemblies of both genomes at a number of different levels of sequencing coverage.

2 Sequence Assembly

The number of published genome sequences is rising daily. In most cases, these sequences are obtained using some variant of the "whole genome shotgun" approach.

In this approach, an *assembly* A of a target sequence G is obtained by randomly sampling many short *reads* of DNA from G, then sequencing these fragments and then assembling them into an approximation A of the target sequence. Depending on the size of G and the amount of resources available, such an assembly project can take a number of months. During this time, the level of sequence *coverage* will grow steadily. This is measured as the x*-fold coverage*, which specifies the average number of sequenced fragments that cover any fixed position in the target sequence.

For practical and statistical reasons [LW88], any such random sampling of fragments from a given target sequence G will miss some portions of G, giving rise to *sequencing gaps*. Clearly, these portions will be missing from the obtained assembly A, too, and they will cause it to break up in to multiple pieces of sequence, called *contigs*. Another cause for break-up are repeats in the genome, which are usually more difficult to assemble correctly. Obviously, the larger the x-fold coverage is, the smaller the number of contigs will be, and the longer these contigs will be. A value of $x = 12$ is considered desirable and an experimental study suggests that one needs at least $x = 5$, before reasonably long contigs can be obtained [MSD$^+$00].

An assembly A produced in this way consists of an unordered and unorientated collection of contigs. (Alternatively, we may also consider an assembly to consist of a set of *scaffolds*, that is, collections of contigs that are ordered and orientated with respect to each other using the "mate-pairs" that arise in "double-barreled shotgun sequencing" [MSD$^+$00].) The mapping problem is to find their correct *layout*, that is an *ordering and orientation* of the set of contigs based on features whose physical positions along the target sequence G are known.

In practice, during the course of an assembly project, a number of preliminary assemblies may be produced from the set of available fragments, that is, for different levels of x-fold coverage. After all fragments have been collected and a final assembly has been produced, then additional directed sequencing techniques must be applied to close any remaining gaps in the assembly, although, due to the relatively high cost of this, many genomes will remain unfinished.

3 Assemblies and Matches

Suppose we are given a target sequence G. An *assembly* $A = \{a_1, \ldots, a_p\}$ of G consists of a collection of *contigs* a_i that are putative sub-strings of G.

Let G and H be two genomes with assemblies $A = \{a_1, \ldots, a_p\}$ and $B = \{b_1, \ldots, b_q\}$, respectively. A local sequence comparison of the two assemblies [AGM$^+$90, DPCS02] gives rise to a collection of *matches* $M = \{m_1, m_2, \ldots, m_r\}$. A match m is specified as $(a, x_1, x_2, b, y_1, y_2, o)$, with $a \in A$, $1 \leq x_1 < x_2 \leq |a|$, $b \in B$, $1 \leq y_1 < y_2 \leq |b|$ and $o \in \{-1, +1\}$, where $|a|$ denotes the length of a. The interpretation of this is that m is a direct match between the interval with indices $[x_1, \ldots, x_2]$ in a and $[y_1, \ldots, y_2]$ in b, if $o = +1$, or a match in which the sequence of the second interval is reverse-complemented, if $o = -1$.

Suppose we are given two assemblies A and B of closely related genomes G and H, and a collection M of matches between them. How can we use the matches in M to order and orientate the contigs in A and B relative to one another?

We say that a match $m \in M$ is *informative*, if it is an overlap- or containment match, but not an end-to-end match. For our purposes, only informative matches are of interest, and this implies that the two assemblies should not be too *correlated*, that is, that contig boundaries should not coincide (that is, the contigs should not start and end at equivalent positions).

Consider an *original match* m between two segments g and h of the original target sequences G and H, respectively. Let A_g, or B_h, denote the set of all contigs in A, or B, that have a non-trivial intersection with g, or h, respectively. The original match m will give rise to a set of matches $M' \subseteq M$ between contigs in A_g and B_h. If the two assemblies are not too correlated, then the set of matches M' imposes a local layout of the contigs contained in A_g and the contigs contained in B_h, as this set of matches must form a single "local" diagonal.

4 The Optimal Syntenic Layout (OSL) Problem

Suppose we are given two assemblies $A = \{a_1, \ldots, a_p\}$ and $B = \{b_1, \ldots, b_q\}$ that are not too correlated, together with a set of matches $M = \{m_1, \ldots, m_r\}$. In the following, we formulate the problem of determining a layout of A that maximizes the number of pairs of extended local diagonals, for a fixed ordering of B. By switching the roles of A and B, it can be used to find an optimal layout for B, too. It will be apparent that the two problems are independent of each other and thus can be solved separately.

We visualize the data as partitioned into an $A \times B$ *comparison grid* Z, where the cell z_{ij} has width $|a_i|$ and height $|b_j|$. The set M_{ij} of all matches between a_i and b_j is displayed inside the cell z_{ij}. A match $m = (a_i, x_1, x_2, b_j, x_1, x_2, o)$ is shown as a *positive* line segment with 45° slope, if it is a direct match, i.e. if $o = +1$, and it is shown as a *negative* line segment with -45° slope, if $o = -1$, see Figure 1.

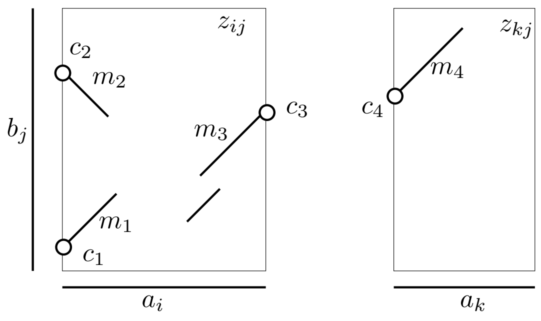


Figure 1: Here we show an example of cells z_{ij} and z_{kj} defined for contigs $a_i, a_k \in A$ and $b_j \in B$. Matches m_1, m_2 and m_4 define left-connectors c_1, c_2 and c_4, respectively, of z_{ij} and z_{kj}. The match m_3 gives rise to a right connector c_3 of z_{ij}. All matches except m_2 are direct. The two connectors c_3 and c_4 fit together and thus form a possible local diagonal extension, indicating that perhaps a_i should be immediately followed by a_k.

Such a line segment α defines a *left connector*, or *right connector*, of cell z_{ij}, if it touches or comes close to the left, or right, *side* of the cell, respectively. Such a connector $c = (y, w, o)$ has a *height* y, which is the position where α touches the corresponding side (or would touch, if it extended to the edge of the cell), a *weight* w, which is the length of α, and an *orientation* o that is ± 1, depending on whether α is a positive or negative line segment.

Consider two cells z_{ij} and z_{kj} contained in the same row. Let C_{ij}^{right} be the set of all right connectors associated with z_{ij} and C_{kj}^{left} be the set of all left connectors associated with z_{kj}. We say that two connectors $c = (y, w, o) \in C_{ij}^{right}$ and $c' = (y', w', o') \in C_{kj}^{left}$ form a *local diagonal extension*, if c' extends c, that is, if $y \approx y'$ and $o = o'$. We define the *weight* of such an extension as $w + w' - |y - y'|$, that is, the sum of weights of the involved matches, penalized by the difference of their heights. (In Section 6 we discuss this question in more detail.)

In practice, for each pair of cell-sides we form such extensions in a one-to-one fashion either greedily or by solving an instance of the maximum weight bipartite matching problem. Clearly, this definition carries over to the other combinations of sides, $\{left, left\}$, $\{left, right\}$ and $\{right, right\}$.

Given two columns a_i and b_j. We define the *score* of matching the ϵ-side of a_i to the δ-side of b_j as the sum of weights of all local diagonal extensions obtained for cells contained in the two columns, with $\epsilon, \delta \in \{left, right\}$. Alternatively, one can also use the maximum weight.

A *layout* of the assembly A is given by a signed permutation

$$\pi : (1, \ldots, p) \mapsto (\pm\pi(1), \ldots, \pm\pi(p)),$$

where $|\pm\pi(i)|$ denotes the position of the contig a_i in the ordering, and $\mathrm{sign}(\pm\pi(i))$ denotes its orientation.

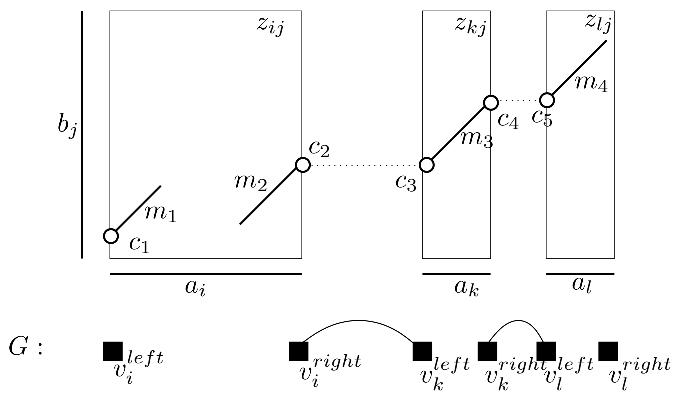


Figure 2: Here we show an example of three cells z_{ij}, z_{kj} and z_{lj}, all in the same row. Match m_2 in cell z_{ij} gives rise to a right connector c_2 that forms a local-diagonal extension with left connector c_3 of match m_3 of z_{kj}. The right connector c_4 of m_3 forms a local-diagonal extension with left connector c_5 of match m_4 in cell z_{lj}. In the corresponding graph G, we link nodes v_i^{right} and v_k^{left}, and we link nodes v_k^{right} and v_l^{left}.

> The *Optimal Syntenic Layout (OSL)* problem is to determine a layout π of A that maximizes the sum of scores of local diagonal extensions.

In terms of the comparison grid, this corresponds to finding an ordering and orientation of the columns (or rows) of the grid such that the sum of scores of pairs of adjacent column-sides (or row-sides, respectively) is maximized.

We now define a simple graph $\mathcal{G} = (V, E, \omega)$, with vertex set V, edge set E and an edge-weight function $\omega : E \rightarrow \mathbb{R}^{\geq 0}$.

For each column a_i of the grid Z we define two nodes, v_i^{left} and v_i^{right}, that correspond to the left and right sides of the column. Consider a pair of nodes v_i^{ϵ} and v_j^{δ} coming from two different columns a_i and a_j, with $\epsilon, \delta \in \{left, right\}$. We define an edge e between the two nodes v_i^{ϵ} and v_j^{δ}, if the score S of matching the ϵ-side of column a_i with the δ-side of column a_j is greater than zero. In this case, we set the weight $\omega(e)$ equal to S. See Figure 2.

Given a layout π of A. We say that an edge $e \in E$ between two nodes $v_i^{\epsilon} \in V$ and $v_j^{\delta} \in V$ is *realized*, if the ϵ-side of v_i is adjacent to the δ-side of v_j in the layout. In other words, we require that $|\pi(i) - \pi(j)| = 1$ and the orientations are appropriate so that the two sides under consideration are next to each other.

By definition of the graph $\mathcal{G}$, we have:

Lemma 4.1 *The OSL problem is equivalent to finding a layout π of A that maximizes the sum of weights of all realized edges in the graph $\mathcal{G}$.*

5 The OSL Problem is Hard

We have:

Lemma 5.1 *The OSL problem is NP-hard.*

Proof: We construct a reduction of the TSP problem with all distances in $\{1, 2\}$ [GJ79]. Given a set $C = \{c_1, \ldots, c_p\}$ of cities and a distance $D(i, j) \in \{1, 2\}$ for every pair of cities. Construct two assemblies, $A = \{a_1, \ldots, a_p\}$, where a_i represents city c_i, and $B = \{b_1, \ldots b_q\}$, with $q = 2p^2$. For any two numbers $1 \le i, j \le p$ set $k = (i-1)p + j \in \{1, \ldots, p^2\}$ and consider two cells z_{ik} and z_{jk}. Place a positive line segment that touches the right side of z_{ik} and another that touches the left side of z_{jk}, so that they form an extension of weight 1. Also, place two such segments touching the left side of z_{ik} and right side of z_{jk}, respectively. If $D(i, j) = 2$, then, additionally, set $k' = p^2 + k$ and place four such line segments in cells $z_{ik'}$ and $z_{jk'}$, too. Hence, if a_i and a_j are adjacent in the obtained layout, then 1 or 2 will be contributed to the score, depending on whether the corresponding edge in the input graph has weight 1 or 2, respectively. Given this construction, the set of optimal layouts of A corresponds precisely to the set of all optimal tours of the cities. $\square$

6 The Local Diagonal Layout Algorithm

The problem of finding a maximum weight matching in $\mathcal{G} = (V, E, \omega)$ can be solved efficiently [Ga73]. Consider such a matching $U \subseteq E$. For the following discussion, we add a set F of additional *contig* edges to the graph: For every pair of nodes $v_i^{left}, v_i^{right} \in V$ coming from the same contig a_i, we add an edge connecting these two nodes. Consider the graph $\mathcal{G}' = (V, U \cup F)$ containing only the matching edges and the contig edges. As the contig edges themselves form a matching, the graph $\mathcal{G}'$ consists only of chains and even-length cycles [Be76].

If the graph contains no cycles, then a solution of the OSL problem is obtained simply by laying out the contigs of A in any way that preserves the layout induced by the chains. If the graph contains one or more cycles, then each such cycle must be broken by removing a matching edge of minimum weight. In this way, each cycle loses less than half of its total weight. Because there may exist another solution that does not involve cycles, in the worst case, breaking cycles in this way may produce a solution that has only half the weight of an optimal solution. Here is a summary of the algorithm:

Algorithm 1 *[Local Diagonal Layout Algorithm]*

```
Input: Assemblies $A$ and $B$, and matches $M$
Output: A layout for $A$
Construct the graph $\mathcal{G} = (V, E, \omega)$, as described above
Compute a maximum matching $U \subseteq E$
```

```
Let $F$ be the set of all contig edges
Construct $\mathcal{G}' = (V, U \cup F, \omega)$
For each cycle $C$ in $\mathcal{G}'$:
    Delete the smallest weight edge in $C \cap U$
Greedily link all resulting chains into one chain visiting all nodes
Traverse the chain and report the resulting layout.
```

We have shown the following result:

Theorem 6.1 *Algorithm 1 computes a* 2*-approximation for the OSL problem.*

Note that if the graph G' does not contain cycles, then the obtained result is optimal. We suspect that such cycles will occur only rarely, and thus the algorithm should usually produce an optimal result in practice.

We now discuss two technical details that are important. First, in the above description, we suggest that each individual match $m \in M$ may give rise to a left or right connector of a cell. In practice, matches obtained by programs such as BLAST are usually short local matches that lie close to a common diagonal along which we really expect to see one long match. To address this, in our implementation of the algorithm, we use *summarized matches* that are obtained by greedily selecting certain diagonals onto which we project all original matches that lie within a window of width $w = 100$, say, around the diagonal. Also, in practice, we say that a summarized match comes close to the side of a cell, and thus gives rise to a connector, if it contains an original match that is within $d = 2000bp$, say, of the side.

Second, in the above description we suggest that the weight of a local diagonal extension should be $w + w' - |y - y'|$, where w and w' are the weights of the two involved (summarized) matches, and $|y - y'|$ is their height different. In practice, we obtain slightly better results using the following formula: $(w)^2 + (w')^2 - h|y - y'|^2$, with $h = 5$, say. We suspect that, as for most algorithms in bioinformatics, additional tuning of certain aspects of the algorithm will lead to a substantially better performance in practice, but this is beyond the scope of this paper.

7 Application to Different Assemblies of Two Strains of *Bdellovibrio Bacteriovorus*

We now illustrate the optimal syntenic layout problem and the application of our algorithm using data from two different sequencing projects, one on *Bdellovibrio bacteriovorus HD100* [RJR$^+$04] and the other on *Bdellovibrio bacterivorus HI100* [REB$^+$04].

The two complete sequences are approximately of the same length and a BLAST comparison of them displays a single main diagonal (consisting of many individual HSPs, of course) from end to end, thus indicating that there have been no significant inversions or transpositions since the separation of the two strains. We would like to emphasize, however, that the existence of such a single main diagonal is *not* a prerequisite for our method.

(For example, we have successfully applied our algorithm to contigs of *Neisseria meningitidis* serogroups A and B, whose comparison displays a diagonal that is broken by a major inversion, not shown here.)

We now compare assemblies of both genomes obtained at different levels of x-coverage. For each application of the algorithm, we report the number of *true positive-*, *false positive-* and *false negative extensions*, i.e. pairs of rows or columns that were correctly, incorrectly or not identified as being adjacent to each other in the true layouts of both assemblies, respectively.

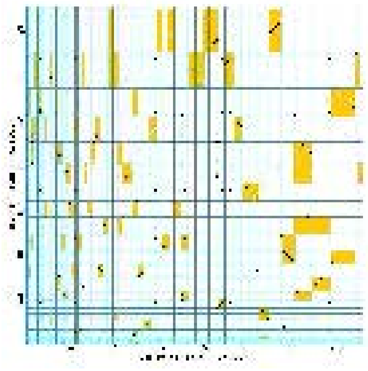
In Figure 3(a), we show two early assemblies of *Bdellovibrio bacteriovorus HI100*, consisting of 69 contigs, and *Bdellovibrio bacterivorus HD100*, consisting of 27 contigs, together with a collection of BLAST matches, obtained using an E-value threshold of 10^{-10}. Here, grey lines depict the cells of the comparison grid Z, whereas original BLAST matches appear as black diagonal lines. In Figure 3(b), we show the result of applying our algorithm to this data. Here, as in all remaining figures, black horizontal and vertical lines indicate where each run of local diagonal extensions (locally ordered and oriented rows or columns of the comparison grid) begin and end, and cells in which extendable diagonals were found are highlighted in grey. Here, the algorithm produced 46, or 9, true positive-, 2 or 0 false positive- and 17, or 14, false negative extensions in the x- or y-sequence, respectively.

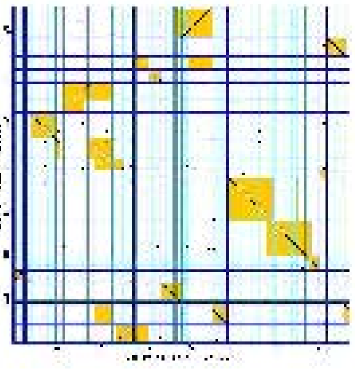
In Figure 3(c) and (d), we compare two further assemblies of the two strains with each other, producing 2, or 0, false positive- and 11, or 2, false negative extensions in (c), and 0, or 0, false positive- and 3, or 2, false negative extensions in (d) in the x- or y-sequence, respectively.

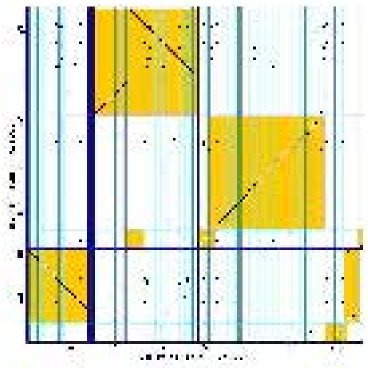
In Figure 3(e), we compare the completed genome of *HD100* with an early assembly of *HI100*, consisting of 69 contigs. Here, the algorithm produced 54 true positive-, 2 false positive- and 8 false negative extensions in the x-sequence.

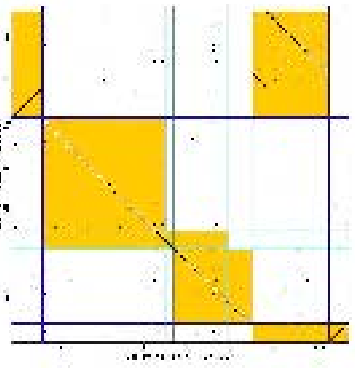
In Figure 3(f), we compare the completed genome of *HD100* with a later assembly of *HI100*, consisting of 7 contigs. Here, the algorithm produced 3 true positive-, 1 false positive- and 0 false negative extensions in the x-sequence. The false positive linking the 5^{th} and 6^{th} contigs in the x-assembly is due to an assembly error in the 7^{th} contig of HI100.

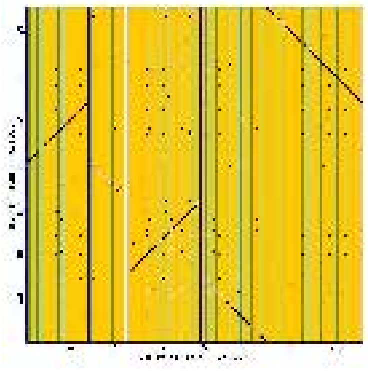
In summary, our implementation of this approach is able to detect and use a large proportion of all extendable local diagonals. The number of false positives is very low, and can be reduced further by masking repeats in the sequences before computing matches. An inspection of the false negatives shows that they are usually due to the fact that no appropriate informative matches are available. It is also interesting to note that this type of analysis can help to identify assembly errors (or inversions or transpositions between the two sequences, if present) in either of the assemblies under comparison, even if both are incomplete, as apparent in Figure 3(d,f).


(a)


(b)


(c)


(d)


(e)

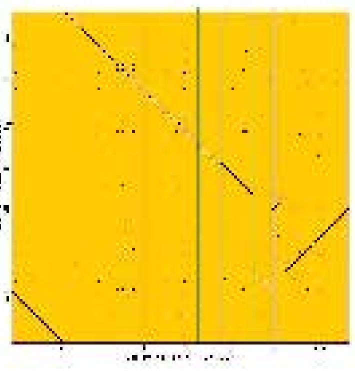

(f)

Figure 3: In each plot [FDH03], we compare assemblies of strains HI100 (x-axis) and HD100 (y-axis). We depict a comparison of early assemblies both (a) before, and (b) after, application of the layout algorithm. In (c) and (d) we show the result of the algorithm when applied to assemblies at different levels of x-coverage. In (e) and (f), we depict a comparison of the finished genome of HD100 with two different assemblies of HI100.

References

[AGM$^+$90] Altschul, S. F., Gish, W., Miller, W., Myers, E. W., und Lipman, D. J.: Basic local alignment search tool. *Journal of Molecular Biology*. 215:403–410. 1990.

[Be76] Berge, C.: *Graphs and hypergraphs*. North Holland. 1976.

[DPCS02] Delcher, A. K., Phillippy, A., Carlton, J., und Salzberg, S. L.: Fast algorithms for large-scale genome alignment and comparison. *Nucleic Acid Research*. 30(11):2478–2483. 2002.

[EBR$^+$04] Eppinger, M., Baar, C., Raddatz, G., Huson, D., und Schuster, S. C.: The power and limitations of genome comparison: A study of ϵ-proteobacteria. submitted. 2004.

[FDH03] Friedrichs, O. D., Dezulian, T., und Huson, D.: A meta-viewer for biomolecular data. *GI Jahrestagung*. 1:375–380. 2003.

[Ga73] Gabow, H. N.: An efficient implementation of edmonds' algorithm for maximum matching on graphs. *Journal of the ACM*. 23:221–234. 1973.

[GJ79] Garey, M. R. und Johnson, D. S.: *Computers and Intractability, a guide to the theory of NP-completeness*. Bell Telephone Laboratories, Inc. 1979.

[KPEL03] Kellis, M., Patterson, N., Endrizzi, M., und Lander, E.: Sequencing and comparison of yeast species to identify genes and regulatory elements. *Nature*. 423:241–254. 2003.

[LW88] Lander, E. S. und Waterman, M. S.: Genomic mapping by fingerprinting random clones: A mathematical analysis. *Genomics*. 2:231–239. 1988.

[MSD$^+$00] Myers, E. W., Sutton, G. G., Delcher, A. L., Dew, I. M., Fasulo, D. P., Flanigan, M. J., Kravitz, S. A., Mobarry, C. M., Reinert, K. H. J., Remington, K. A., Anson, E. L., Bolanos, R. A., Chou, H.-H., Jordan, C. M., Halpern, A. L., Lonardi, S., Beasley, E. M., Brandon, R. C., Chen, L., Dunn, P. J., Lai, Z., Liang, Y., Nusskern, D. R., Zhan, M., Zhang, Q., Zheng, X., Rubin, G. M., Adams, M. D., und Venter, J. C.: A whole-genome assembly of Drosophila. *Science*. 287:2196–2204. 2000.

[NW70] Needleman, S. und Wunsch, C.: A general method applicable to the search for similarities in the amino acid sequences of two proteins. *JMB*. 48:443–453. 1970.

[REB$^+$04] Rendulic, S., Eppinger, M., Baar, C., Raddatz, G., Jagtap, P., Rosinus, A., Lanz, C., Keller, H., Thiel, A., Velicer, G., Schmidt, P., und Schuster, S.: Comparative and functional genomic studies on prey-dependent and prey-independent strains of *Bdellovibrio bacteriovorus*. In Preparation. 2004.

[RJR$^+$04] Rendulic, S., Jagtap, P., Rosinus, A., Eppinger, M., Baar, C., Lanz, C., Keller, H., Lambert, C., Evans, K., Till, R., Goesmann, A., Meyer, F., Sockett, R., und Schuster, S.: A predator unmasked: The life cycle of *Bdellovibrio bacteriovorus* from a genomic perspective. *Science*. 303:689–692. 2004.

[SW81] Smith, T. und Waterman, M.: Identification of common molecular subsequences. *JMB*. 147:195–197. 1981.