Aligning Two Fragmented Sequences

Vamsi Veeramachaneni Piotr Berman Webb Miller
Dept. of Computer Science and Engineering
The Pennsylvania State University
University Park, PA 16802

Abstract

Upon completion of the human and mouse genome sequences, world-wide sequencing capacity will turn to other complex organisms. Current strategies call for many of these genomes to be incompletely sequenced. That is, holes will remain in the known sequence, and the relative order and orientation of the known sequence fragments may not be determined. Sequence comparison between two genomes of this sort may allow some of the fragments to be oriented and ordered relative to each other by computational means. We formalize this as an optimization problem, show that the problem is MAX-SNP hard, and develop a polynomial time algorithm that is guaranteed to produce a solution whose score is within a factor 3 of optimal.

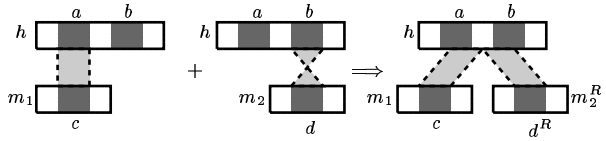
1. Introduction

As international projects determine the genome sequences of the handful of official model species, attention is turning to plans for sequencing many additional complex organisms. In the shotgun assembly phase, common to all sequencing approaches, several copies of a particular stretch of the genome are randomly partitioned into small fragments. Approximately 500 basepairs of each fragment are determined using variations of the Sanger method [1]. Overlapping sets of these *reads*, can be assembled into *contigs*, i.e., presumably contiguous sections of the genomic sequence. Ideally, the contigs form non-overlapping fragments that account for most of the target genome sequence. But the order and orientation of the contigs along the chromosome is unknown, or at least imperfectly known. In particular, for an arbitrary contig, h, it may be unknown whether h or its reverse complement, denoted h^R, is present, where h^R is formed by reversing h, interchanging A and T everywhere and interchanging C and G everywhere.

Two approaches have been proposed for overcoming this problem. The clone by clone approach [2] adopted by the Human Genome Project(HGP) starts by finding a minimal tiling set of clones that covers the target genome. These clones are sequenced one at a time using the shotgun approach. Finally contigs of different clones are ordered, oriented with respect to each other using the clone map. On the other hand, the whole genome shotgun assembly approach [3] skips the physical mapping step and sequences unmapped genomic clones. For further assembly it uses a library of pairs of reads, called *mates*, from the ends of long inserts randomly sampled from the genome. The presence of these mates in different contigs serves to order the contigs and give the approximate distance between them.

However, determining the complete sequence of a genome is quite expensive with either approach. Because researchers have been extracting biological information by studying conserved regions ([4], [5]) genetic data banks have rich contig information for many species. By comparing the conserved regions present in contigs of two organisms that are close in evolutionary terms, it might still be possible to infer some order/orient relationships. This process was manually performed in [6].

The figure below illustrates the sort of inference that is possible. Contig h (say, of human) includes region a, which aligns with region c in contig m_1 (say, of mouse). Another region of h, denoted b, aligns with d^R, the reverse complement of region d of mouse contig m_2. We infer that m_1 precedes m_2^R, relative to the orientation in which h is given.



We model the problem of determining order/orient relationships from alignments between contigs as follows. Data consists of a set of "h-contigs" and a set of "m-contigs", where each contig is simply an ordered list of conserved regions having associated alignment scores. We use $\sigma(a, b)$ to denote the score of the alignment between a and b, where a or a^R is a conserved region of an h-contig and b or b^R is a conserved region of an m-contig. The sample

dataset shown in Fig. 1 consists of contigs $h_1 : \langle a, b, c\rangle$, $h_2 : \langle d\rangle$, $m_1 : \langle s, t\rangle$, $m_2 : \langle u, v\rangle$ and the alignment scores $\sigma(a, s) = 4$, $\sigma(a, t) = 1$, $\sigma(b, t^R) = 3$, $\sigma(c, u) = 5$, $\sigma(d, t) = \sigma(d, v^R) = 2$.

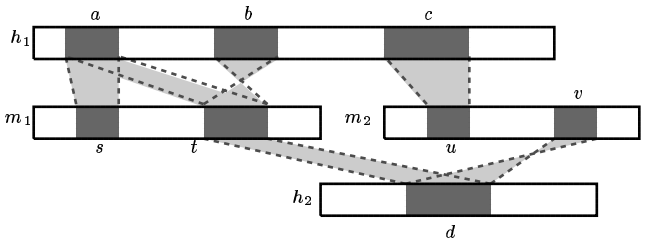


Figure 1. Sample set of data

Alignments involving conserved regions in contig h_1 may serve to orient and order several m-contigs relative to each other. Some of these m-contigs may in turn orient and order h_1 relative to additional h-contigs, and so on. This leads to an "island" of contigs that are oriented and ordered relative to one another. With ideal data, this process would partition the set of contigs into islands, such that inter-island order/orient relationships cannot be determined from the alignments.

In reality, the set of given alignments is frequently inconsistent with any proposed orientation and ordering of the contigs. Simple examples are shown in Fig. 2. In the first example, the $a - c$ alignment supports the current orientation, while the $b - d^R$ alignment calls for reversal of m. The second example violates our requirement that aligning regions be in the same order in the two sequences. More complex examples arise in practice when regions have been shuffled by evolutionary processes, when incorrect alignments are computed, and when contigs are incorrectly assembled from the shorter segments.

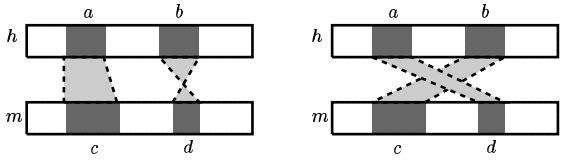


Figure 2. Inconsistent alignment sets

Our goal is to determine orientations and an order for each of the two sets of contigs that, possibly together with deletions of some of the conserved regions, gives two equal-length and consistently ordered lists of conserved regions showing high overall similarity. Ideally, this would mean maximizing the sum of the scores σ. For a simple example, consider the dataset given several paragraphs above. We can delete (i.e., ignore) b and t, reverse h_2 and place it after h_1 (giving $\langle a, c, d^R\rangle$), then place m_1 before m_2 in their given orientation (giving $\langle s, u, v\rangle$), which yields the score $\sigma(a, s) + \sigma(c, u) + \sigma(d^R, v) = 4 + 5 + 2 = 11$. See Fig. 3 for a picture of the solution.

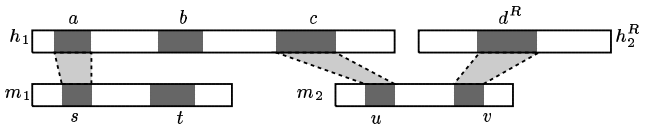


Figure 3. Solution to the data set of Fig. 1.

Note that once orientations and an order of the contigs are chosen, it is easy to decide how sites should be deleted to maximize the score — this is simply the classic problem of aligning two lists of symbols.

One of our results indicates that no polynomial-time algorithm can be guaranteed to orient and order the contigs so as to always maximize the resulting score. Indeed, even if we make a number of simplifying assumptions, such as (1) each conserved region is involved in precisely one alignment (e.g., for each a, $\sigma(a, b) > 0$ for just one b), (2) there is only one m-contig and (3) each h-contig has only two conserved regions, the problem of computing an optimal set of orient/order operations is MAX-SNP hard (Theorem 1).

We develop a $(3 + \epsilon)$ approximation algorithm (Theorem 5) for the order/orient problem. The formal developments presented in this paper, including results showing how algorithms for certain simpler problems can be combined to solve a more general problem, provide a conceptual framework for designing effective algorithms for computing high-scoring orient/order operations.

This paper is an extended abstract. The full version is available online at `http://bio.cse.psu.edu`.

2. Problem statement with variations

2.1. Consensus Sequence Reconstruction — CSR

Assume that we have two sets of DNA *fragments*, one for each species. Let us call these sets $\mathcal{H}$ and $\mathcal{M}$. We view each fragment as a sequence of *regions*. An *occurrence* of a region in a sequence can be *normal* or *reversed*.

Formally, we view each region as a symbol of a duplicated alphabet $\widetilde{\Sigma} = \Sigma \cup \Sigma^R$, and each fragment as a word from $\widetilde{\Sigma}^*$. To clarify the meaning of the reversal operation, we list its properties:

- $\Sigma \cap \Sigma^R = \emptyset$;
- for $a \in \Sigma$, $a^R \in \Sigma^R$, and for $a \in \Sigma^R$, $a^R \in \Sigma$;
- for $u, v \in \widetilde{\Sigma}^*$, $(uv)^R = v^R u^R$;
- for $u \in \widetilde{\Sigma}^*$, $(u^R)^R = u$;
- for $a, b \in \widetilde{\Sigma}$, $\sigma(a, b) = \sigma(a^R, b^R)$, where σ is a function $\sigma : \widetilde{\Sigma} \times \widetilde{\Sigma} \to \mathbf{R}$.

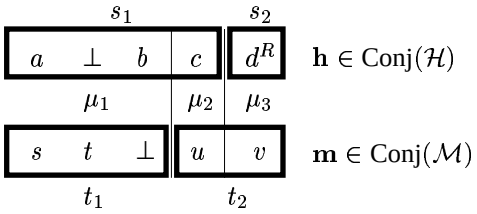


Figure 4. The conjecture pair representing the solution shown in Fig. 3.

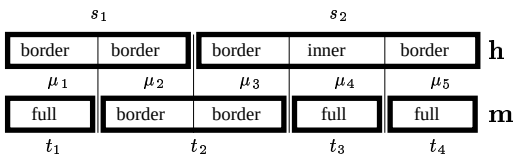


Figure 5. A conjecture pair $(\mathbf{h}, \mathbf{m})$ divided into matches and the sites classified by Def. 3.

We introduce an extra padding symbol $\perp$ such that $\perp^R = \perp$ and we extend the score function σ by setting

$$\sigma(a, \perp) = \sigma(\perp, a) = 0, \quad \forall a \in \widetilde{\Sigma}$$

For $s \in \widetilde{\Sigma}^*$ we define the *set of padded sequences* $\mathcal{P}s$ as the set of sequences obtained from s by inserting the padding symbol $\perp$ an arbitrary number of times.

For $s, t \in (\widetilde{\Sigma} \cup \{\perp\})^*$ where $s = a_1 a_2 \dots a_l$ and $t = b_1 b_2 \dots b_{l'}$, we define

$$\mathrm{Score}(s,t) = \begin{cases} 0 & \text{if } l \neq l' \\ \sum_{i=1}^{l} \sigma(a_i, b_i) & \text{otherwise} \end{cases}$$

Our general goal is to find an optimal conjecture for a consensus sequence for $\mathcal{H}$ and $\mathcal{M}$. More formally,

Definition 1 *For a set of fragments, $\mathcal{F} = \{f_1, \dots, f_k\}$, we define Conj($\mathcal{F}$) the set of valid conjecture sequences. A conjecture $\mathbf{f} \in$ Conj($\mathcal{F}$) is formed in three stages*

1. *For each fragment f_i we select some padded sequence $s_i \in \mathcal{P}f_i$*
2. *Some of s_i's are replaced by their reversals*
3. *$\mathbf{f} = s_{\pi(1)} \dots s_{\pi(k)}$, for some permutation π of $[1, k]$*

A *conjecture pair* is $(\mathbf{h}, \mathbf{m}) \in \mathrm{Conj}(\mathcal{H}) \times \mathrm{Conj}(\mathcal{M})$. Our goal is to maximize $\mathrm{Score}(\mathbf{h}, \mathbf{m})$.

2.2. Consistent match sets

Our algorithm will build conjecture pairs from smaller parts called *matches*, which pair together intervals selected from fragments of $\mathcal{H}$ and $\mathcal{M}$.

Given a fragment $h = a_1 \dots a_n$, the *site* $h(i, j)$ represents the contiguous subfragment $a_i \dots a_j$. A *match* is a pair of sites from fragments of different species.

Definition 2 *A conjecture pair $(\mathbf{h}, \mathbf{m})$ with a positive score produces a set of matches as follows:*

1. *Suppose $\mathbf{h}$ is formed as $s_1 s_2 \dots s_k$ and $\mathbf{m}$ is formed as $t_1 t_2 \dots t_l$ in Step 3 of Definition 1. We view this pair as a single word $\mathbf{w}$ where letters are columns of two symbols of $\widetilde{\Sigma} \cup \{\perp\}$.*

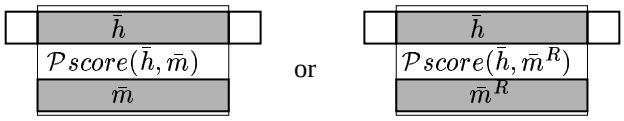


Figure 6. Match score for a full match

2. *As shown in Fig. 4, we split $\mathbf{w}$ at ends of s_i's and t_i's. The resulting pieces can be called* padded matches.
3. *Given a padded match, we obtain a match (pair of sites) by splitting it into two rows and deleting all $\perp$s from both the rows.*

A set of matches is consistent *if it is produced from some conjecture pair. The set of matches consistent with the pair shown in Fig. 4 is $\mu_1 = (h_1(1,2), m_1(1,2))$, $\mu_2 = (h_1(3,3), m_2(1,1))$ and $\mu_3 = (h_2^R(1,1), m_2(2,2))$.*

Definition 3 *If $h = h(1, n)$ then the sites in h can be classified as:*

full*:*	*$h(1, n)$ or, equivalently, h*
border*:*	*$h(1, i)$ or $h(i, n)$*
inner*:*	*none of the above*

A match that involves a full site is called a *full match*, a match that involves a border site is called a *border match*. In Fig. 5 μ_1, μ_4, μ_5 are full matches, and μ_2, μ_3 are border matches. One can see that we need to consider these two kinds of matches only.

Definition 4 *Given a site $\bar{h}$ in some fragment of $\mathcal{H}$ and a site $\bar{m}$ in some fragment of $\mathcal{M}$, we formulate the definition for* match score *MS($\bar{h}, \bar{m}$) in several steps.*

- *For $\bar{h}, \bar{m} \in \widetilde{\Sigma}^*$,*

$$\mathcal{P}score(\bar{h}, \bar{m}) = \max_{u \in \mathcal{P}\bar{h}} \max_{v \in \mathcal{P}\bar{m}} Score(u, v)$$

- *If one of the sites $\bar{h}$, $\bar{m}$ is full then the match score of $\bar{h}$ and $\bar{m}$ (also described by Fig. 6) is*

$$MS(\bar{h}, \bar{m}) = \max(\mathcal{P}score(\bar{h}, \bar{m}), \mathcal{P}score(\bar{h}, \bar{m}^R))$$

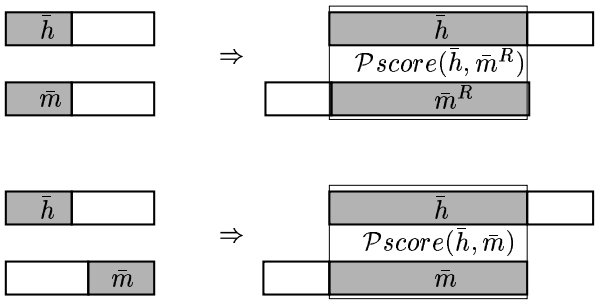


Figure 7. Matches formed from border sites.

- *If neither $\bar{h}$ nor $\bar{m}$ is a full site, then the match score of $\bar{h}$ and $\bar{m}$ is defined according to the method described by Fig. 7.*

Our algorithm does not depend on the way MS$(\bar{h}, \bar{m})$ is defined. However, this definition provides a correct model for our sequence reconstruction problem.

The score of a match and the padded sequences that support that score, represent the optimum alignment of the participating sites. We are interested in finding a consistent set of matches S with the maximum Score$(S) = \sum_{\mu \in S}$ MS(μ).

Remark 1

- *Instead of using padded sequences as building blocks of conjecture sequences, we can just as easily use subsequences formed by deleting arbitrary characters from a sequence. The score function for matches and the discussion of consistent match sets remains unchanged under this formulation.*
- *Given a conjecture pair $(\mathbf{h}, \mathbf{m})$, if S is the set of matches derived from $(\mathbf{h}, \mathbf{m})$ then Score(S) = Score$(\mathbf{h}, \mathbf{m})$.*
- *Given a consistent set of matches S we can easily compute a conjecture pair $(\mathbf{h}, \mathbf{m})$, such that Score(S) = Score$(\mathbf{h}, \mathbf{m})$.*

According to the last remark, we can formulate an equivalent version of the CSR problem: find a consistent set of matches with maximum total score.

3. Simpler versions of the problem

3.1. Consistent Subsets of Integer Pairs — CSoP

We will now show that a very restricted version of CSR is MAX-SNP hard. In particular, we impose the following restrictions:

1. the alphabet is of the form $\Sigma = \{a_1, \ldots, a_{2n}\}$ and $\mathcal{M} = \{a_1 a_2 \ldots a_{2n}\}$;
2. $\mathcal{H} = \{a_{i(1)}a_{j(1)}, \ldots, a_{i(n)}a_{j(n)}\}$, where the pairs $\{i(k), j(k)\}$ form a partition of $[1, 2n]$ and $i(k) < j(k)$ for $k \in [1, n]$;
3. $\sigma(a_i, a_j) = 1$ if $i = j$ and 0 otherwise.

In those terms the task is to find a set $U \subset \{1, 2, \ldots, 2n\}$ such that if $\{i(k), j(k)\} \subset U$ and $i(k) < l < j(k)$ then $l \notin U$ and such that $|U|$ is maximal. We call this problem Consistent Subsets of Pairs, CSoP. We will show that

Theorem 1 *CSoP is MAX-SNP hard*

Proof. Consider a solution U to an CSoP instance. We say that U is normal if it contains at least one element in each pair $\{i(k), j(k)\}$. Suppose that U is disjoint with an input pair $\{i(k), j(k)\}$ and we try to insert $i(k)$ to U, this insertion can create an invalid solution only if for some k' we have $i(k') \in U$, $j(k') \in U$ and $i(k') < i(k) < j(k')$. In this case we can replace U with $U' = U - \{i(k')\} \cup \{i(k)\}$, $|U'| = |U|$ the number of pairs disjoint with U' is lower. We may conclude that for every solution U there exists a solution U' such that $|U'| = |U|$ and U' intersect every one of the given pairs. We say that U' is a *normal solution*.

To prove our claim, we will reduce 3-MIS to CSoP.

The input to 3-MIS is a 3-regular graph (V, E), with $2n$ nodes, a feasible solution is an independent set of nodes, and the goal is to maximize the size of this independent set. Berman and Karpinski have formally shown in [7] that 3-MIS is MAX-SNP hard. We choose the following representation of the input graph: an $2n \times 3$ matrix A such that $\{i, j\}$ is an edge iff $j \in \{A[i, 1], A[i, 2], A[i, 3]\}$. We also require that the consecutive nodes are never adjacent, i.e., there are no edges of the form $\{i, i + 1\}$ (for $n > 6$ we can order the nodes in such a manner using Dirac's theorem [8]).

In our approximation preserving reduction, the instance translation is as follows: $\mathcal{M} = \{a_1 \ldots a_{10n}\}$; $\mathcal{H} = \mathcal{H}_{\text{nodes}} \cup \mathcal{H}_{\text{edges}}$, where

$$\begin{aligned} \mathcal{H}_{\text{nodes}} &= \{\, \{a_{5i-4}a_{5i}\} : \, 1 \le i \le n \,\} \text{ and} \\ \mathcal{H}_{\text{edges}} &= \{\, \{a_{5i-b}a_{5j-c}\} : \, i < j, \; A[i, b] = j \text{ and} \\ &\qquad A[j, c] = i \,\}. \end{aligned}$$

Consider a normal solution U. One can show that U contains exactly one element in each *edge pair* $\{5i - b, 5j - c\}$, otherwise there exists node k such that $i < k < j$, hence $5i - b < 5k - 4 < 5k < 5j - c$, hence the *node pair* $\{5k - 4, 5k\}$ is disjoint with U and U is not normal.

Consider now two node pairs that are contained in U, $\{5i - 4, 5i\}$ and $\{5j - 4, 5j\}$. One can see that no edge connects i and j, otherwise the respective edge pair would be disjoint with U. Define $W = \{i : \{5i - 4, 5i\} \subset U\}$.

As we observed, W is an independent set. Moreover, the size of U equals $5n + |W|$.

Consider now an idependent set W. For every edge e there exists an endpoint of e, say $i(e)$ such that $i(e) \in W$, we can assume that $e = \{i(e), A[i(e), b(e)]\}$. We can form a normal solution with $5n + W$ elements as follows

$$\{5i : i \in V\} \cup \{5i(e) - b(e) : e \in E\} \cup \{5i - 4 : i \in W\}.$$

Therefore this reduction preserves approximability. ❑

3.2. Reducing CSR to 1-CSR

We now consider 1-CSR, i.e., the CSR problem with the following restriction: set $\mathcal{M}$ consists of exactly one sequence. We will show the following theorem.

Theorem 2 *If there exists an approximation algorithm $\mathcal{A}$ that solves 1-CSR with approximation ratio r, then there exists an approximation algorithm $\mathcal{A}'$ that solves CSR with approximation ratio $2r$.*

Proof. For $\mathcal{F} = \{u_1, \ldots, u_n\}$ we define $\mathcal{F}' = \{u_1 \ldots u_n\}$, a set containing only the concatenation of all words from $\mathcal{F}$, in some arbitrary order. The algorithm $\mathcal{A}'$ processes the input instance of CSR, $(\mathcal{H}, \mathcal{M}, \sigma)$, as follows: it runs $\mathcal{A}$ twice, on $(\mathcal{H}, \mathcal{M}', \sigma)$ and on $(\mathcal{M}, \mathcal{H}', \sigma)$, and selects the better of the two solutions.

Let Opt(X) be the score of the optimum solution for instance X. The proof can be completed by showing that
$\text{Opt}(\mathcal{H}, \mathcal{M}', \sigma) + \text{Opt}(\mathcal{M}, \mathcal{H}', \sigma) \geq \text{Opt}(\mathcal{H}, \mathcal{M}, \sigma)$ ❑

3.3. 1-CSR and Interval Selection Problem – ISP

A 1-CSR problem instance has the form $(\mathcal{H}, m, \sigma)$. Because each fragment of $\mathcal{H}$ is involved in at most one match, we can assume that in each match the site from $\mathcal{H}$ is full. Thus each match in a solution can be described as $(k, [i, j])$, which denotes pair $(h_k, m(i, j))$. Selecting such a match yields profit $\text{MS}(h_k, m(i, j))$.

We can reduce 1-CSR to a more abstract Interval Selection Problem, ISP for short, where we are given set A of integer intervals and a non-negative profit function $p : [1, k] \times A \to R^+$. The task is to select at most one interval of A for each $i \in [1, k]$, so that the selected intervals are disjoint and the sum of profits is maximal. ISP was studied in the context of scheduling by Bar-Noy *et al.* [9], who described an algorithm with ratio 2. Later in [10] Berman and DasGupta described a *Two Phase Algorithm* that obtains ratio 2 and runs in time $O(n \log n)$, where $n = k|A|$.

Our reduction defines as A the set of all subintervals of $[1, |m|]$ and for each fragment $h_i \in \mathcal{H}$ sets $p(i, [d, e]) = \text{MS}(h_i, m(d, e))$. Clearly an approximation algorithm for ISP yields an algorithm for 1-CSR with exactly the same approximation ratio.

Corollary 1 *There exists a polynomial time algorithm for the CSR problem with approximation factor 4.*

4. Approximation algorithms for CSR

4.1. Iterative improvements

We will maintain the solution to a CSR problem instance as a consistent set of matches. To form tools for solving the general problem, we will first describe how to search for one type of matches only i.e. only border matches or only full matches. The algorithms we use there are selected in such a way that later we will be able to combine them into an algorithm that searches for both types of matches. We tackle different versions of the problem in the following manner:

- We define an iterative *improvement algorithm*
  - The algorithm is defined by set $\mathcal{I}$ of *improvement methods*, i.e. a finite set of routines that have a constant number of parameters of the form $f(i, j)$ where f is a fragment. For $\mathcal{R} \in \mathcal{I}$, and a parameter vector p, an *improvement attempt* $I = \mathcal{R}(p)$ changes the current solution by discarding some matches and making some new matches.
  - *gain*(I), the gain of an improvement attempt I, is the increase in total score after the improvement attempt I; if a given I is not applicable to the current legal set then $\text{gain}(I) = 0$.
  - The algorithm starts with an empty set of matches and makes improvement attempts with positive gain until none exists.
- To ensure that our algorithm runs in polynomial time, we use the scaling method described in [11]. This approach increases the approximation ratio by a factor $(k + 1)/k$, where k is an upper bound on the number of matches. Thus, when we prove an approximation ratio τ, the ratio actually proven has the form $\tau(k + 1)/k$ or $\tau + \epsilon$. However, in practice, alignment scores should have few precision bits and this step should not be necessary.
- In the analysis, we use the optimum solution, Opt, and the set of matches generated by our algorithm, L, to define a collection of improvement attempts, $\mathcal{J}$. $\mathcal{J}$ is constructed such that each attempt $I \in \mathcal{J}$ removes matches $\phi(I) \subseteq L$ and creates matches $\omega(I) \subseteq Opt$. Since the algorithm has terminated, no improvement has a positive gain which leads to the inequality

$$\sum_{I \in \mathcal{J}} \text{Score}(\omega(I)) \leq \sum_{I \in \mathcal{J}} \text{Score}(\phi(I))$$

By showing that the score of each match of Opt appears in the term on the left exactly n_1 times and that the score of each match of L appears on the right at most n_2 times, we have $n_1 \times \text{Score}(Opt) \leq n_2 \times \text{Score}(L)$. In this manner we prove that the algorithm defined by the set of improvement methods has approximation ratio $\frac{n_2}{n_1} + \epsilon$.

In defining the improvement methods and in the subsequent analysis we use the following notions:

- The *solution graph* of a set of matches S is the bipartite graph $(\mathcal{H} \cup \mathcal{M}, \mathcal{E})$, where $\{h, m\} \in \mathcal{E}$ iff S contains a match with sites in h and m.
- The connected components of this graph are called *islands*.
- In an island that consists of one fragment only, the fragment is *simple*.
- In an island that consists of two fragments only, one of the fragments is *simple* and the other *multiple*.
- In other islands, fragments that participate in a single match are *simple* and fragments that participate in more than one match are *multiple*.

Definition 5 *In the following definitions f is a fragment, $\bar{f}, \breve{f}$ are sites in f, S a consistent set of matches and F a set of fragments*

- *Mult(S) is the set of all multiple fragments of S.*
- *Simp(S) is the set of all simple fragments of S.*
- *Site $f(i,j)$ is contained in $f(i',j')$ if $i' \leq i \leq j \leq j'$.*
- *$\widehat{S}_{\mathcal{H}}$ is the set if all sites of fragments of $\mathcal{H}$ that participate in matches of S. $\widehat{S}_{\mathcal{M}}$ is defined similarly. $\widehat{S} = \widehat{S}_{\mathcal{H}} \cup \widehat{S}_{\mathcal{M}}$.*
- *$f(i,j)$ is* hidden *by $f(i',j')$ if $i' < i \leq j < j'$, if $f(i',j') \in \widehat{S}$, then we also say that $f(i,j)$ is hidden by S.*

4.2. Full CSR

In Full CSR problem we are limiting the legal solutions to a given CSR instance to those that contain full matches only.

Consider the solution graph of a solution to a Full CSR problem instance. Because each match in this solution contains a full site, for each edge in our graph one of the ends has one neighbor only. Consequently, in each island, at most one node is a multiple fragment.

Our improvement methods create new full matches using Two Phase Algorithm, TPA(B, S), where

- B is a union of sites of $\mathcal{M}$ and defines
  Sites$(B) = \{\bar{m} : \bar{m}$, viewed as an interval, is contained in $B\}$;
- S is the current solution and is used to define the profit function
  $p(h, \bar{m}) = \text{MS}(h, \bar{m}) - \text{Cb}(h, S)$.

We run TPA(B, S) with index set $\mathcal{H}$, interval set Sites(B) and profit function p. In our algorithms TPA(B) is a shorthand for TPA(B, S) where S is the current solution.

Let S be the current set of matches. In the improvement methods described below, a site $\bar{f}$ may need to be *prepared* for a match. The manner of *preparation* of the site depends on the classification of f:

- $f \in$ Simp(S): detach f from its match (if any).
- $f \in$ Mult(S): if the site is hidden by S it cannot be prepared (and the improvement that specifies a match of $\bar{f}$ cannot proceed). Otherwise, restrict any match of the form $(g, \breve{f})$ to $(g, \breve{f} - \bar{f})$. Note that if $\breve{f}$ is contained in $\bar{f}$, g becomes completely detached.

Our iterative algorithm *Full Improve* has one improvement method.

$I_1(f, \bar{g}, \breve{g})$
If $\breve{g}$ contains $\bar{g}$ and is not hidden by S,
1. Prepare the sites f and $\breve{g}$.
2. Match f with $\bar{g}$ (we plug in f to site $\bar{g}$).
3. Run TPA$(\breve{g} - \bar{g})$.
4. If g is detached from some site $\bar{f}_1$ during preparation of $\breve{g}$ in Step 1, run TPA$(\bar{f}_1)$.

We say that $\bar{g}$ is the target of this improvement attempt.

Execution of attempt $I_1(f, \bar{g}, \breve{g})$ is shown in Fig. 8. Preparation of $\breve{g}$ detaches g from f_1. Sites on which TPA is run are filled with slanted lines

Theorem 3 *Algorithm Full Improve solves the Full CSR problem with approximation factor $3 + \epsilon$.*

Proof. Omitted due to space constraints. ❑

4.3. Border CSR

In Border CSR problem we consider problem instances where the optimum solution contains border matches only.

There is a simple maximum weight matching based algorithm with approximation ratio 2 for the Border CSR problem. However, we prefer an alternate algorithm, *Border Improve*, with approximation ration 3 since it can later be combined with the algorithm of the previous section to solve the general problem.

The algorithm for the Full CSR problem creates full matches only. Therefore, each island of the solution contains at most one multiple fragment. We call such islands *1-islands*. The algorithm for the Border CSR problem allows each multiple fragment to participate in at most one

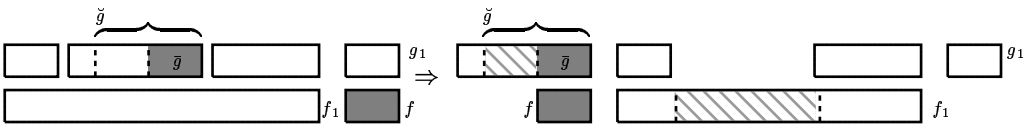


Figure 8. Execution of improvement attempt $I_1(f, \bar{g}, \breve{g})$

border match. So the solution may also contain *2-islands* — islands with two multiple fragments sharing a border match.

Since all sites chosen by the Border Improve algorithm are border sites, we will occasionally refer to them simply as sites in this section. The algorithm repeatedly prepares chosen sites on pairs of fragments and forms border matches. A site is prepared as described in the previous section. In addition, if the site belongs to the multiple fragment of a 2-island, we first *break* the 2-island by removing the match between the two multiple fragments. This ensures that our solution consists of 1-islands and 2-islands only. We have two improvement methods.

$I_2(\bar{f}, \bar{g})$. Prepare the sites $\bar{f}, \bar{g}$ and match them.

$I_3(\bar{f}_1, \bar{g}_1, \bar{f}_2, \bar{g}_2)$. Applicable if f_1, g_1 are multiple fragments of the same 2-island. Prepare all four sites. Make the matches $(\bar{f}_1, \bar{g}_2)$ and $(\bar{g}_1, \bar{f}_2)$.

Let L be the solution generated by Border Improve. With the knowledge of the the optimum solution, Opt, we will construct a multi-set of improvement attempts $\mathcal{J}$. Each improvement attempt in $\mathcal{J}$ removes some matches of L and creates matches of Opt. $\mathcal{J}$ will have the property that if all the improvement attempts in it are carried out

- each match of L will be removed 12 times
- each match of Opt will be attempted 4 times

When the algorithm terminates because all attempts fail, adding the inequalities representing the failure of attempts of $\mathcal{J}$ gives us

$$12 \times \text{Score}(L) \geq 4 \times \text{Score}(Opt) \qquad (1)$$

which is the required result.

All the improvement attempts in $\mathcal{J}$ try to form matches present in Opt. Thus, in the description of $\mathcal{J}$ an attempt of method I_2 is described as $I_2(\bar{f})$, the other site being implicit. Similarly, an attempt of method I_3 is specified as $I_3(\bar{f}, \bar{g})$, where f, g are the multiple fragments of the same 2-island.

We call the sites that are specified in an attempt the *explicit parameters* and the sites that are implied the *implicit parameters*. We construct $\mathcal{J}$ as follows:

- Let f be any simple fragment or multiple fragment in a 1-island of L. Let f^1, f^2 be the border sites of f in Opt. Then $\mathcal{J}$ contains two improvement attempts $I_2(f^1)$, two improvement attempts $I_2(f^2)$.
- Let f, g be the two multiple fragments of a 2-island in L. Let f^1, f^2 be the border sites of f and let g^1, g^2 be the border sites of g. Then $\mathcal{J}$ contains the four improvement attempts – $I_3(f^1, g^1)$, $I_3(f^1, g^2)$, $I_3(f^2, g^1)$ and $I_3(f^2, g^2)$.

Lemma 1 *The score of each (border) match of Opt is added exactly 4 times in gain*$(\mathcal{J})$.

Proof. From the construction of $\mathcal{J}$ described above, it is easy to see that each border site of Opt is the explicit parameter of an I_2 or I_3 improvement attempt exactly twice. Because this applies to both sites of a border match, each match of Opt is attempted 4 times. ❑

Lemma 2 *The score each border match of L is lost at most 12 times in gain*$(\mathcal{J})$.

Proof. Consider the border match formed by the two multiple fragments, f, g, in a 2-island. Let f^1, f^2 be the border sites of h and let g^1, g^2 be the border sites of g in Opt.

- The match between f and g is broken 4 times because of the improvement attempts $I_3(f^1, g^1)$, $I_3(f^1, g^2)$, $I_3(f^2, g^1)$, $I_3(f^2, g^2)$. These are the only attempts of $\mathcal{J}$ in which f^1, f^2, g^1, g^2 are explicit parameters.
- Each of the sites f^1, f^2, g^1, g^2 is an implicit parameter of two improvement attempts in $\mathcal{J}$. These 8 improvement attempts break the 2-island during the preparation of the concerned site.

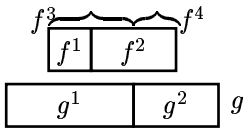
Thus, the score of the match is lost 12 times overall. ❑

Lemma 3 *The score each full match of L is lost at most 12 times in gain*$(\mathcal{J})$.

Proof. Consider any full match $(f, \bar{g}) \in L$. Let f^1, f^2 be the border sites of f and let g^1, g^2 be the border sites of g in Opt.

- The sites f^1, f^2 participate in 4 attempts each. These attempts detach f from g during preparation.

- As indicated by the figure below the 4 attempts in which g^1 participates restrict the portion of the match represented by site f^3. Similarly, the 4 attempts in which g^2 participates restrict the the portion of the match represented by site f^4. Overall these restrictions subtract the score of the match exactly 4 times.



Thus, the match loses its score at most 12 times overall. ❑

Theorem 4 *Algorithm Border Improve solves the Border CSR problem with approximation factor* $3 + \epsilon$.

Proof. The proof follows from inequality 1, Lemma 1, Lemma 2 and Lemma 3. ❑

4.4. General CSR

We now consider the general CSR problem. A site is prepared in exactly the same manner as in Section 4.3. Thus, the solution generated by the algorithm consists of 1-islands and 2-islands only.

The iterative improvement algorithm, *CSR Improve*, consists of method I_1 from Section 4.2 and methods I_2, I_3 from Section 4.3. I_2, I_3 are modified by treating the border sites as targets of I_1 attempts. Thus, for each border site an additional site that contains the border site needs to be specified. Since the modifications are similar for both methods, only I_2 is explained in detail.

$I_2(f^1, f^2, g^1, g^2)$
Applicable if f^1, g^1 are border sites within f^2, g^2
1. Prepare f^2, g^2.
2. Match the border sites f^1, g^1.
3. If f was detached from some site $\bar{g}_1$ in Step 1, run TPA($\{\bar{g}_1, g^2 - g^1\}$) else run TPA($g^2 - g^1$).
4. If g was detached from some site $\bar{f}_1$ in Step 1, run TPA($\{\bar{f}_1, f^2 - f^1\}$) else run TPA($f^2 - f^1$).

Fig. 9 shows a sample I_2 improvement attempt.

Also, if an I_2 or I_3 attempt breaks a 2-island during preparation, the attempt can be combined with an I_1 attempt that targets the newly exposed border site (or part of it).

Theorem 5 *Algorithm CSR Improve solves the CSR problem with approximation ratio* $3 + \epsilon$.

Proof. Omitted due to space constraints. ❑

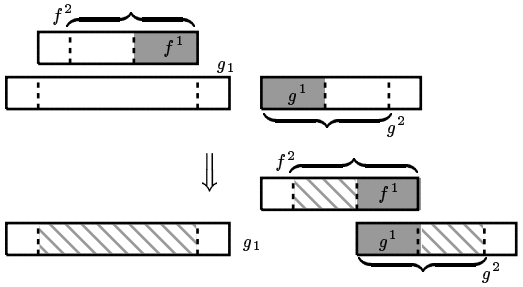


Figure 9. In $I_2(f^1, f^2, g^1, g^2)$ improvement attempt, f is detached from g_1 when the site f^2 is prepared. After the shaded border sites are matched, TPA is run on the sites filled with slanted lines.

5. Acknowledgments

This work was supported by grant HG02238 from the National Human Genome Research Institute.

References

[1] F. Sanger and A. R. Coulson. Dna sequencing with chain-terminating inhibitors. *Proc. Natl. Acad. Sci. USA*, 74(12):5463–5468, 1977.

[2] The Sanger Center & The Genome Sequencing Center. Towards a complete human genome sequence. *Genome Research*, 8:1097–1108, 1998.

[3] J. Weber and G. Myers. Whole genome shotgun sequencing. *Genome Research*, 7:401–409, 1997.

[4] J. Bouck *et. al.* Analysis of the quality and utility of random shotgun sequencing at low redundancies. *Genome Research*, 8:1074–1084, 1998.

[5] M. McClelland *at. al.* Comparison of the *escherichia coli* k12 genome with sampled genomes of *klebsiella pneumoniae* and three *salmonella enterica* serovars, typhimurium, typhi and paratyphi. *Nucleic Acids Res.*, 28:4974–4986, 2000.

[6] P. Onyango *et. al.* Sequence and comparative analysis of the mouse 1 megabase region orthologous to the human 11p15 imprinted domain. *Genome Research*, 10:1697–1710, 2000.

[7] P. Berman and M. Karpinski. On some tighter inapproximability results. *Automata, Languages and Programming, ICALP'99*, 1999.

[8] G. Dirac. Some theorems on abstract graphs. *Proc. London Math. Soc*, 2:69–81, 1952.

[9] A. Bar-Noy, S. Guha and B. Schieber. Approximating the throughput of multiple machines in real-time scheduling. *Proc. 31st ACM STOC*, pages 622–631, 1999.

[10] P. Berman and B. DasGupta. Multi-phase algorithms for throughput maximization for real-time scheduling. *Journal of Combinatorial Optimization*, 4(3):307–323, 2000.

[11] B. Chandra and M. M. Halldórsson. Greedy local improvement and weighted packing approximation. *SODA*, 1999.