Discrete Applied Mathematics 127 (2003) 119–143

DISCRETE APPLIED MATHEMATICS

Aligning two fragmented sequences

Vamsi Veeramachaneni, Piotr Berman, Webb Miller*

Department of Computer Science and Engineering, The Pennsylvania State University, University Park, PA 16802, USA

Received 1 July 2001; accepted 27 November 2001

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.
© 2002 Elsevier Science B.V. All rights reserved.

Keywords: Sequence assembly; MAX-SNP hardness; Approximation algorithms

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 [9]. 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

* Corresponding author. Tel.: +814-865-4551; fax: +814-865-3176.
E-mail address: webb@cse.psu.edu (W. Miller).

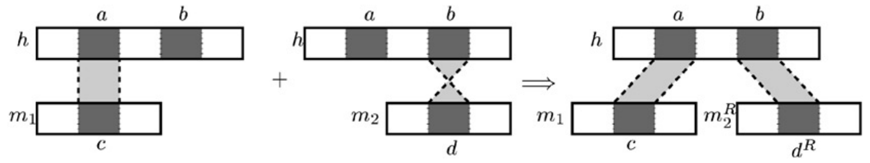

Fig. 1. Use of sequence comparison to orient/order contigs. Contig h (say, of human) includes region a, which aligns with region c in contig m_1 (say, of mouse). Also, another region of h, denoted b, aligns with d^R, the reverse complement of region d of mouse contig m_2. We infer that m_1 precedes m_2^R, relative to the orientation in which h is given. Note that the distance between m_1 and m_2^R *cannot* be inferred from such comparisons.

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 [10] adopted by the Human Genome Project (HGP) starts by finding a minimal tiling set of clones that covers the target genome. Then the individual 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 [11] 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. The result of this assembly process is, therefore, a collection of *scaffolds*, where each scaffold is a set of contigs that are ordered, oriented and positioned with respect to each other.

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 [4,7], 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 by [8]. Fig. 1 illustrates the sort of inference that is possible.

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. An example of a permissible data set 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 $\sigma(a,s)=4$, $\sigma(a,t)=1$, $\sigma(b,t^R)=3$, $\sigma(c,u)=5$, $\sigma(d,t)=\sigma(d,v^R)=2$. See Fig. 2. In this preliminary analysis of the problem, we have made the critical assumption that any two conserved regions are either identical or completely distinct. That is, we do not model any sort of partial overlap or strict containment between two conserved regions from the same species.

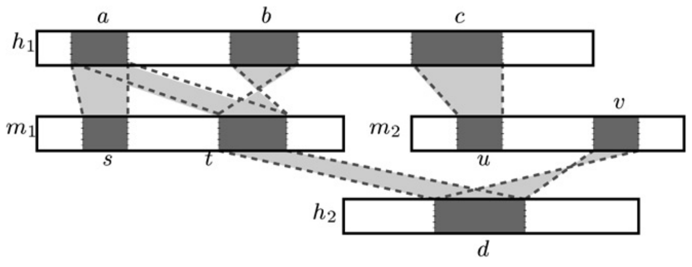

Fig. 2. Picture of a sample set of data, discussed in the text.

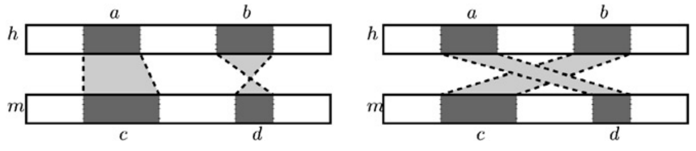

Fig. 3. Two potential inconsistencies among alignments between contigs. In the first example, contig h contains regions a and b, where a aligns with region c of contig m, and b aligns with d^R, where d is another region of m. The $a-c$ alignment supports the current orientation, while the $b-d$ alignment calls for reversal of m. The second example violates our requirement that aligning regions be in the same order in the two sequences.

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”[1] 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. 3. 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.

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 data set 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. 4 for a picture of the solution.

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

[1] While similar to scaffolds of [11], islands present a different combinatorial problem because they involve fragments of different species, do not imply any distance information and cannot overlap with other islands.

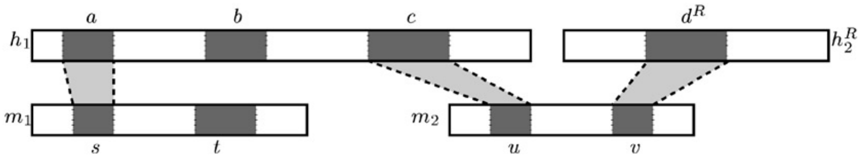

Fig. 4. Solution to the orient/order problem of Fig. 2. All alignments pictured in Fig. 2 that are inconsistent with this layout have been discarded.

of aligning two lists of symbols. (Here, however, each symbol of the "sequence" denotes a conserved region, rather than an individual nucleotide.) The difficulty lies with determining an optimal set of orient/order operations.

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 2).

This formal result indicates that there exists a number $\alpha < 1$ such that any polynomial-time orient/order algorithm will sometimes produce a solution whose total score is less than α times the optimal score. In particular, this result implies that for any existing heuristic one can generate data such that the heuristic result will be far from the correct one. This poses a challenge that can be addressed in two ways:

- Characterize the types of data that would "fool" the heuristic. Any time the heuristic is used, show that the input does not contain data with these "bad" properties.
- Find an algorithm that is designed on different principles and compare the two outcomes.

While the first option is preferable, it is difficult to formalize. Approximation algorithms offer alternative design principles to greedy heuristics and are the focus of this paper.

We develop a $(3 + \varepsilon)$ approximation algorithm (Theorem 6) 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.

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 $\mathscr{H}$ and $\mathscr{M}$. 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 $\tilde{\Sigma} = \Sigma \cup \Sigma^R$, and each fragment as a word from $\tilde{\Sigma}^*$. 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 $u, v \in \tilde{\Sigma}^*$, $(uv)^R = v^R u^R$;
- for $u \in \tilde{\Sigma}^*$, $(u^R)^R = u$;
- for $a, b \in \tilde{\Sigma}$, $\sigma(a,b) = \sigma(a^R, b^R)$, where σ is a function $\sigma : \tilde{\Sigma} \times \tilde{\Sigma} \to \boldsymbol{R}$.

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

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

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

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

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

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

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

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

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

2.2. Consistent match sets

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

Given a fragment $h = a_1 \ldots a_n$, the *site* $h(i,j)$ represents the contiguous subfragment $a_i \ldots a_j$. 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 $s_1 s_2 \ldots s_k$ and $\mathbf{m}$ is formed as $t_1 t_2 \ldots t_l$ in Step 3 of Definition 1. We view this pair as a single word $\mathbf{w}$ where letters are columns of two symbols of $\tilde{\Sigma} \cup \{\perp\}$.

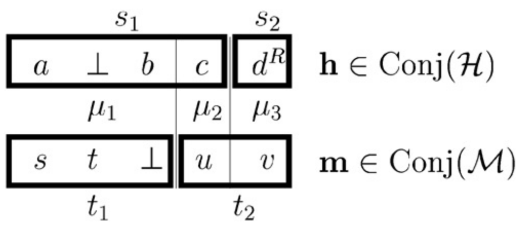
s_1
s_2
a $\perp$ b
c
d^R
$\mathbf{h} \in \mathrm{Conj}(\mathcal{H})$
μ_1
μ_2
μ_3
s t $\perp$
u
v
$\mathbf{m} \in \mathrm{Conj}(\mathcal{M})$
t_1
t_2

Fig. 5. The conjecture pair representing the solution shown in Fig. 4. Thick lines indicate padded fragments (or their reversals), thin lines separate matches. The set of matches consistent with this pair is $\mu_1 = (h_1(1,2), m_1(1,2))$, $\mu_2 = (h_1(3,3), m_2(1,1))$ and $\mu_3 = (h_2^R(1,1), m_2(2,2))$.

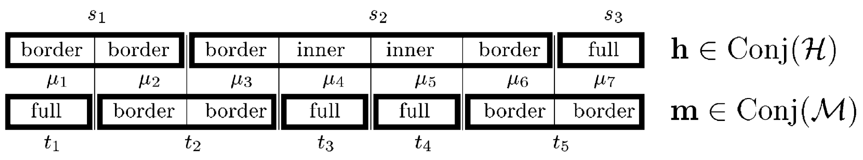
s_1
s_2
s_3
border
border
border
inner
inner
border
full
$\mathbf{h} \in \mathrm{Conj}(\mathcal{H})$
μ_1 μ_2 μ_3 μ_4 μ_5 μ_6 μ_7
full
border
border
full
full
border
border
$\mathbf{m} \in \mathrm{Conj}(\mathcal{M})$
t_1 t_2 t_3 t_4 t_5

Fig. 6. A conjecture pair $(\boldsymbol{h}, \boldsymbol{m})$ divided into matches and the sites classified according to Definition 3.

2. As shown in Fig. 5, 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.

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. 6 $\mu_1, \mu_4, \mu_5, \mu_7$ are full matches, and μ_2, μ_3, μ_6 are border matches. One can see that we need to consider these two kinds of matches only. Note that a match is a full match exactly when both ends of the respective padded match correspond to the ends of the same padded sequence e.g., sequence t_3 in Fig. 6.

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

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

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

- If one of the sites $\bar{h}$, $\bar{m}$ is full then the match score of $\bar{h}$ and $\bar{m}$ is

$$\mathrm{MS}(\bar{h}, \bar{m}) = \max(\mathscr{P}\,score(\bar{h}, \bar{m}), \mathscr{P}\,score(\bar{h}, \bar{m}^R)) \quad \text{(See Fig. 7).}$$

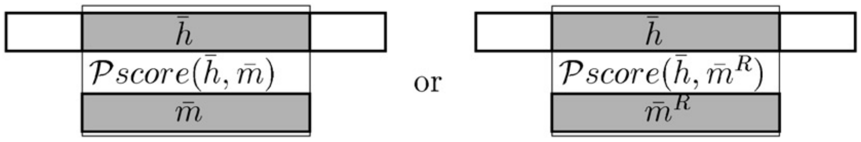
$\bar{h}$
$\mathcal{P}score(\bar{h},\bar{m})$
$\bar{m}$
or
$\bar{h}$
$\mathcal{P}score(\bar{h},\bar{m}^R)$
$\bar{m}^R$

Fig. 7. Matches formed from an inner site and a full site.

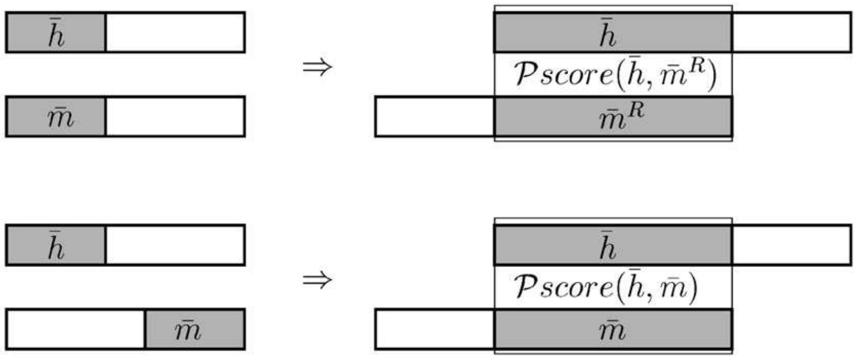

Fig. 8. Matches formed from border sites.

- 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. 8.

Our algorithm does not depend on the way $\mathrm{MS}(\bar{h},\bar{m})$ 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 $Score(S) = \sum_{\mu \in S} \mathrm{MS}(\mu)$.

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. Unambiguous CSR—UCSR

We form a restricted version of the CSR problem by demanding that $\sigma(a,b)=0$ for $a\neq b$. Moreover, we demand that every letter occurs exactly once in each of $\mathscr{H}$ and $\mathscr{M}$ (counting a^R as an occurrence of a). Suppose we also use subsequences as building blocks for conjectures. Under these restrictions, if we choose a letter for one of the two conjectures, it is determined what letter should match it in the other conjecture, so we call this the *unambiguous* CSR problem, or UCSR. It is easy to see that an optimum solution $(\mathbf{h},\mathbf{m})$ to an UCSR instance satisfies $\mathbf{h}=\mathbf{m}$, so we can restrict the valid solutions to single sequences from $Conj(\mathscr{H})\cap Conj(\mathscr{M})$ and replace σ with σ' so that $\sigma'(a)=\sigma(a,a)$. This way, for a valid solution $\mathbf{h}=a_1\ldots a_l$ we have $Score(\mathbf{h})=\sum_{i=1}^{l}\sigma'(a_i)$.

We can show that UCSR, while simpler, is not easier to approximate than CSR.

Lemma 1. *Let* $\varepsilon>0$. *There exist polynomial time computable functions* τ_0, *and* τ_1 *such that for every instance* X *of CSR*

1. $\tau_0(X)$ *is an instance of UCSR*;
2. *for every solution* $(\mathbf{h},\mathbf{m})$ *of* X, *there exists a solution* f *of* $\tau_0(X)$ *such that* $Score(f)=Score(\mathbf{h},\mathbf{m})$;
3. *for every solution* x *of* $\tau_0(X)$, $\tau_1(x)$ *is a solution of* X *that satisfies* $Score(\tau_1(x))\geqslant Score(x)(1-\varepsilon)$.

Proof. Let $X=(\mathscr{H},\mathscr{M},\sigma)$ be an instance of CSR. Let $\Sigma=\{a_1,\ldots,a_K\}$ be the set of letters that occur in $\mathscr{H}\cup\mathscr{M}$. By replacing multiple occurrences of a letter with distinct replica, and modifying the generator of Score, σ, appropriately, we modify X to an equivalent instance of CSR where each letter occurs in $\mathscr{H}\cup\mathscr{M}$ only once, and no letter occurs in its reversed form. Let $p=\lceil 1/\varepsilon\rceil$ and $s=2pK$. For each letter $a_i\in\Sigma$ and $1\leqslant l\leqslant s$ we define

$$u_l^i=a_{1,l}^i a_{2,l}^i\ldots a_{K,l}^i \quad\text{and}\quad v_l^i=b_{1,l}^i b_{2,l}^i\ldots b_{K,l}^i.$$

Next, we define

$$w_l^i=\begin{cases} u_l^i v_l^i & \text{if } a_i \text{ occurs in } \mathscr{H},\\ u_l^i(v_{s+1-l}^i)^R & \text{if } a_i \text{ occurs in } \mathscr{M}.\end{cases}$$

Finally, $x^i=w_1^i\ldots w_s^i$.

Now we are ready to define $\tau_0(X)=(\mathscr{H}',\mathscr{M}',\sigma')$. We obtain sets $\mathscr{H}'$ and $\mathscr{M}'$ from $\mathscr{H}$ and $\mathscr{M}$ by replacing each a_i with x^i. Next, we reduce the alphabet size of $\tau_0(X)$ by 50% by identifying the following pairs of letters: $a_{j,l}^i$ with $a_{i,l}^j$ and $b_{j,l}^i$ with $b_{i,l}^j$. This identification allows us to define $\sigma'(a_{j,l}^i)=\sigma(a_i,a_j)/s$ and $\sigma'(b_{j,l}^i)=\sigma(a_i,a_j^R)/s$.

To show Property 2, consider a solution $(\mathbf{h},\mathbf{m})$ of X, where $\mathbf{h}=(c_1\ldots c_L)$ and where $\mathbf{m}=(d_1\ldots d_L)$. The corresponding solution of $\tau_0(X)$ is the sequence $\rho(c_1,d_1)\ldots$

$\rho(c_L, d_L)$, where $\rho(c,d)$ is the word formed out of all the letters that are common to the replacement words of c and d. More formally, we define

$$\rho(c,d) = \begin{cases} a^i_{j,1}a^i_{j,2}\dots a^i_{j,s} & \text{if } c = a_i \text{ and } d = a_j, \\ (a^i_{j,1}a^i_{j,2}\dots a^i_{j,s})^R & \text{if } c = a_i^R \text{ and } d = a_j^R, \\ b^i_{j,1}b^i_{j,2}\dots b^i_{j,s} & \text{if } c = a_i \text{ and } d = a_j^R, \\ (b^i_{j,1}b^i_{j,2}\dots b^i_{j,s})^R & \text{if } c = a_i^R \text{ and } d = a_j. \end{cases}$$

One can see that this is a valid solution for $\tau_0(X)$. Moreover, we replace a pair of letters (c_i, d_i) with a word of length s, in which every letter scores $\sigma(c_i, d_i)/s$, thus the overall score is unchanged.

Consider a solution f of $\tau_0(X)$. Before we define $\tau_1(f)$ and prove Property 3, we will make some preliminary observations. Consider a word x^i that replaced a letter from $\mathscr{H}$ of the original CSR instance X. Because x^i is a sub-word of a word from $\mathscr{H}$, if f contains any letters from x^i , then they form a contiguous sub-word of f, let us call it y^i.

In turn, y^i can be split into subsequences that share letters with various x^js that replaced letter occurrences in $\mathscr{M}$. Again, each such subsequence forms a contiguous sub-word of y^i, say y^i_j, and there are at most K such sub-words. Now we can make an observation about the positions of the letters of y^i within x^i (or of $(x^i)^R$). If two consecutive letters belong to the same y^i_j, their position in x^i differ by at least $2K$, and otherwise they differ by at least 1. However, there can be at most $2K-1$ pairs of the second kind. Therefore if we have $l+(2K-1)+1$ letters in y^i, the positions of the first and the last differ by at least $2Kl+(2K-1)$, hence $2Kl+(2K-1)<2Ks$ and $l \leqslant s-1$. We can conclude that y^i contains less than $s+2K$ letters.

We define a pair of symbols (letters or reversed letters) (c^i, d^i) such that a symbol a from $\rho(c^i, d^i)$ occurs in y^i and there it has the largest score. Note that $\sigma'(a) > Score(y^i)/(s+2K)$, thus

$$\sigma(c^i, d^i) > Score(y^i)\frac{s}{s+2k} = Score(y^i)\left(1 - \frac{2K}{s+2K}\right) > Score(y^i)(1-\varepsilon). \quad (1)$$

If y^i is empty, then both c^i and d^i are words as well. Finally, we define

$$\tau_1(f = y^{\pi(1)}\dots y^{\pi(K)}) = (c^{\pi(1)}\dots c^{\pi(K)}, d^{\pi(1)}\dots d^{\pi(K)}) = (\mathbf{h}, \mathbf{m}).$$

Inequality (1) shows that $Score(\mathbf{h}, \mathbf{m}) > Score(f)(1-\varepsilon)$. □

The following theorem is an immediate consequence of Lemma 1.

Theorem 1. *If there exists a polynomial time algorithm that solves the UCSR problem with approximation ratio c, then there exists a polynomial time algorithm that solves the CSR problem also with ratio c.*

3.2. Consistent subsets of integer pairs—CSoP

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

1. the alphabet is of the form $\Sigma = \{a_1, \dots, a_{2n}\}$ and $\mathscr{M} = \{a_1 a_2 \dots a_{2n}\}$;
2. $\mathscr{H} = \{a_{i(1)}a_{j(1)}, \dots, a_{i(n)}a_{j(n)}\}$, 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 $U \subset \{1, 2, \dots, 2n\}$ 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 2. *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, 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 [3] have formally shown that 3-MIS is MAX-SNP hard (in folklore, this was an immediate consequence of the original paper on MAX-SNP class by Papadimitriou and Yannakakis). 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 [6]).

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

$$\mathscr{H}_{\text{nodes}} = \{\{a_{5i-4}a_{5i}\}\colon 1 \leqslant i \leqslant n\}$$

and

$$\mathscr{H}_{\text{edges}} = \{\{a_{5i-b}a_{5j-c}\}\colon i < j,\ A[i,b] = j \text{ and } A[j,c] = i\}.$$

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 $W=\{i\colon \{5i-4,5i\}\subset U\}$. As we observed, W is an independent set. Moreover, the size of U equals $5n+|W|$.

Consider now an independent 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:

$$\{5i\colon i \text{ is a node}\}\cup\{5i(e)-b(e)\colon e \text{ is an edge}\}\cup\{5i-4\colon i\in W\}.$$

Therefore this is an approximation preserving reducibility. □

3.3. Reducing CSR to 1-CSR

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

Theorem 3. *If there exists an approximation algorithm $\mathscr{A}$ that solves* 1-*CSR with approximation ratio r, then there exists an approximation algorithm $\mathscr{A}'$ that solves CSR with approximation ratio* $2r$.

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

Let $Opt(X)$ be the score of the optimum solution for instance X. It will suffice to show that

$$Opt(\mathscr{H},\mathscr{M}',\sigma)+Opt(\mathscr{M},\mathscr{H}',\sigma)\geqslant Opt(\mathscr{H},\mathscr{M},\sigma) \tag{2}$$

because inequality (2) implies that the solution found by $\mathscr{A}'$ has a score that is at least

$$\max\left(\frac{1}{r}Opt(\mathscr{H},\mathscr{M}',\sigma),\frac{1}{r}Opt(\mathscr{M},\mathscr{H}',\sigma)\right)\geqslant\frac{1}{2r}Opt(\mathscr{H},\mathscr{M},\sigma).$$

Assume that $\mathscr{H}=\{h_1,\dots,h_k\}$ and $\mathscr{M}=\{m_1,\dots,m_{k'}\}$. For convenience, assume that each letter appears only once in $\mathscr{H}\cup\mathscr{M}$. Consider an optimum solution for $(\mathscr{H},\mathscr{M},\sigma)$, i.e., a pair of words $\mathbf{h}=s_{\pi(1)}\dots s_{\pi(k)}\in Conj(\mathscr{H})$ and $\mathbf{m}=t_{\pi(1)}\dots t_{\pi(k')}\in Conj(\mathscr{M})$, where $s_{\pi(i)}$ is a (possible reversed) padded sequence of h_i (and similarly for t_i). Given that $(\mathbf{h},\mathbf{m})=(a_1\dots a_l,b_1\dots b_l)$ for some l and $Score(\mathbf{h},\mathbf{m})=\sum_{i=1}^{l}\sigma(a_i,b_i)$, we will also view this solution as the sequence of pairs $(a_1,b_1),\dots,(a_l,b_l)$. We will paint these pairs with two colors, say blue and yellow, and then we assemble a solution of $(\mathscr{H},\mathscr{M}',\sigma)$ from all the blue pairs, and a solution of $(\mathscr{M},\mathscr{H}',\sigma)$ from all the yellow pairs. By the very construction, the scores of these two solutions add to $Score(\mathbf{h},\mathbf{m})$.

The coloring of pairs requires some preliminary steps. First, we tag every letter in s_j with j, and we do the same with t_j. This way, a pair (a_i,b_i) has a pair of tags, say

(j,j'). If a pair has tag (j,j'), we will say that j is an $\mathscr{H}$-partner of j', and j' is an $\mathscr{M}$-partner of j. The partners can be ordered as follows: if for some $i < i'$ the pairs (a_i,b_i) and $(a_{i'},b_{i'})$ have tags (j,j_1) and (j,j_2), then j_1 is an earlier $\mathscr{M}$-partner of j than j_2; similarly, if these tags are (j_1,j) and (j_2,j), then j_1 is an earlier $\mathscr{H}$-partner of j than j_2. We will paint the tags, and each will have the color of its tag. Our coloring rules are the following:

- If j' is the first $\mathscr{M}$-partner of j, then we paint the tag (j,j') with blue.
- If j is the first $\mathscr{H}$-partner of j', then we paint the tag (j,j') with yellow.

A blue solution is formed in a simple manner: we rearrange the blue pairs (and reverse them if necessary) in such a way that their b's form a subsequence of $m_1 \ldots m_{k'}$. To show that we have formed a correct solution for $(\mathscr{H},\mathscr{M}',\sigma)$ it remains to show that the pairs with a's from a particular s_j form a contiguous part of this solution, and that they are ordered as in h_j. If such a pair is present, then, because it is blue, it has a tag (j,j') where j' is the first $\mathscr{M}$-partner of j. The pairs with tag (j,j') form a contiguous part of pairs with b's from $t_{j'}$. When we form the blue solution from the original one, we cut the original solution into pieces corresponding to various t's, reverse if necessary, remove the yellow pairs and reassemble in a new order. During this process, the fragment with tags (j,j') remains contiguous and keeps its original ordering.

The yellow solution for $(\mathscr{M},\mathscr{H}',\sigma)$ is formed identically. To show (2) it suffices to show that each tag is painted, so each $\sigma(a_i,b_i)$ will be added to the score of one of the two solutions (if a tag is painted with two colors, then $\sigma(a_i,b_i)$ is added to both scores, thus increasing the left-hand side of (2)). Consider a tag (j,j'). The positions of a_i's from s_j form an integer interval, say $[d,e]$, and positions of b_i's from $t_{j'}$ form another interval, say $[d',e']$. Thus a pair (a_i,b_i) has tag (j,j') if and only if $\max(d,d') \leqslant i \leqslant \min(e,e')$. One can see that if $d' \leqslant d$ then j' is the first $\mathscr{M}$-partner of j', and if $d \leqslant d'$, then j is the first $\mathscr{M}$-partner of j. In the former case tag (j,j') is blue, and in the latter it is yellow. $\square$

3.4. 1-CSR and interval selection problem—ISP

A 1-CSR problem instance has the form $(\mathscr{H},m,\sigma)$. Because each fragment of $\mathscr{H}$ is involved in at most one match, we can assume that in each match the site from $\mathscr{H}$ 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 $\mathrm{MS}(h_k,m(i,j))$.

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[2] by Bar-Noy et al. [1], who described an algorithm with ratio 2. Later Berman and DasGupta [2] described a *two phase algorithm* that obtains ratio 2 and runs in time $\mathrm{O}(n\log n)$, where $n = k|A|$.

[2] More general versions of ISP are considered with different A_i for each $i\in[1,k]$.

Our reduction defines as A the set of all subintervals of $[1,|m|]$ and for each fragment $h_i \in \mathscr{H}$ sets $p(i,[d,e]) = \mathrm{MS}(h_i, m(d,e))$. 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 $\mathscr{I}$ 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 $\mathscr{R} \in \mathscr{I}$, and a parameter vector p, an *improvement attempt* $I = \mathscr{R}(p)$ 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 *gain*$(I) = 0$.
  - The algorithm starts with an empty set of matches and makes improvement attempts with positive gain until none exists.
- If the scores are large numbers, gains can be comparatively very small and we cannot easily bound the running time. To ensure that our algorithm runs in polynomial time, we use the scaling method described by Chandra and Halldórsson [5] as follows. We first use the simple algorithm of Corollary 1 to obtain a solution with score X. We run the local improvement algorithm after truncating the weights of all match scores to integer multiples of X/k^2, where k is an upper bound on the number of possible matches. Since the score of the optimal solution is at most $4X$ and each local improvement has gain at least X/k^2, the number of improvements is limited to $4k^2$. This approach underestimates the optimal solution by at most X/k and hence, increases the approximation ratio by a factor $(k+1)/k$. Thus, when we prove an approximation ratio τ, the ratio actually proven has the form $\tau(k+1)/k$ or $\tau + \varepsilon$. 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, $\mathscr{J}$. $\mathscr{J}$ is constructed such that each attempt $I \in \mathscr{J}$ 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

$$\sum_{I\in\mathscr{I}} Score(\omega(I)) \leqslant \sum_{I\in\mathscr{I}} Score(\phi(I)).$$

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 $n_1 \times Score(Opt) \leqslant n_2 \times Score(L)$. In this manner we prove that the algorithm defined by the set of improvement methods has approximation ratio $(n_2/n_1) + \varepsilon$.

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 $(\mathscr{H} \cup \mathscr{M}, \mathscr{E})$, where $\{h,m\} \in \mathscr{E}$ 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, $\bar{f},\check{f}$ 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 $i' \leqslant i \leqslant j \leqslant j'$.
- Site $f(i,j)$ is adjacent to $f(i',j')$ if $i' = j+1$ or $j' = i-1$.
- For $f \in Mult(S)$, $Match(\bar{f},S) = \{g \mid (g,\check{f}) \in S$ and $\check{f}$ is contained in $\bar{f}\}$. We can extend the definition to sets of sites.
- $\mathrm{Cb}(f,S)$, the contribution of the fragment f to the solution S, is the sum of scores of all match scores in S involving f. The contribution of a set of fragments F is the sum $\mathrm{Cb}(F,S) = \sum_{f\in F} \mathrm{Cb}(f,S)$.
- $\hat{S}_{\mathscr{H}}$ is the set if all sites of fragments of $\mathscr{H}$ that participate in matches of S. $\hat{S}_{\mathscr{M}}$ is defined similarly. $\hat{S} = \hat{S}_{\mathscr{H}} \cup \hat{S}_{\mathscr{M}}$.
- $f(i,j)$ is *hidden* by $f(i',j')$ if $i' < i \leqslant j < j'$, if $f(i',j') \in \hat{S}$, then we also say that $f(i,j)$ is hidden by S. Note that if $(f,\bar{g}) \in S$, then only the border sites of f are not hidden.

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 $\mathscr{M}$ 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 $p(h,\bar{m})=\mathrm{MS}(h,\bar{m})-\mathrm{Cb}(h,S)$.

We run TPA(B,S) with index set $\mathscr{H}$, 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.

Lemma 2. *Let Opt be any optimal solution. A run of TPA(B,S) creates a set of matches with the sum of scores at least*

$$hope(B,S)=\frac{1}{2}\sum_{f\in Match(B,Opt)}(Cb(f,Opt)-Cb(f,S)).$$

Proof. From the definition of the profit function, we see that each fragment $f\in Match(B,Opt)$ participates with score $\mathrm{Cb}(f,Opt)-\mathrm{Cb}(f,S)$. Since TPA has approximation ratio 2 the result follows. □

Lemma 3. *If for each fragment of $\mathscr{H}\cup\mathscr{M}$ we know whether it is simple or multiple in Opt, then Full CSR has a 2-approximation algorithm.*

Proof. Let $\mathscr{H}=\mathscr{H}_{\mathrm{simp}}\cup\mathscr{H}_{\mathrm{mult}}$, where $\mathscr{H}_{\mathrm{simp}}=\mathscr{H}\cap Simp(Opt)$. Similarly let $\mathscr{M}=\mathscr{M}_{\mathrm{simp}}\cup\mathscr{M}_{\mathrm{mult}}$. Let S be the set of matches generated by the following algorithm:
1. Run TPA$(\mathscr{H}_{\mathrm{mult}})$ with index set $\mathscr{M}_{\mathrm{simp}}$.
2. Run TPA$(\mathscr{M}_{\mathrm{mult}})$ with index set $\mathscr{H}_{\mathrm{simp}}$.

Since no fragment is involved in both TPA runs, from Lemma 2 it follows that:

$$\begin{aligned}Score(S)&\geqslant\tfrac{1}{2}\,\mathrm{Cb}(Match(\mathscr{H}_{\mathrm{mult}}),Opt)+\tfrac{1}{2}\,\mathrm{Cb}(Match(\mathscr{M}_{\mathrm{mult}}),Opt)\\&\geqslant\tfrac{1}{2}\,(\mathrm{Cb}(\mathscr{M}_{\mathrm{simp}},Opt)+\mathrm{Cb}(\mathscr{H}_{\mathrm{simp}},Opt))\\&\geqslant\tfrac{1}{2}\,\mathrm{Cb}(Simp(Opt),Opt)\\&\geqslant\tfrac{1}{2}\,Score(Opt).\qquad\square\end{aligned}$$

The algorithm of Lemma 3 can be used only if the role that each fragment plays in *Opt* is known. Since this information is not generally available, the rest of this section describes a more complicated iterative improvement algorithm.

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).

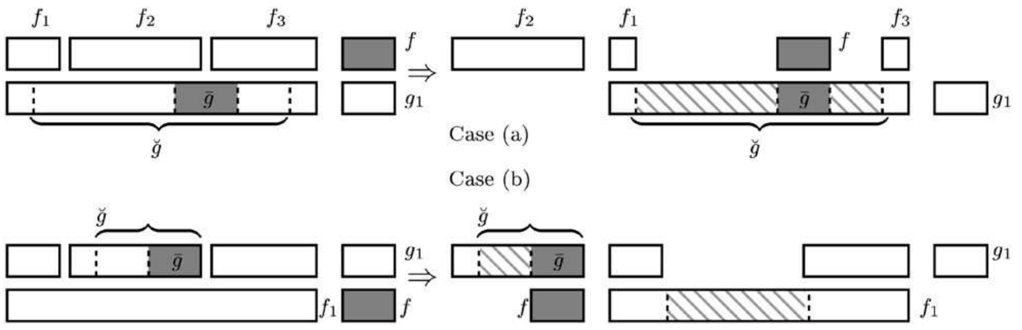

Fig. 9. Two cases (a), (b) of improvement attempt $I_1(f,\bar{g},\breve{g})$ are shown. In both cases preparation of f detaches it from g_1. In (a) preparation of $\breve{g}$ detaches f_2 and restricts matches made by f_1, f_3. In (b) preparation of $\breve{g}$ detaches g from f_1. Sites on which TPA is run are filled with slanted lines.

- $f \in Mult(S)$: 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 $\bar{g}$ (intuitively 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.
```

Fig. 9 shows two cases of an I_1 improvement attempt. Only Steps 1–3 are executed in the first case. Step 4 is executed in the second case because g is detached during preparation.

Let L be the consistent set of matches generated by our improvement algorithm. In the analysis that follows, we assume that the optimum legal solution is Opt.

Definition 6. Suppose $g \in Mult(Opt)$.

- If $(\bar{f}, \bar{g}) \in L$ and $\bar{g}$ hides $\breve{g} \in \widehat{Opt}$ we say that f *owns TPA site* $\breve{g}$. This defines "owns" and "TPA sites".
- A *TPA zone* is the union of adjacent TPA sites.
- If $(\bar{f}, \bar{g}) \in Opt$ and $\bar{g}$ is not part of any TPA zone, then $\bar{g}$ is a *plug-in site*. See Fig. 10.

Remark 2. The following remarks can be easily verified:

- Each TPA zone has a single owner.
- A TPA zone always extends between two plug-in sites (but not vice versa).

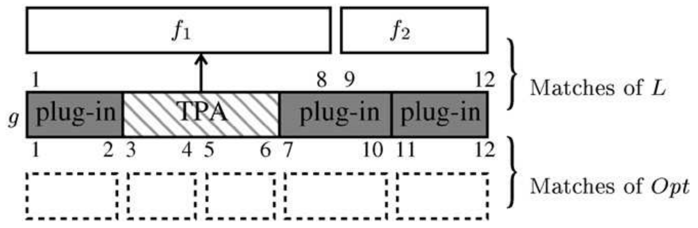

Fig. 10. $g \in Mult(L)$ is partitioned into TPA zones and plug-in sites. Arrows point to the owners. Matches of Opt are shown using fragments with dashed outlines. $g(1,8) \in \hat{L}$ hides $g(3,4), g(5,6) \in \widehat{Opt}$. Therefore $g(3,6) = g(3,4) \cup g(5,6)$ is a TPA zone owned by f_1. $g(1,2), g(7,10), g(11,12)$ are not hidden by any site of $\hat{L}$ so they are plug-in sites.

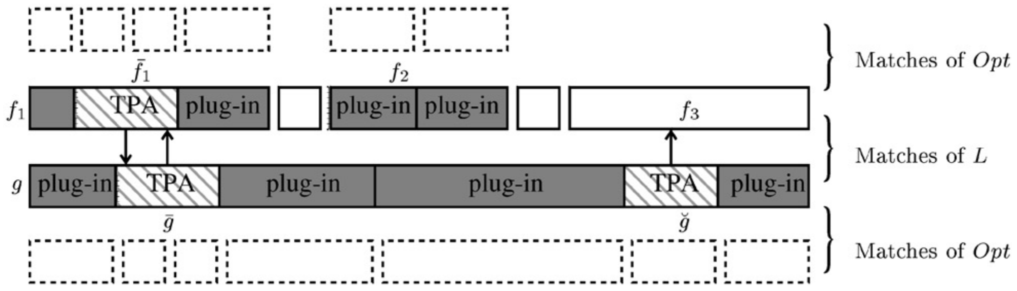

Fig. 11. TPA zones associated with plug-in sites. Fragments $f_1, f_2, g \in Mult(Opt)$ are partitioned into TPA zones and plug-in sites. TPA zone $\bar{f}_1$ is associated with the two plug-in sites adjacent to it because $f_1 \in Simp(L)$. TPA zone $\bar{g}$ of $g \in Mult(L)$ is not associated with its neighbors because its owner $f_1 \in Mult(Opt)$. TPA zone $\breve{g}$ of $g \in Mult(L)$ is associated with its neighbors because its owner $f_3 \in Simp(Opt)$.

- Any fragment of $Mult(Opt)$ is partitioned into TPA zones and plug-in sites.
- Any fragment of $Simp(L) \cap Mult(Opt)$ has at most one TPA zone.

Definition 7. TPA zone $\bar{g}$ *is associated with* plug-in site $\breve{g}$ if
- $\bar{g}$ is adjacent to $\breve{g}$ and
- $g \in Simp(L)$ or$g \in Mult(L)$ but some $f \in Simp(Opt)$ owns $\bar{g}$. See Fig. 11.
- zone($\breve{g}$) is the union of plug-in site $\breve{g}$ and the TPA zones associated with it.

We now construct the collection (multi-set) of improvement attempts $\mathscr{I}$. Each attempt of $\mathscr{I}$ targets a plug-in site. For a plug-in site $\bar{g}$, we will use the short-hand notation $I_1(\bar{g})$ to denote the attempt $I_1(f, \bar{g}, \text{zone}(\bar{g}))$, where $(f, \bar{g}) \in Opt$. For each plug-in site $\bar{g}$, $\mathscr{I}$ has one improvement attempt $I_1(\bar{g})$.

In our analysis, we now estimate $gain(\mathscr{I})$. In the improvement attempts of the algorithm a match is either made explicitly in Step 2 of an I_1 improvement method or as a result of a TPA run. The construction of $\mathscr{I}$ ensures that the explicit matches attempted are matches of Opt and that TPA runs are made only on TPA zones of multiple fragments. Therefore, using hope as a lower bound on the sum of scores of matches created by a TPA run, we can compute a lower bound of $gain(\mathscr{I})$ in which the positive terms are scores of matches of Opt and the negative terms are scores of matches of L. We show that in this estimate of $gain(\mathscr{I})$:

- the score of each match of Opt is added exactly once,
- the score of each match of L is subtracted at most 3 times.

Since no improvement attempt has a positive gain we then have

$$0 \geqslant gain(\mathscr{I}) \geqslant Score(Opt) - 3 \times Score(L) \tag{3}$$

and the required approximation ratio follows.

Lemma 4. *TPA is applied to each TPA zone exactly twice.*

Proof. Let $\bar{g}$ be a TPA zone.

- If $\bar{g}$ is associated with the neighboring two plug-in sites, the I_1 attempts that target these plug-in sites run TPA on $\bar{g}$ in Step 3.
- Otherwise, $\bar{g}$ is owned by some $f \in Mult(Opt)$ and f has two plug-in sites. The improvement attempts that target these plug-in sites, detach f from g and run TPA on $\bar{g}$ in Step 4. □

Lemma 5. *The score of each match of Opt is added exactly once in $gain(\mathscr{I})$.*

Proof. Consider any match $(f, \bar{g}) \in Opt$. Since $g \in Mult(Opt)$ can be partitioned into plug-in sites and TPA zones there are two possibilities:

- $\bar{g}$ is a plug-in site: the attempt $I_1(\bar{g}) \in \mathscr{I}$ so $\mathrm{MS}(f, \bar{g})$ is added once to the estimate of $gain(\mathscr{I})$.
- $\bar{g}$ is a TPA site: according to Lemma 4 two TPA runs are made on the TPA zone $\breve{g}$ containing $\bar{g}$. Since $f \in Match(\breve{g}, Opt)$ it follows from Lemma 2 that together these runs add $\mathrm{Cb}(f, Opt) = \mathrm{MS}(f, \bar{g})$ to the estimate of $gain(\mathscr{I})$. □

Note that since

$$Score(L) = \sum_{f \in Simp(L)} \mathrm{Cb}(f, L) = \sum_{f \in Mult(L)} \mathrm{Cb}(f, L) \tag{4}$$

all the negative terms in the estimate of $gain(\mathscr{I})$ can be expressed in the form $-(hl_f + apl_f + ppl_f)\mathrm{Cb}(f, L)$ where

- hl_f, or *hope loss of f*, is caused by the use of $\mathrm{Cb}(f, L)$ in the estimates of hope;
- apl_f, or *active preparation loss of f*, is caused by detaching f during preparation of sites on fragment f itself;
- ppl_f, or *passive preparation loss of f*, is caused by detaching f during preparation of sites on other fragments.

Lemma 6. *For $f \in Simp(Opt)$ we have $hl_f + apl_f = 1$.*

Proof. Suppose $f \in Simp(Opt)$, i.e., $(f, \bar{g}) \in Opt$ for some $\bar{g}$. There are two cases to consider:

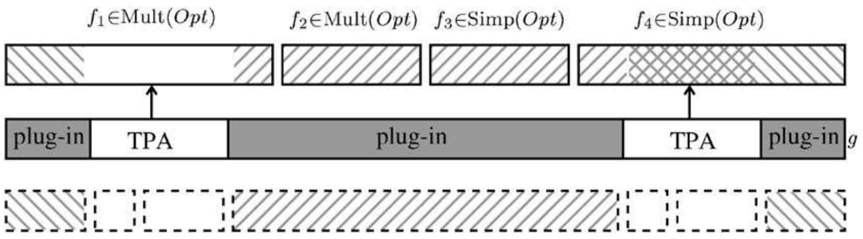

Fig. 12. The four types of matches in which $g \in Mult(Opt) \cap Mult(L)$ participates in L. Matches are restricted (removed) by I_1 attempts. Shading is used to denote the portion of the match restricted (removed) and the fragment that plugs in and is responsible for the restriction (removal).

- $\bar{g}$ is a plug-in site: $apl_f = 1$ because f is detached in Step 1 of the attempt $I_1(\bar{g}) \in \mathscr{J}$.
- $\bar{g}$ is a TPA site: By Lemma 4 two TPA runs are made on the TPA zone containing this TPA site; $hl_f = 1$ because (by Lemma 2) f contributes $-\frac{1}{2}\text{Cb}(f, L)$ to the hope of each TPA run, thus $hl_f = \frac{1}{2} + \frac{1}{2}$. $\square$

Lemma 7. *For $f \in Mult(Opt) \cap Simp(L)$ we have $hl_f + apl_f = 2$.*

Proof. Clearly, $hl_f = 0$. From Remark 2 it follows that f has exactly two border plug-in sites, say $\bar{f}, \breve{f}$. The improvements attempts $I_1(\bar{f})$, $I_1(\breve{f})$ detach f during preparation. So $apl_f = 2$. $\square$

Lemma 8. *The score of each match of L is lost at most three times in gain($\mathscr{J}$).*

Proof. By Eq. (4) it suffices to show that for every $f \in Simp(L)$, $hl_f + apl_f + ppl_f \leqslant 3$.

Consider the match $(f, \bar{g}) \in L$ and suppose $g \in Simp(Opt)$. By Lemma 6, $hl_g + apl_g = 1$. Equivalently this loss of $\text{Cb}(g, L)$ can be construed as a single passive preparation loss for each $f \in Match(g, L)$ because $\text{Cb}(g, L) = \sum_{f \in Match(g,L)} \text{Cb}(f, L)$. Therefore, $ppl_f = 1$ and using Lemma 6, Lemma 7 we have $hl_f + apl_f + ppl_f \leqslant 3$.

Now consider the match $(f, \bar{g}) \in L$ when $g \in Mult(Opt)$. We need to consider the four cases that are illustrated in Fig. 12.

Suppose that $f \in Mult(Opt)$, by Lemma 7 it suffices to show that $ppl_f \leqslant 1$. There are two cases:

- if f owns a TPA zone $\breve{g}$ the attempts that target the two plug-in sites adjacent to $\breve{g}$ can restrict the match. But as Fig. 12 shows for f_1, the restricted portions are non-overlapping and together subtract the score of the match at most once. So $ppl_f \leqslant 1$.
- if $\bar{g}$, the match of f (f_2 in Fig. 12) is contained within a plug-in site $\breve{g}$ then $ppl_f = 1$ because of the improvement attempt $I_1(\breve{g})$.

Suppose that $f \in Simp(Opt)$, by Lemma 6 it suffices to show that $ppl_f \leqslant 2$. There are two cases:

- if $\bar{g}$, the match of f (f_3 in Fig. 12) is contained within a plug-in site $\breve{g}$ then $ppl_f = 1$ because of the improvement attempt $I_1(\breve{g})$.

- if f (f_4 in Fig. 12) owns a TPA zone $\breve{g}$, $\breve{g}$ is associated with the two plug-in sites adjacent to it. Attempts that target these plug-in sites can restrict the match. These restrictions may overlap but together amount to at most twice the score of $(f, \bar{g})$. Thus, $ppl_f \leqslant 2$. □

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

Proof. The approximation ratio follows from inequality (3), Lemmas 5 and 8. □

4.3. Border CSR

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

Lemma 9. *There exists a polynomial time algorithm for the Border CSR problem with approximation factor* 2.

Proof. Consider the solution graph $(\mathscr{H} \cup \mathscr{M}, \mathscr{E})$ of the optimum solution to a Border CSR instance $(\mathscr{H}, \mathscr{M}, \sigma)$. Because each fragments has only two borders and every site must be a border site, this is a degree 2 bipartite graph. Thus, we can partition $\mathscr{E}$ into two matchings $\mathscr{A}$ and $\mathscr{B}$. Clearly the sum of the scores of the matches represented by one of the sets $\mathscr{A}$ or $\mathscr{B}$ is at least 50% of the total score. In this set each fragment participates in at most one match; thus we can assume that all sites in the matches are full.

Consequently, to find a legal solution to our instance of Border CSR with score at least 50% of the optimum, it suffices to find an optimum legal solution in which every match consists of two full sites. The latter we can find by applying the algorithm for the maximum weight matching to the bipartite graph with node set $\mathscr{H} \cup \mathscr{M}$ and edge weight function

$$w(\{h, m\}) = \mathrm{MS}(h, m). \qquad \square$$

However, we prefer an alternate algorithm with approximation ratio 3, *Border Improve*, for the Border CSR problem. Unlike the algorithm of Lemma 9, we will be able to combine this algorithm with the algorithm for the Full CSR problem discussed in the previous section.

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 border match. So, in addition to 1-islands, the solution may 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

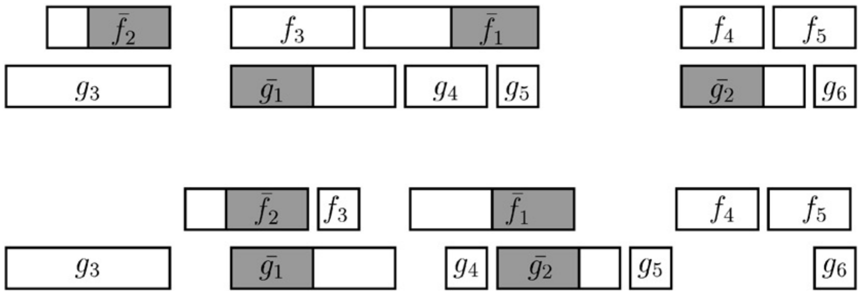

Fig. 13. $I_3(\bar{f}_1,\bar{g}_1,\bar{f}_2,\bar{g}_2)$ improvement attempt breaks the 2-island formed by f_1, g_1 and the 2-island formed by f_5, g_2. The matches in which f_3, g_4 participate are restricted while the matches in which f_2, f_4, g_5 participate are removed during preparation.

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 $(\bar{g}_1,\bar{f}_2)$.

Fig. 13 shows a sample I_3 improvement attempt.

Let L be the solution generated by Border Improve. With the knowledge of the optimum solution, *Opt*, we will construct a multi-set of improvement attempts $\mathscr{I}$. Each improvement attempt in $\mathscr{I}$ removes some matches of L and creates matches of *Opt*. $\mathscr{I}$ 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 four times.

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

$$12 \times Score(L) \geqslant 4 \times Score(Opt), \tag{5}$$

which is the required result.

All the improvement attempts in $\mathscr{I}$ try to form matches present in *Opt*. Thus, in the description of $\mathscr{I}$ 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 $\mathscr{I}$ 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 $\mathscr{I}$ 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 $\mathscr{I}$ contains the four improvement attempts—$I_3(f^1,g^1)$, $I_3(f^1,g^2)$, $I_3(f^2,g^1)$ and $I_3(f^2,g^2)$.

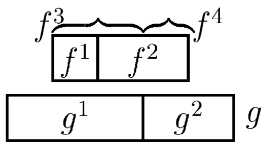

Fig. 14. $(f,\bar{g}) \in L$ is shown with the border sites that the fragments have in *Opt*. f^3, f^4 represent the portion of the match restricted by attempts involving g^1, g^2, respectively.

Lemma 10. *The score of each* (*border*) *match of Opt is added exactly four times in gain*($\mathscr{J}$).

Proof. From the construction of $\mathscr{J}$ 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 four times. □

Lemma 11. *The score each border match of L is lost at most* 12 *times in gain*($\mathscr{J}$).

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 four 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 $\mathscr{J}$ 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 $\mathscr{J}$. These eight 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 12. *The score each full match of L is lost at most* 12 *times in gain*($\mathscr{J}$).

Proof. Consider any full match $(f,\bar{g}) \in L$. Since $hl_f = 0$ it suffices to show that $apl_f + ppl_f \leqslant 12$. 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 four attempts each. $apl_f = 8$ because these attempts detach f from g during preparation.
- As indicated by Fig. 14 the four attempts in which g^1 participates restrict the portion of the match represented by site f^3. Similarly, the four attempts in which g^2 participates restrict the portion of the match represented by site f^4. $ppl_f = 4$ because overall these restrictions subtract the score of the match exactly four times.

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

Theorem 5. *Algorithm Border Improve solves the Border CSR problem with approximation factor* $3 + \varepsilon$.

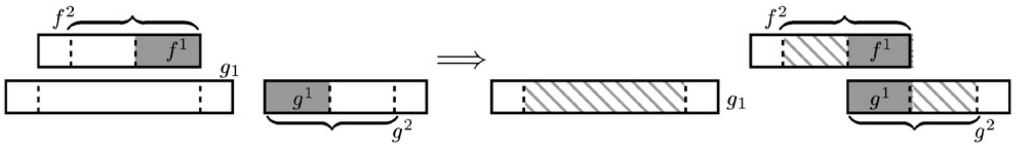

Fig. 15. In $I_2(f^1,f^2,g^1,g^2)$ improvement attempt, f is detached from g_1 when the site f^2 is prepared. After the shaded border sites are matched, TPA is run on the sites filled with slanted lines.

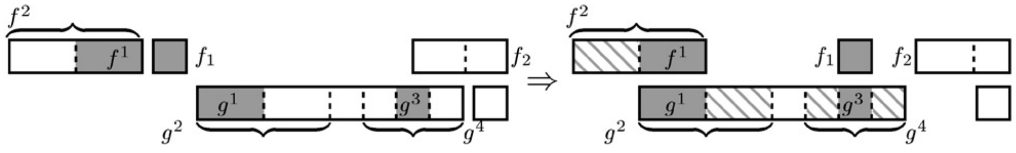

Fig. 16. $I_2(f^1,f^2,g^1,g^2)$ attempt breaks the 2-island formed by g and f_2. $I_1(f_1,g^3,g^4)$ can be combined with it. TPA runs are made on the sites filled with slanted lines.

Proof. The proof follows from inequality (5), Lemmas 10–12. □

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 contained in $f^2, g^2$, respectively
    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. 15 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). Fig. 16 shows a valid combination of attempts.

Let L be the solution generated by the CSR Improve algorithm and let *Opt* be some optimal solution. We can partition every fragment $f \in Mult(Opt)$ into plug-in sites and TPA zones using Definition 6.

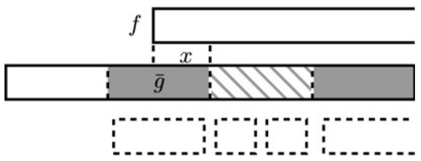

Fig. 17. f, g are the multiple fragments of a 2-island. Plug-in sites are shaded and TPA zones are filled with slanted lines. Only some of the matches are shown for clarity.

We can now construct a collection of improvement attempts $\mathscr{J}$. The I_2, I_3 attempts in $\mathscr{J}$ try to form border matches of *Opt*. An I_2 attempt of $\mathscr{J}$ with one explicit parameter, say $I_2(\bar{f})$, represents the attempt $I_2(\bar{f}, \text{zone}(\bar{f}), \bar{g}, \text{zone}(\bar{g}))$ where $(\bar{f}, \bar{g}) \in Opt$ is a border match.

Similarly, I_3 attempts can be described by two explicit parameters—the border plug-in sites of the two multiple fragments of the same 2-island. $\mathscr{J}$ is constructed as follows:

1. Suppose $f, g \in Mult(Opt)$ are the 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 $\mathscr{J}$ contains the four improvement attempts—$I_3(f^1, g^1)$, $I_3(f^1, g^2)$, $I_3(f^2, g^1)$ and $I_3(f^2, g^2)$.
2. Suppose $\bar{f}$ is a plug-in site that participates in a border match in *Opt* and is not used as a parameter in Step 1, then $\mathscr{J}$ has two improvement attempts $I_2(\bar{f})$.
3. Suppose $\bar{f}$ is a plug-in site that participates in a full match in *Opt*, then $\mathscr{J}$ has four improvement attempts $I_1(\bar{f})$. Whenever possible, these I_1 improvement attempts are combined with the I_2, I_3 attempts of Steps 1,2.

Lemma 13. *The score of each match of Opt is added exactly four times in gain*($\mathscr{J}$).

Proof. Consider a match $(\bar{f}, \bar{g}) \in Opt$ where $g \in Mult(Opt)$. g can be partitioned into plug-in sites and TPA zones. There are now three possibilities:

- $(\bar{f}, \bar{g})$ is a border match: Lemma 10 shows that $\text{MS}(\bar{f}, \bar{g})$ is added four times in $\mathscr{J}$.
- $\bar{g}$ is an inner plug-in site and $\bar{f} = f$: the 4 $I_1(\bar{g})$ attempts added in Step 3 of the construction of $\mathscr{J}$ ensure that $\text{MS}(f, \bar{g})$ is added four times in $\mathscr{J}$.
- $\bar{g}$ is a TPA site and $\bar{f} = f$: Lemma 4 and the above two items imply that TPA is run on the TPA zone containing $\bar{g}$ exactly eight times. From Lemma 2 it is easy to see that $\text{Cb}(f, Opt) = \text{MS}(f, \bar{g})$ is added four times. □

Lemma 14. *The score of each match of L is lost at most* 12 *times in gain*($\mathscr{J}$).

Proof. It follows from Lemma 11 that the score of each border match of L is lost at most 12 times.

One special instance that needs to be handled is shown in Fig. 17. As mentioned, the match of the 2-island formed by the multiple fragments f, g is broken 12 times because of I_2, I_3 attempts. However, the portion of the match marked x is also restricted in the four $I_1(\bar{g})$ attempts of $\mathscr{J}$. To prevent x from being lost 16 times overall it is necessary to combine $I_1(\bar{g})$ attempts with I_2, I_3 attempts that break the 2-island.

Analysis along the lines of Lemma 8 can now be used to show that each full match of L loses its score at most 12 times. □

Theorem 6. *Algorithm CSR Improve solves the CSR problem with approximation ratio* $3 + \varepsilon$.

Proof. The proof follows from Lemmas 13 and 14. □

Acknowledgements

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

References

[1] A. Bar-Noy, S. Guha, J. Naor, B. Schieber, Approximating the throughput of multiple machines in real-time scheduling, Proceedings of the 31st ACM STOC, Atlanta, GA, USA, 1999, pp. 622–631.
[2] P. Berman, B. DasGupta, Multi-phase algorithms for throughput maximization for real-time scheduling, J. Combin. Optim. 4 (3) (2000) 307–323.
[3] P. Berman, M. Karpinski, On some tighter inapproximability results, in: J. Wiedermann, P. van Emde Boas, M. Nielen (Eds.), Automata, Languages and Programming, ICALP'99, Lecture Notes in Computer Science, Vol. 1644, Springer, Berlin, 1999.
[4] J. Bouck, W. Miller, J. Gorrell, D. Muzny, R.A. Gibbs, Analysis of the quality and utility of random shotgun sequencing at low redundancies, Genome Res. 8 (1998) 1074–1084.
[5] B. Chandra, M.M. Halldórsson, Greedy local improvement and weighted packing approximation, SODA, 1999.
[6] G.A. Dirac, Some theorems on abstract graphs, Proc. London Math. Soc. 2 (1952) 69–81.
[7] M. McClelland, L. Florea, K. Sanderson, S. Clifton, J. Parkhill, C. Churcher, G. Dougan, R. Wilson, W. Miller, 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 (2000) 4974–4986.
[8] P. Onyango, W. Miller, J. Lehoczky, C. Leung, B. Birren, S. Wheelan, K. Dewar, A.P. Feinberg, Sequence and comparative analysis of the mouse 1 megabase region orthologous to the human 11p15 imprinted domain, Genome Res. 10 (2000) 1697–1710.
[9] F. Sanger, S. Nicklen, A.R. Coulson, DNA sequencing with chain-terminating inhibitors, Proc. Natl. Acad. Sci. USA 74 (12) (1977) 5463–5468.
[10] The Sanger Center & The Genome Sequencing Center, Towards a complete human genome sequence, Genome Res. 8 (1998) 1097–1108.
[11] J. Weber, G. Myers, Whole genome shotgun sequencing, Genome Res. 7 (1997) 401–409.