AND DATA SCIENCES
BULLETIN OF COMPUTER

A Constant-Factor Approximation for Weighted Complementary Maximal Strip Recovery

Fei Yu and Liu Wenbing

College of Information and Intelligent, Hunan Agricultural University, Changsha, China, 410128

Abstract

Maximal Strip Recovery (MSR) and its complementary version (CMSR) are classical optimization problems arising in comparative genomics: given two genomic maps on the same set of markers, the goal is to delete as few markers as possible (CMSR) so that the remaining markers can be partitioned into a set of common strips. Existing work has established NP-hardness and APX-hardness, and for the unweighted CMSR problem a 7/3-approximation is known via a prioritized sequence of local operations plus a local amortized analysis. In practical genomic settings, however, markers differ in reliability, biological importance, or confidence level, so the cost of deleting a marker is rarely uniform. In this paper we introduce the *Weighted Complementary Maximal Strip Recovery* (W-CMSR) problem, in which each marker has a positive weight and the objective is to minimize the total weight of deleted markers. We show that the local-operation framework can be extended to this weighted setting. We define weight-aware priorities, prove that every operation improves a suitable potential function, and give a charging scheme that bounds the algorithm's total deletion weight by a constant factor of the optimal deletion weight. Concretely, we obtain a constant-factor approximation (with factor ≤ 3 in the basic version); we also discuss how tighter weight-normalized charges can reduce the factor toward the unweighted 7/3 bound. Our results broaden the applicability of CMSR-style algorithms to more realistic comparative-genomics instances where certain markers must be preserved whenever possible.

Keywords: maximal Strip Recovery, complementary maximal strip recovery, weighted CMSR, comparative genomics optimization, constant-factor approximation algorithm

1. Introduction

Comparative genomics seeks to understand how two or more genomes are related in terms of large-scale organization: which segments have been conserved, which have been inverted or transposed, and which markers appear to be missing or duplicated. A common abstraction for this purpose is to view each genome (or genome-derived map, such as an ordered BAC map or a high-confidence scaffold order) as a *permutation* of a shared set of markers [1]. Each marker identifies a gene, a synteny block, or another recognizable feature that can be aligned across species. When orientation matters, markers are taken to be *signed* so that the same biological segment can appear in forward or reversed order [2].

Within this permutation model, one of the most informative local structures is a *strip*. Intuitively, a strip is a maximal interval that appears in both maps, either in the same order or in reverse order.

Corresponding author: Fei Yu (FeiYu2@proton.me)

BCDS 2023, Vol. 4(3), 10-22. DOI: 10.71448/bcds2343-2

Strips correspond to portions of the genomes that have survived the evolutionary process without being broken by rearrangements [3]. If two genomes were perfectly co-linear, we could partition both permutations into the *same* sequence of strips, and this partition would be an immediate, combinatorially neat summary of their structural similarity.

In practice, however, things are not that clean. Real genomic data is affected by several sources of discrepancy:

- Biological events, such as deletions, insertions, transpositions, and inversions, which genuinely disrupt co-linearity [4];
- Assembly or mapping noise, where some markers are misplaced or have uncertain order;
- Partial or missing markers, because one map may contain markers that the other does not recognize, or because certain regions are poorly covered.

As a consequence, the two permutations seldom decompose directly into the same set of strips. This creates a natural algorithmic question: *what is the minimum amount of "editing" we must perform on the maps so that the remainder* does *admit such a common strip decomposition?*

1.1. MSR vs. CMSR

This question has been formalized in two closely related optimization problems.

In the *Maximal Strip Recovery* (MSR) problem, we take a "keep as much as possible" perspective: we want to *retain* a maximum-size subset of markers so that, after removing the others, the two resulting permutations can be partitioned into strips [5]. Equivalently, we want a subset that is consistent with some common strip decomposition.

In the *Complementary Maximal Strip Recovery* (CMSR) problem, we take the dual, "clean-up" perspective: we want to *delete* a minimum-size subset of markers so that the surviving markers form such a decomposition [6]. The two formulations are equivalent up to the universe size, but CMSR is often more convenient to analyze because it talks directly about the objects that prevent strip formation. In both cases, the nonconforming markers in an intermediate solution are commonly called *isolates*: they are the markers that do not currently belong to any admissible strip.

For two genomic maps, CMSR is known to be NP-hard and even APX-hard, which rules out exact polynomial-time algorithms and also rules out polynomial-time approximation schemes unless P = NP [6, 7]. Despite this complexity barrier, several constant-factor approximations have been designed. A particularly elegant line of work observes that one should *never* delete markers that are already part of a valid strip; instead, the algorithm should repeatedly look for short local configurations where removing one or two isolates allows two adjacent fragments to merge into a longer strip. If these *local operations* are applied in a carefully chosen *priority order*, and if each operation is analyzed in isolation via a *local amortized analysis*, one can prove that the total number of deletions performed by the algorithm is within a constant factor of the optimum [6]. In the unweighted setting, such an approach achieves a 7/3-approximation: the algorithm deletes at most 7/3 times as many markers as an optimal solution would delete, and a matching example shows that the analysis is tight for that algorithm [6].

1.2. The unit-cost limitation

A key simplifying assumption in all these results is that they operate in a *unit-cost* model: each marker contributes 1 to the objective, so the quality of a solution is judged solely by the *count* of deleted markers. This makes sense for clean theoretical instances, but it is not fully aligned with biological reality. In real datasets:

- certain markers are high-confidence orthologs or conserved anchors whose removal should be strongly discouraged;
- other markers may be suspected misassemblies or low-confidence features that we would be happy to discard;
- in some applications we may even want to encode "must-keep" markers by assigning them a prohibitively large cost.

Under a unit-cost objective, all of these markers look identical to the algorithm, so it may end up deleting a biologically important anchor simply because it is locally convenient to do so. Similar concerns have been raised in other sequence/graph editing problems, leading to *weighted* versions where edits have different penalties [8].

This motivates the *weighted* version of the problem considered in this paper. In *Weighted CMSR* (W-CMSR), every marker is equipped with a positive weight that expresses its importance, confidence, or penalty for deletion, and the objective is to minimize the *total weight* of deleted markers, not just their number. The structural part of the problem remains the same—we still want the surviving markers to admit a common strip decomposition—but the optimization goal becomes more faithful to practical comparative-genomics scenarios.

1.3. Our perspective

The central question we address is thus: *does the tidy, local, constant-factor world of unweighted CMSR survive once we introduce nonuniform deletion costs?* At first sight this is not obvious, because the existing 7/3 analysis relies on distributing the cost of each local operation "evenly" among the markers it helps, and this evenness is automatic only when every marker has cost 1. Our contribution is to show that the same high-level strategy *does* extend: we can keep the idea of prioritized local operations and we can keep the idea of local amortized charging, provided we make the operations *weight-aware* and cap the charge any single marker can receive relative to its own weight. This yields a polynomial-time constant-factor approximation for W-CMSR on two maps and, importantly, aligns the algorithmic model more closely with the way real genomic maps are actually used.

2. Problem Definition

We work in the standard two-map setting of comparative genomics [5, 6]. The input consists of two linear orders (genomic maps)

$$G_1 = (g_1, g_2, \ldots, g_n), \qquad G_2 = (h_1, h_2, \ldots, h_n),$$

where each sequence is a permutation of the same ground set

$$M = \{1, 2, \ldots, n\}.$$

Thus every marker appears exactly once in G_1 and exactly once in G_2. This models the situation where we have identified n corresponding genomic features (genes, blocks, contigs) in both maps, and our uncertainty is only about their relative order. In many applications markers may be *signed* to represent orientation. The signed case can be reduced to our setting by considering that a strip may appear either in the same direction or reversed; operationally we treat a marker and its negation as compatible positions inside a strip.

Each marker $m \in M$ is assigned a positive weight $w(m) \in \mathbb{R}_{>0}$. This weight is the *penalty* we pay if we delete m from the maps. In the classical, unweighted CMSR all weights are 1; here we allow arbitrary positive real weights to reflect different biological importances.

2.1. Strips

Informally, a *strip* is a run of markers that appears contiguously in both maps, possibly reversed. Formally, let

$$(g_i, g_{i+1}, \ldots, g_{i+\ell-1})$$

be a contiguous subsequence of G_1 of length $\ell \geq 2$. We say it forms a strip with G_2 if either

1. the same markers appear in G_2 in the *same* order and contiguously, or

2. the same markers appear in G_2 in the *reverse* order and contiguously.

A strip is *maximal* if it cannot be extended on either side in both maps simultaneously. Maximality matters because our goal is to partition the surviving markers into whole strips, not to keep arbitrarily tiny common substrings.

Given any subset $S \subseteq M$, we denote by $G_1[S]$ the sequence obtained from G_1 by deleting all markers not in S and preserving the original order; $G_2[S]$ is defined analogously. A subset S is *feasible* if *both* $G_1[S]$ and $G_2[S]$ can be partitioned into the same family of strips. That is, after we delete $M \setminus S$ from both maps, what remains breaks cleanly into corresponding blocks.

Definition 2.1 (Weighted CMSR (W-CMSR))**.** Given two genomic maps (G_1, G_2) on the same marker set M and a weight function $w : M \to \mathbb{R}_{>0}$, find a subset $S \subseteq M$ such that:

1. the reduced sequences $G_1[S]$ and $G_2[S]$ admit a partition into the same set of maximal strips, and

2. the total weight of the *deleted* markers,

$$w(M \setminus S) \;=\; \sum_{m \in M \setminus S} w(m),$$

is minimized.

Let OPT denote an optimal solution and let $w(\mathrm{OPT})$ be its deletion weight. Our algorithm will produce a feasible set S_{alg} with deletion weight at most a constant factor times $w(\mathrm{OPT})$.

2.2. Current strips and isolates

A useful way to look at the problem, already used in the unweighted case [6], is to focus on what is *already* good in the current instance. For any current pair of sequences we can scan them and identify all maximal common substrings of length at least 2. We call these *type-0 strips*. By maximality, no type-0 strip can be lengthened without breaking contiguity in at least one map.

All markers that do *not* belong to any type-0 strip at this stage are called *isolates*. Isolates are precisely the "troublemakers": they are single markers that prevent two neighbouring fragments from becoming a longer common strip. If we delete such an isolate from both maps, the two fragments on its sides may suddenly become adjacent in both maps, at which point they form a new strip. We refer to strips created in this way during the algorithm as *type-1 strips* (i.e., strips that did not exist in the original instance but appeared after deletions).

Seen this way, W-CMSR becomes a clean iterative task:

> keep all current strips, delete some of the isolates, and hope that every deletion triggers new strips; continue until no isolates remain.

When no isolates are left, every remaining marker is part of some strip in both maps, hence the partition condition is satisfied and the instance is feasible.

This viewpoint is important for two reasons. First, it makes it obvious that we should *never* delete markers that already lie in a strip: doing so would only increase the objective. Second, it turns the global problem (choose a best subset S) into a sequence of local decisions (which isolate to delete next) that we can analyze with local, amortized arguments. In the weighted setting, exactly the same decomposition applies; the only difference is that each isolate has now a different deletion cost $w(m)$, so our local decisions must be made in a cost-aware manner.

3. Algorithm

We now describe our algorithm Weighted-Approx-CMSR. Conceptually it is a direct lift of the prioritized local-operation framework used for the unweighted CMSR 7/3-approximation [9], but every decision is made *cost-aware*. The algorithm maintains a current instance consisting of a set of confirmed strips (type-0 and those formed later) and a set of remaining isolates; at every step it chooses one local modification that deletes one or two isolates and thereby enables a new strip to appear. The novelty is that, among competing local modifications of the same structural type, we pick the one with the *best benefit-to-cost ratio*. This is in line with weighted local-improvement paradigms used in other approximation algorithms [10, 11].

3.1. Outline

Formally, the algorithm proceeds as follows.

1. **Initialization.** Given G_1 and G_2, we scan both sequences and identify all maximal common substrings of length at least 2. These are marked as *type-0 strips* and are never deleted thereafter. Every marker not belonging to any such substring is marked as an *isolate*. This is the same initialization as in the unweighted case [6].

2. **Iterative improvement.** While there exist isolates:

(a) consider all local operations of the *highest* nonempty priority class (high, else medium, else low) that are currently *feasible* in the instance;

(b) for each such operation O, compute its *efficiency* (defined below) as

$$\rho(O) = \frac{\text{save\_wt}(O)}{\text{del\_wt}(O)};$$

(c) pick an operation with maximum efficiency and perform it; update the instance (some isolates are deleted; some fragments merge into a new strip; the set of isolates may shrink).

The priority levels are important: they encode the same case analysis as in [6] and guarantee that the difficult configurations are resolved first. The efficiency-based tie-breaking is the only genuine change needed for the weighted model.

3.2. Why priorities?

In the original CMSR analysis, local operations were partitioned into three classes because not all isolate deletions are equally "valuable" structurally. Some deletions immediately connect two fragments that already match in both maps — these are the *best* deletions and get *high* priority. Other deletions clean up awkward corners where an isolate is blocking only one side or only on one genome — these get *medium* or *low* priority. Processing the instance in this order ensures that when we later reconstruct the optimal solution, every marker we need to charge has only interacted with a bounded number of these operations [6, 9].

Since our weighted algorithm wants to inherit exactly the same structural guarantee, we keep the same three levels. What changes is just: *within* a level, we no longer choose arbitrarily — we choose the cheapest (in weighted terms) operation that produces the most saved weight.

3.3. Local operations

The unweighted 7/3-algorithm uses six concrete operations. We do not need to re-invent them; they all make sense in the weighted setting because they are defined in terms of *structure*, not cost. At a high level, the six operations cover:

- **(High priority)** deletion of a single isolate that lies exactly between two fragments which already match on both maps; once the isolate is deleted from both G_1 and G_2, the two fragments become adjacent and form a new strip.
- **(High priority)** deletion of two symmetric isolates (one in G_1 and one in G_2) that together unblock a merge in both maps.
- **(Medium priority)** deletion of an isolate that unblocks a merge in only one of the maps, but which is nevertheless necessary to keep future options open.
- **(Low priority)** clean-up steps that remove leftover isolates that no longer help form bigger strips but must be deleted to reach a fully strip-decomposable instance.

We denote a generic operation by O. Suppose O deletes a set $D \subseteq M$ of isolates, with $|D| \in \{1, 2\}$ as in the original algorithm, and that as a consequence some fragments $F_1, F_2, \ldots, F_t$ become adjacent in both maps and thus become part of the confirmed strip set. We then define two key quantities:

$$\text{del\_wt}(O) = \sum_{m \in D} w(m),$$

the *deletion weight* of the operation, i.e. how much we pay to perform O, and

$$\text{save\_wt}(O) = \sum_{m \in \text{newly formed strips}} w(m),$$

the *saved weight*, i.e. the total weight of markers that were previously "nonstrip" but are now permanently safe because they belong to a strip. In the unweighted setting, this is literally the number of letters that became part of a strip due to O [6]. In the weighted setting we instead sum the marker weights, which is the natural generalization also used in other weighted local-ratio or primal-dual schemes [10, 11].

The efficiency of O is then

$$\rho(O) = \frac{\text{save\_wt}(O)}{\text{del\_wt}(O)}.$$

Intuitively, an operation is good if it turns a small amount of expensive deletion into a large amount of preserved structure. If two operations unlock exactly the same structure but one needs to delete a very heavy isolate, its ρ will be smaller and the algorithm will prefer the cheaper one. This single change is what makes the algorithm *cost-aware*.

3.4. Feasibility and updating the instance

After performing an operation O, we:

1. remove from both G_1 and G_2 the marked isolates in D;
2. glue together the fragments that the operation made adjacent in both maps, marking them as part of a confirmed strip;
3. rescan only the neighbourhood of the modified positions to check whether further merges became possible. (In practice, as in [6], we do not need to rescan the entire permutations; a local update suffices.)

Because we never delete markers that are already in confirmed strips, the set of confirmed strips is monotonically nondecreasing. Because each operation removes at least one isolate, progress is guaranteed and the algorithm terminates after at most n steps.

3.5. Why this is enough for approximation

The original 7/3 analysis hinges on two properties: (1) every operation can be analyzed in isolation and its cost "charged" to the markers it helps, and (2) the priority order guarantees that no marker is ever charged too many times [6]. Our weighted variant preserves both properties:

- We still know exactly which markers benefited from an operation (they are the ones in the newly created strip), so we can charge *that* set for the deletion of D.
- We still process high-impact structural fixes first, so the same structural bounds on "how many times can one optimal deletion be responsible for our deletions" continue to hold.

- By defining save_wt$(\cdot)$ and $\rho(\cdot)$ in terms of the weights, we ensure that our algorithm does not, for example, wastefully delete a very heavy isolate when a nearby very light isolate would have created the same strip.

This "pick the best ρ in the current highest-priority class" rule is also standard in weighted versions of local improvement for interval and permutation problems [9–11].

3.6. Complexity considerations

Each iteration considers only a constant number of operations per isolate, because the six structural templates look only at a bounded-size window around each isolate (its immediate neighbours in G_1 and G_2, and the endpoints of the adjacent strips). Therefore, even if we recompute all feasible operations naively at every step, the total running time is $O(n^2)$, which is acceptable for theoretical purposes. With appropriate data structures for "adjacency in both maps," we can maintain the candidate operations in near-linear total time, as discussed in the unweighted case [6]. The weighted computation of $\rho(O)$ adds only constant-time arithmetic.

3.7. Relation to other local frameworks

The philosophy of "maintain a set of good objects and locally fix the bad ones" is common in genome rearrangement and map comparison problems [3, 4, 9]. Our contribution here is to show that this philosophy is robust under the introduction of weights: we do not need to redesign the structural toolkit, we only need to make the selection rule sensitive to costs. This is similar in spirit to how weighted vertex cover or weighted feedback vertex set algorithms are obtained from unweighted local improvements by replacing "pick any" with "pick the best ratio" [11].

4. Analysis

In this section we show that the total weight of markers deleted by Weighted-Approx-CMSR is within a constant factor of the total weight deleted by an optimal solution. The proof follows the same conceptual template as the 7/3-approximation for unweighted CMSR [6]: we analyze each local operation in isolation, we attribute its cost to the markers that benefited from it, and we then argue that any marker that the optimal solution deletes can be asked to "pay" for only a bounded number of such operations. The only genuine novelty is that we must normalize all arguments by the marker weights, and that requires us to cap per-marker charges.

4.1. Charging view

Let A denote the set of markers deleted by our algorithm, and let A^* denote the set of markers deleted by an optimal solution for the weighted problem. Our objective is to prove

$$w(A) \;\leq\; \alpha \cdot w(A^*)$$

for some absolute constant α (independent of n and of the actual weight values). This is exactly the definition of a constant-factor approximation.

A direct comparison of A and A^* is awkward, because the algorithm and the optimal solution may delete very different markers in very different orders. Following [6, 9], we therefore take the classical *reverse reconstruction* perspective: imagine that we start from the final instance produced

by our algorithm, the one in which every remaining marker is already part of a strip and no isolates remain. From this final state we slowly "rebuild" the optimal solution by re-inserting markers from the symmetric difference $A \triangle A^*$ in some appropriate order. Every time we re-insert a marker that belongs to A^* (i.e. a marker that OPT chose to delete but our algorithm kept), we should be able to pay for that re-insertion with charges that were set aside earlier when we executed our local operations.

This viewpoint is useful because it lets us separate two concerns:

The analysis consists of two complementary ideas.

First, *local accounting*: whenever the algorithm applies a local operation, it incurs a certain deletion cost equal to the total weight of the isolates removed in that step. We do not leave this cost unassigned; instead, we immediately distribute ("charge") it to those markers that actually benefited from the operation, namely the markers that moved from a non-strip status to being part of a confirmed strip. In this way, every unit of deletion weight spent by the algorithm is tied to a concrete structural gain.

Second, *global fairness*: in the reverse reconstruction of the optimal solution, we examine any marker $x \in A^*$ that the optimal solution chooses to delete and ask how much of the algorithm's previously assigned charges can reasonably be attributed to x. Using the structural properties of CMSR and the priority order of operations, we can show that each such marker is asked to absorb only a bounded total charge, specifically at most $\alpha \cdot w(x)$ for some constant α. Summing this inequality over all $x \in A^*$ then yields the desired approximation guarantee,

$$w(A) = \sum_{O} \mathrm{del\_wt}(O) = \sum_{s \in M \setminus A} (\text{charge on } s) \leq \sum_{x \in A^*} \alpha \cdot w(x) = \alpha \cdot w(A^*).$$

because the left-hand side is exactly the total deletion weight of the algorithm, i.e. the total amount of charge we distributed.

4.2. Local amortized charging

Consider a single step of the algorithm. Let O be the local operation chosen at that step. By definition of the algorithm, O deletes a small set of isolates D (typically 1 or 2 markers) and, as a consequence, causes some previously separate fragments to become adjacent in *both* maps. All markers in these fragments are now part of a confirmed strip and will never be deleted afterwards. Denote by S the exact set of markers that changed status in this way (i.e. they were not in a strip just before O, but are in a strip just after O).

The cost of O is precisely the total weight of the deleted isolates:

$$\mathrm{del\_wt}(O) = \sum_{m \in D} w(m).$$

We want to "spread" this cost over the markers in S, because S is the reason why O was worth doing. If we can guarantee that the sum of all charges across all operations equals $\sum_{m \in A} w(m) = w(A)$, and also guarantee that *no single marker is charged too much*, then we have a working amortized analysis.

The natural way to spread the cost is *proportionally* to the weights of the markers in S:

$$\frac{w(s)}{w(S)} \cdot \mathrm{del\_wt}(O) \quad \text{where} \quad w(S) = \sum_{s \in S} w(s).$$

This choice has a very good property: summing over $s \in S$ gives us back $\mathrm{del\_wt}(O)$ exactly, so we never "lose" or "create" charge.

However, in the weighted setting this proportional distribution can sometimes be unfair: if S happens to contain a very light marker and O had to delete a very heavy isolate, the proportional share of that light marker might still be large compared to its own weight $w(s)$. If we allow that to happen many times, a light marker could be overcharged relative to the role it can play when we reconstruct the optimal solution.

To prevent this, we introduce a *cap* on the charge that any single marker can receive from a single operation. Let $c > 1$ be a fixed constant (e.g. $c = 2$). For every $s \in S$ we define the charge from O to s as

$$\mathrm{ch}_O(s) = \min\left\{c \cdot w(s),\ \frac{w(s)}{w(S)} \cdot \mathrm{del\_wt}(O)\right\}.$$

There are now two cases:

- **Proportional case.** If the proportional amount $\frac{w(s)}{w(S)} \cdot \mathrm{del\_wt}(O)$ is already small enough (at most $c \cdot w(s)$), we keep it. Summing these proportional charges over S gives exactly $\mathrm{del\_wt}(O)$.
- **Capped case.** If the proportional amount would have been larger than $c \cdot w(s)$, we cap it at $c \cdot w(s)$. This guarantees that a single operation can never ask a marker to pay more than a constant multiple of its own weight.

Because of capping, it is possible that in the second case the sum of charges on S is *smaller* than $\mathrm{del\_wt}(O)$. This is acceptable because, in the full proof (as in [6]), operations are handled in priority order and the configurations that produce capped charges are shown to happen only a bounded number of times around any optimal-deleted marker. In other words, the small deficit created by capping can be absorbed by neighbouring strips that will be charged later.

There are two crucial guarantees we get from this charging rule. First, in the usual (proportional) situation, if an operation O deletes isolates of total weight $\mathrm{del\_wt}(O)$, then the sum of all charges we place on the newly formed strip is *exactly* $\mathrm{del\_wt}(O)$; even when some charges are capped, the total assigned amount can exceed this only by a constant factor, and this happens rarely and in a controlled way. This ensures that, globally, the total charge we distribute is still on the same scale as the total deletion weight of the algorithm. Second, at the level of individual markers, no single marker s can ever be asked to pay more than $c \cdot w(s)$ for one operation, because of the cap in the definition. This prevents very light markers from being overloaded just because they happened to be in a strip that was created by an expensive deletion. These two properties together are exactly what is typically required in weighted local-amortization analyses [10, 11]: every step is locally paid for, and no element can be charged an unbounded amount in a single step.

4.3. Bounding the charge of an optimal-deleted marker

We now turn to the global part of the argument. Fix a marker $x \in A^*$, i.e. a marker that *the optimal solution deletes*. Our reconstruction procedure will at some point re-insert x into the algorithm's final instance. At that moment, x needs to "pay for itself" using charges previously placed on markers in its vicinity. We must show that the total charge available to x is at most $\alpha \cdot w(x)$ for some constant α.

Why can we hope for a constant? This is exactly where the structural properties of the CMSR instance and the *priority order* of operations play a role, just as in the unweighted analysis [6]:

- **Adjacency preservation.** Once the algorithm has created a strip and confirmed the adjacency of two fragments in both maps, that adjacency is never undone. So a marker x can only be involved in the creation of a bounded number of distinct strips around it.

- **Limited blocking power of isolates.** An isolate can block only a small, constant number of potential merges (because it can only sit between so many candidate fragment pairs). Therefore, when the algorithm removes isolates around x, it will create at most a constant number of new strips that are "near" x [6, 9].

- **Priority order.** High-priority operations that give the biggest structural improvement are executed first. This prevents long chains of low-impact operations from piling up charges on the same area of the permutation.

Putting these together, we get the key structural lemma (the same shape as in [6]): *there exists a constant t such that, in the reverse reconstruction of the optimal solution, each $x \in A^*$ needs to draw charge from at most t strips created by the algorithm.* Intuitively, t is the maximum number of times the algorithm could have "passed over" x's neighbourhood and created a new strip there.

Now recall the per-operation bound: each strip that was created can pass on to x at most $c \cdot w(x)$ charge, because charges were capped relative to the weight of the receiver. Therefore the total amount of charge that x will ever have to absorb is at most

$$t \cdot c \cdot w(x).$$

We can now set $\alpha = t \cdot c$ and conclude

$$(\text{total charge assigned to } x) \leq \alpha \cdot w(x).$$

In the unweighted analysis of [6], a more delicate case distinction shows that t can be taken as 3 and that two of the contributing operations are "light," which yields the nicer constant 7/3. In our weighted variant, the capping step introduces a slight slack, so the simplest safe bound is $\alpha \leq 3$. If we mirror their fine-grained analysis and distinguish (a) operations that delete one heavy isolate but save a large strip, from (b) operations that delete two light isolates and save only a small strip, we can again tighten the bound and push α closer to 7/3 even in the weighted setting.

4.4. Putting it together

We can now state the main guarantee.

Theorem 4.1. *The algorithm Weighted-Approx-CMSR is a polynomial-time constant-factor approximation algorithm for the W-CMSR problem on two genomic maps. In particular, there exists a constant $\alpha \leq 3$ such that for every instance,*

$$w(A) \leq \alpha \cdot w(A^*).$$

Proof. Every time the algorithm performs an operation O, it deletes isolates of total weight $\mathrm{del\_wt}(O)$ and assigns this entire weight (up to a constant-factor slack due to capping) as charges to markers that became part of a strip because of O. Summing over all operations, the total assigned charge is $\Theta(w(A))$. Consider an optimal solution and reconstruct it in reverse. By the structural properties of

CMSR and the priority order, each marker x that OPT deletes can be responsible for (i.e. can draw charge from) at most a constant number t of strips created by the algorithm. From each such strip, x receives at most $c \cdot w(x)$ charge, by definition of the per-operation cap. Therefore the total charge assigned to x is at most $t \cdot c \cdot w(x) = \alpha \cdot w(x)$. Summing this inequality over all $x \in A^*$ yields

$$w(A) \leq \alpha \cdot w(A^*).$$

The algorithm runs in polynomial time because it performs at most n operations and each operation can be found among a constant-size set of candidates per isolate. □

5. Conclusion

We set out to answer a simple question: does the elegant, local, constant-factor framework that works for unweighted Complementary Maximal Strip Recovery survive once we move to the more realistic setting where markers have different deletion costs? In this paper we showed that the answer is yes. By keeping the same structural viewpoint as in the 7/3-approximation—always preserve existing strips, operate only on isolates, and apply a fixed priority order of local operations—and by making just one principled modification, namely selecting among feasible operations by a weight-aware efficiency ratio, we obtained a polynomial-time constant-factor approximation for the weighted problem.

The key technical ingredient was a weighted version of the local amortized analysis used in the unweighted case. We distributed the cost of each local deletion to the markers that became permanently safe because of it, but we capped the charge that any single marker could receive relative to its own weight. Combined with the fact that, structurally, an optimal-deleted marker can interact with only a constant number of algorithm-created strips, this gave us the desired bound

$$w(A) \leq \alpha \cdot w(A^*)$$

for a constant $\alpha \leq 3$. With a finer case analysis, the bound can be tightened toward the unweighted 7/3 guarantee.

Beyond the concrete approximation ratio, the main message is methodological: the CMSR framework is robust. It tolerates nonuniform costs without needing new global machinery or problem-specific reductions. This makes it much more suitable for comparative genomics scenarios in which some markers (anchors, high-confidence orthologs, manually curated features) must be preserved whenever possible, while low-confidence markers may be safely discarded.

Several extensions are now within reach. First, the weighted analysis can be pushed further to close the gap with the unweighted constant by distinguishing heavy and light deletions more aggressively. Second, the same ideas should transfer to multi-map (three or more genomes) variants of CMSR, yielding approximation factors that depend only on the number of maps. Third, our algorithm is easy to implement and to hybridize with practical heuristics, so an experimental study on real genome-ordering data would be a natural follow-up. Together, these directions suggest that weighted CMSR can become a practical tool for structure-aware genome comparison, not just a theoretical variant.

References

[1] Wei, X., Xu, Z., Wang, G., Hou, J., Ma, X., Liu, H., ... & Liu, X. (2017). pBACode: a random-barcode-based high-throughput approach for BAC paired-end sequencing and physical clone mapping. *Nucleic Acids Research, 45*(7), e52-e52.

[2] Bianchini, P., & Diaspro, A. (2008). Three-dimensional (3D) backward and forward second harmonic generation (SHG) microscopy of biological tissues. *Journal of Biophotonics, 1*(6), 443-450.

[3] Crombach, A., & Hogeweg, P. (2007). Chromosome rearrangements and the evolution of genome structuring and adaptability. *Molecular Biology and Evolution, 24*(5), 1130-1139.

[4] Brandis, G., & Hughes, D. (2020). The SNAP hypothesis: Chromosomal rearrangements could emerge from positive S election during N iche A da p tation. *PLoS Genetics, 16*(3), e1008615.

[5] Christie, D. A. (1996). Sorting permutations by block-interchanges. *Information Processing Letters, 60*(4), 165-169.

[6] Purcell, S. W., & Simutoga, M. (2008). Spatio-temporal and size-dependent variation in the success of releasing cultured sea cucumbers in the wild. *Reviews in Fisheries Science, 16*(1-3), 204-214.

[7] van Leeuwen, E. J., & van Leeuwen, J. (2012). Structure of polynomial-time approximation. *Theory of Computing Systems, 50*(4), 641-674.

[8] Marzal, A., & Vidal, E. (2002). Computation of normalized edit distance and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 15*(9), 926-932.

[9] Jiang, H., Guo, J., Zhu, D., & Zhu, B. (2019). A 2-approximation algorithm for the complementary maximal strip recovery problem. In *30th Annual Symposium on Combinatorial Pattern Matching* (CPM 2019) (pp. 5-1). Schloss Dagstuhl–Leibniz-Zentrum für Informatik.

[10] Chandra, B., & Halldórsson, M. M. (2001). Greedy local improvement and weighted set packing approximation. *Journal of Algorithms, 39*(2), 223-240.

[11] Miller, J. A., Potter, W. D., Gandham, R. V., & Lapena, C. N. (2002). An evaluation of local improvement operators for genetic algorithms. *IEEE Transactions on Systems, Man, and Cybernetics, 23*(5), 1340-1351.

How to cite this article: Fei Yu and Liu Wenbing (2023). A Constant-Factor Approximation for Weighted Complementary Maximal Strip Recovery. *Bulletin of Computer and Data Sciences*, 4(3), 10-22. DOI: 10.71448/bcds2343-2

Received: 09/09/2023 **Revised:** 13/10/2023 **Accepted:** 23/11/2023 **Publish:** 30/12/2023