An Approximation Algorithm for the Minimum Co-Path Set Problem

Zhi-Zhong Chen* Guohui Lin† Lusheng Wang‡

Abstract

We present an approximation algorithm for the problem of finding a minimum set of edges in a given graph G whose removal from G leaves a graph in which each connected component is a path. It achieves a ratio of $\frac{10}{7}$ and runs in $O(n^{1.5})$ time, where n is the number of vertices in the input graph. The previously best approximation algorithm for this problem achieves a ratio of 2 and runs in $O(n^2)$ time.

Keywords: Approximation algorithms, graph algorithms, co-path sets, radiation hybrid mapping, NP-hardness.

1 Introduction

Throughout this paper, a graph means a simple undirected graph (i.e., it has neither parallel edges nor self-loops). A *co-path set* of a graph G is a set F of edges in G such that removing the edges in F from G leaves a graph in which each connected component is a path. A co-path set of G is *minimum* if its size is minimized over all co-path sets of G. We are interested in the problem of finding a minimum co-path set in a given graph. For convenience, we denote this problem by MCPS.

MCPS has applications in radiation hybrid (Rh) mapping, which is a popular and powerful technique for mapping unique DNA sequences onto chromosomes and whole genomes [3, 4, 7, 8]. Essentially, in Rh mapping experiments, chromosomes of the target organism are randomly broken into small DNA fragments through gamma radiation. A random subset of these DNA fragments incorporate or retain with healthy hamster cells and grow up to yield a hybrid cell line. The process is repeated many times and the co-retention rate of a pair of markers, which are labeled chromosomal loci, indicates their physical distance on the chromosome. The underlying mechanism

*Corresponding author. Department of Mathematical Sciences, Tokyo Denki University. Hatoyama, Saitama 350-0394, Japan. Email: zzchen@mail.dendai.ac.jp.

†Department of Computing Science, University of Alberta. Edmonton, Alberta T6G 2E8, Canada. Email: ghlin@cs.ualberta.ca.

‡Department of Computer Science, City University of Hong Kong, Tat Chee Avenue, Kowloon, Hong Kong. Email: cswangl@cityu.edu.hk.

is that, when two markers are physically close to each other on the chromosome, the probability that these two markers are broken down by the gamma radiation is low, and so with a high probability they are either co-present in or co-absent from a DNA fragment. A subset of markers that are co-present in or co-absent from Rh panels (that is, DNA fragments) is referred to as a *cluster*. Given the marker set $S = \{1, 2, \ldots, n\}$ and the collection $\mathcal{C} = \{C_1, C_2, \ldots, C_m\}$ of clusters, the *radiation hybrid map construction* problem requires the computation of a linear ordering of the markers in which the markers in each given cluster C_i appear consecutively. Due to observation errors, a cluster of markers co-occurring in DNA fragments could be mistaken and hence there may be no such required linear ordering. In such a case, we want to remove as few given clusters as possible so that there is a required ordering for the leftover clusters. This minimization problem is exactly MCPS when each given cluster is of size 2.

The maximization counterpart of MCPS is the well-known *maximum path cover* problem (MPC), which is the problem of finding a maximum subgraph H of a given graph G such that each connected component of H is a path. MPC arises in a number of applications, among others, in parallel programs, distributed systems mappings, and the code optimization problems (see [9] and the references therein). MPC is also closely related to the *travelling salesman problem with distances one and two* and several approximation algorithms have been designed for MPC [1, 9].

Unlike MPC, we are aware of only one previously known approximation algorithm for MCPS, namely, the algorithm given in [3]. The algorithm in [3] achieves a ratio of 2 and runs in $O(n^2)$ time, where n is the number of vertices in the input graph. It is based on the obvious fact that a minimum co-path set of a graph $G = (V, E)$ contains at least $\frac{1}{2}\sum_{v \in V \,:\, d_G(v) > 2}(d_G(v) - 2)$ edges, where $d_G(v)$ is the degree of v in G. In this paper, we give a better approximation algorithm for MCPS via a completely different approach. Our algorithm achieves a ratio of $\frac{10}{7}$ and runs in $O(n^{1.5})$ time. It is based on both local search and dynamic programming. In more detail, given a graph G, our algorithm computes two co-path sets S_1 and S_2 of G and outputs the smaller between the two. S_1 is computed via local search while S_2 is computed via dynamic programming. The computations of S_1 and S_2 will be closely related to each other.

The remainder of this paper is organized as follows. Section 2 gives some basic definitions and notations in graph theory that will be used throughout this paper. Section 3 presents our algorithm and its analysis. Section 4 lists several open questions related to radiation hybrid mapping.

2 Basic Definitions

Let G be a graph. We denote the vertex (respectively, edge) set of G by $V(G)$ (respectively, $E(G)$). The *degree* of a vertex v in G, denoted by $d_G(v)$, is the number of vertices adjacent to v in G. A *cycle* in G is a connected subgraph of G in which each vertex is of degree 2. A *path* in G is either a single vertex or a connected subgraph of G in which exactly two vertices are of degree 1 and the others are of degree 2. Each vertex of degree at most 1 in a path P is called an *endpoint* of P, while each vertex of degree 2 in P is called an *inner vertex* of P and each edge incident to no endpoint of P is called an *inner edge* of P. The *length* of a path P is the number of edges in P.

A *path set* of G is a subset F of $E(G)$ such that $E(G) - F$ is a co-path set of G. A *maximum path set* of G is a path set of G whose size is maximized over all path sets of G. A *path component* (respectively, *cycle component*) of G is a connected component of G that is a path (respectively, cycle). A *path-cycle cover* of G is a subgraph H of G such that $V(H) = V(G)$ and $d_H(v) \leq 2$ for every $v \in V(H)$. Note that each connected component of a path-cycle cover of G is a path or cycle. The *size* of a path-cycle cover $\mathcal{C}$ of G is the number of edges in $\mathcal{C}$. A *maximum path-cycle cover* of G is a path-cycle cover of G whose size is maximized over all path-cycle covers of G. A *matching* of G is a set of pairwise nonadjacent edges of G. A *maximum matching* of G is a matching of G whose size is maximized over all matchings of G.

Let C be a cycle of G. A *chord* of C in G is an edge $e \in E(G) - E(C)$ such that both endpoints of e appear in C. If C has no chords at all, C is *chord-free*; otherwise, it is *chord-sensitive*. An *antenna* of C in G is an edge $e \in E(G) - E(C)$ such that exactly one endpoint of e appears in C. The *foot* (respectively, *tip*) of an antenna e of C in G is the endpoint of e that appears (respectively, does not appear) in C.

For a subset U of $V(G)$, let $G - U$ denote the graph obtained from G by deleting the vertices of U and the edges incident to them, and let $G[U]$ denote the *subgraph of G induced by U*, i.e., the graph $G-(V(G)-U)$. For a set F of (unordered) pairs of vertices of G, let $G+F$ denote the graph $(V(G), E(G) \cup F)$, and let $G-F$ denote the graph $(V(G), E(G) - F)$. For a set $\mathcal{H} = \{H_1, \ldots, H_k\}$ of vertex-disjoint subgraphs of G with $\bigcup_{i=1}^{k} V(H_i) = V(G)$, *modifying G by merging each subgraph in $\mathcal{H}$ into a super-vertex* is the operation of constructing a new graph G' from G and $\mathcal{H}$, where the vertices of G' are 1, 2, ..., k and the edges of G' are those $\{i, j\} \subseteq \{1, \ldots, k\}$ such that G has an edge $\{u, v\}$ with $u \in V(H_i)$ and $v \in V(H_j)$. For each edge $e = \{i, j\}$ of G', we say that an edge $\{u, v\}$ of G *corresponds to* e if $u \in V(H_i)$ and $v \in V(H_j)$.

A *tree-decomposition* of G is a pair $(\{X_i \ : \ i \in I\}, T)$, where $\{X_i \ : \ i \in I\}$ is a family of subsets of $V(G)$ and T is a tree with $V(T) = I$ such that the following three conditions hold:

- $\bigcup_{i \in I} X_i = V$.
- For every edge $\{v, w\} \in E$, there is a subset X_i, $i \in I$ with $v \in X_i$ and $w \in X_i$.
- For all $i, j, k \in I$, if j lies on the path from i to k in T, then $X_i \cap X_k \subseteq X_j$.

The *treewidth* of a tree-decomposition $(\{X_i \ : \ i \in I\}, T)$ is $\max\{|X_i|-1 \ : \ i \in I\}$. The *treewidth* of G is the minimum treewidth of a tree-decomposition of G, taken over all possible tree-decompositions of G. It is well-known that a lot of optimization problems on graphs of fixed treewidths can be solved in polynomial time via dynamic programming.

G is *outerplanar* if it is planar and has a plane embedding in which all of its vertices appear on the same face. It is known that the treewidth of an outerplanar graph is at most 2 [2].

3 The Approximation Algorithm for MCPS

Throughout this section, fix an input graph G to MCPS and let $n = |V(G)|$. Since each minimum co-path set of G contains at least $|E(G)| - (n-1)$ edges, we may assume that $|E(G)| \leq \frac{10}{3}(n-1)$; otherwise, simply outputting $E(G)$ achieves an approximation ratio of $\frac{10}{7}$. We may also assume that G is connected; otherwise, we can use our algorithm to find a co-path set for each connected component of G and then output the union of the co-path sets. We may further assume that G is not a cycle; otherwise, an arbitrary edge of G forms a minimum co-path set of G. Due to the last two assumptions, each chord-free cycle in G has at least one antenna.

The outline of our algorithm is as follows. Given G, the algorithm first performs a preprocessing of G and then finds two co-path sets S_1 and S_2 of G. The algorithm just outputs the smaller one between S_1 and S_2.

The remainder of this section is organized as follows. Section 3.1 details the preprocessing. Sections 3.2 and 3.3 detail the computation of co-path sets S_1 and S_2, respectively. Section 3.4 analyzes the approximation ratio achieved by the algorithm.

3.1 Preprocessing

The preprocessing is based on the following lemma.

Lemma 3.1 *The following statements hold:*

1. *If C is a chord-free cycle in G having exactly one antenna in G, then for either edge e of C incident to the foot of the antenna, G has a minimum co-path set containing e.*

2. *If C is a chord-free cycle in G having exactly two antennas in G and the two antennas have the same foot, then for either edge e of C incident to the common foot of the antennas, G has a minimum co-path set containing e.*

3. *If C is a chord-free cycle in G having exactly two antennas in G and the foots of the antennas are different but adjacent, then G has a minimum co-path set containing the edge e of C between the foots of the antennas.*

Proof. Let B be an arbitrary minimum co-path set of G. Let e be as stated in the statements. If B contains the antennas of C, then B must contain exactly one edge f of C because C is a chord-free cycle in G. If $f = e$, we are done. Otherwise, modifying B by replacing f with e yields a new minimum co-path set of G. So, we may assume that B does not contain at least one antenna of C in G. We now prove the statements separately as follows.

Statement 1. Since B does not contain the antenna of C, B must contain exactly one edge f of C incident to the foot of the antenna because B is minimum and the degree of each vertex in $G - B$ is at most 2. If $f = e$, then we are done. Otherwise, modifying B by replacing f with e yields a new minimum co-path set of G.

Statement 2. If B contains neither antenna of C, then B must contain both edges of C incident to the common foot of the antennas because the degree of each vertex in $G - B$ is at most 2. So, suppose that B contains exactly one antenna of C. Then, we can prove the statement as in the proof of Statement 1.

Statement 3. If B contains exactly one of the antennas of C, then we can prove the statement as in the proof of Statement 1. So, suppose that B contains neither antenna of C. Then, since B is minimum, it must contain e. □

For convenience, for each $i \in \{1, 2, 3\}$, we call cycle C described in Statement i of Lemma 3.1 a *type-i breakable cycle* of G. A *breakable cycle* of G simply means a type-i breakable cycle of G for some $i \in \{1, 2, 3\}$. For a breakable cycle C of G, we call each edge e of C described in Lemma 3.1 a *removable edge* of C. *Breaking a breakable cycle* C in G is the operation of modifying G by removing one arbitrary removable edge of C.

Lemma 3.1 suggests that we repeat breaking a breakable cycle in G until none exists in G. After getting rid of breakable cycles, we then find a co-path set S in the leftover graph H. The final output is $S \cup R$, where R is the set of edges removed when breaking breakable cycles in G. The point is that if $|S|$ is at most r times the minimum size of a co-path set of H for some $r > 1$, then $|S \cup R|$ is at most r times the minimum size of a co-path set of G. This point is true because of Lemma 3.1.

Breaking breakable cycles in G one by one is too time consuming. That is, if we keep looking for another breakable cycle in G after breaking one, then we may have to pay a quadratic running time. So, a better way is to first find all breakable cycles of G and then break them simultaneously. We first explain how to find all breakable cycles of G in linear time. The idea is to consider $G[V_2]$, where V_2 is the set of all vertices of degree 2 in G. Obviously, $G[V_2]$ can be constructed in linear time. Each connected component of $G[V_2]$ must be a path and may be a portion of a breakable cycle of G. Conversely, for each breakable cycle C of G, there is a path component in $G[V_2]$ that is a portion of C. To decide if a path component P of $G[V_2]$ is a portion of a type-1 (respectively, type-2) breakable cycle of G, it suffices to verify whether P has two endpoints and they share a neighbor of degree 3 (respectively, 4) in G. To decide if a path component P of $G[V_2]$ is a portion of a type-3 breakable cycle of G, we distinguish two cases. In the first case, P is a single vertex; we verify whether its two neighbors in G are adjacent and both are of degree 3 in G. In the second case, P has two endpoints; we verify whether their neighbors not in P are adjacent in G and both are of degree 3 in G. Obviously, all the verifications can be done in linear total time. Consequently, all breakable cycles of G can be found in linear total time. So, it remains to show how to break them efficiently. The next lemma is obvious but useful for our purpose.

Lemma 3.2 *The following statements hold:*

1. *If two breakable cycles C_1 and C_2 of G are vertex disjoint, then C_2 remains to be a breakable cycle (of the same type) of G after C_1 is broken.*

2. *Each breakable cycle of G can share a vertex with at most one other breakable cycle of G.*

3. *If two breakable cycles C_1 and C_2 of G share exactly one vertex, then they are both of type-2 and breaking C_1 changes the type of C_2 from type-2 to type-1 but does not change the removable edges of C_2.*

4. *If two breakable cycles of G share exactly two vertices, then they are both of type-3 and the edge between the two shared vertices is the unique removable edge of both cycles, implying that breaking one of them also breaks the other.*

5. *No two breakable cycles of G can share more than two vertices.*

By Lemma 3.2, after finding all breakable cycles of G, we can get rid of them by simply removing one arbitrary removable edge from each breakable cycle. It is obvious that breaking a type-1 breakable cycle does not yield a new breakable cycle. However, breaking a type-2 or type-3 breakable cycle may yield a new breakable cycle. So, for each type-2 or type-3 breakable cycle C that has been broken, we need to continue to check if breaking C has yielded new breakable cycles and break them if there is any. We next describe the details. Suppose that C is a type-2 or type-3 breakable cycle that has been broken by removing a breakable edge $\{x, y\}$ of C, where x is the foot of an antenna of C (and so is y if C is of type-3). If C is of type-2, then we proceed by performing the following steps:

1. Let z_1 and z_2 be the two neighbors of x that are not vertices of C. (*Comment:* If in the current graph G, one of z_1 and z_2 has degree 2, the other has degree 3, and there is a path P from z_1 to z_2 such that each inner vertex of P has degree 2, then P together with the two edges $\{x, z_1\}$ and $\{x, z_2\}$ forms a type-3 breakable cycle whose breakable edge is $\{x, z_i\}$, where $1 \leq i \leq 2$ and z_i has degree 3.)

2. If in the current graph G, one of z_1 and z_2 (say, z_1) has degree 2 and the other has degree 3, then perform the following steps:

   (a) Find the longest path P such that z_1 is an endpoint of P, x is not a vertex of P, and each inner vertex of P has degree 2 in the current graph G. (*Comment:* P can be found in time linear in the length of P.)

   (b) If P is of length 3 or more, then shrink it to a new path of length 2 without changing its endpoints. (*Comment:* We need to memorize the correspondence between the old path P and the new path because we need to recover the old path after the preprocessing. For convenience, we hereafter still use P to denote the new shrunk path.)

   (c) If z_2 is also an endpoint of P, then remove edge $\{x, z_2\}$ from G and stop looking for new breakable cycles. (*Comment:* Performing this step does not yield a new breakable cycle.)

On the other hand, if C is of type-3, then we proceed by performing the following steps:

3. Let $\{x_1, x_2\}$ be the edge removed by breaking C. (*Comment:* The leftover of C in the current graph G is a path from x_1 to x_2.)

4. For each $i \in \{1, 2\}$, let P_i be the longest path in the current graph G such that x_i is an endpoint of P_i and each inner vertex of P_i is not a vertex of C but has degree 2 in the current graph G. (*Comment:* P_i can be found in time linear in the length of P_i.)

5. For each $i \in \{1, 2\}$, if P_i is of length 3 or more, then shrink it to a new path of length 2 without changing its endpoints. (*Comment:* We need to memorize the correspondence between the old path and the new path because we need to recover the old path after the preprocessing. For convenience, we hereafter still use P_i to denote the new shrunk path.)

6. For each $i \in \{1, 2\}$, let y_i be the endpoint of P_i other than x_i. (*Comment:* If $y_1 = y_2$ and the degree of y_1 in the current graph G is 3 or 4, the leftover of C together with P_1 and P_2 forms a new type-1 or type-2 breakable cycle. Similarly, if $y_1 \neq y_2$, $\{y_1, y_2\} \in E(G)$, and both y_1 and y_2 have degree 3 in the current graph G, then the leftover of C together with P_1 and P_2 forms a new type-3 breakable cycle.)

7. If $y_1 = y_2$ and the degree of y_1 in the current graph G is 3, then remove one of the two edges adjacent to y_1 from G and stop looking for new breakable cycles. (*Comment:* If the condition in this step holds, we break a new type-1 breakable cycle without yielding a new breakable cycle.)

8. If $y_1 = y_2$ and the degree of y_1 in the current graph G is 4, then perform the following two steps (which are similar to Steps 1 and 2):

   (a) Let z_1 and z_2 be the two neighbors of y_1 that are not vertices of P_1 or P_2.

   (b) If in the current graph G, one of z_1 and z_2 (say, z_1) has degree 2 and the other has degree 3, then perform the following steps:

      i. Find the longest path P such that z_1 is an endpoint of P, y_1 is not a vertex of P, and each inner vertex of P has degree 2 in the current graph G.

      ii. If P is of length 3 or more, then shrink it to a new path of length 2 without changing its endpoints. (*Comment:* We need to memorize the correspondence between the old path P and the new path because we need to recover the old path after the preprocessing. For convenience, we hereafter still use P to denote the new shrunk path.)

      iii. If z_2 is also an endpoint of P, then remove edge $\{y_1, z_2\}$ from G and stop looking for new breakable cycles.

9. If $y_1 \neq y_2$, $\{y_1, y_2\} \in E(G)$, and both y_1 and y_2 have degree 3 in the current graph G, then remove edge $\{y_1, y_2\}$ from G and go to Step 3 by viewing the newly broken (type-3) breakable cycle as C.

Since the paths P, P_1, and P_2 in the above steps can be found in time linear in their lengths and we shrink them to a length-2 new path if they are of length 3 or more, finding and breaking

the new breakable cycles by the above steps can be done in linear total time. The old paths that were shrunk can be recovered in linear total time in the reverse order in which they were shrunk.

As the final step of the preprocessing, our algorithm computes a maximum path-cycle cover $\mathcal{C}$ in G. If $\mathcal{C}$ is connected, then it is either a Hamiltonian path or cycle of G. In the former case, $E(G) - E(\mathcal{C})$ is clearly a minimum co-path set of G. In the latter case, after the removal of one arbitrary edge from $\mathcal{C}$, $E(G) - E(\mathcal{C})$ is again a minimum co-path set of G. So, we may further assume that $\mathcal{C}$ is disconnected. Then, every cycle of $\mathcal{C}$ has at least one antenna in G.

Throughout the remainder of this section (i.e., Section 3), we assume that our algorithm has completed the preprocessing. As an immediate consequence of this assumption, G has no breakable cycles.

The next lemma will help us compute S_2 in Section 3.3.

Lemma 3.3 *G has a minimum co-path set B such that for every chord-free cycle C in $\mathcal{C}$ with exactly two antennas in G, B contains at least one antenna of C in G.*

Proof. Consider an arbitrary minimum co-path set B of G. Suppose that C is a chord-free cycle in $\mathcal{C}$ with exactly two antennas in G such that B contains neither antenna of C in G. Let e be one of the antennas of C in G. Since B is a co-path set of G, B contains at least one edge e' of C incident to the foot of antenna e. We modify B by removing e' and adding e. After this modification, B remains to be a minimum co-path set of G (because C is not a breakable cycle of G), and contains at least one antenna of C in G. By repeating this modification, we can ensure that for every chord-free cycle C in $\mathcal{C}$ with exactly two antennas in G, B contains at least one antenna of C in G. □

Throughout the remainder of this paper, fix a minimum co-path set B of G such that for every chord-free cycle C in $\mathcal{C}$ with exactly two antennas in G, B contains at least one antenna of C in G. B exists because of Lemma 3.3.

Related to $\mathcal{C}$, we next define several notations that will be used frequently hereafter.

- Let p be the number of path components in $\mathcal{C}$.
- Let k be the number of cycles in $\mathcal{C}$.
- Let c be the number of chord-sensitive cycles in $\mathcal{C}$.

Lemma 3.4 $|B| \geq |E(G)| - |E(\mathcal{C})|$.

Proof. Since $G - B$ is a path-cycle cover of G and $\mathcal{C}$ is a maximum path-cycle cover of G, we have $|E(G)| - |B| \leq |E(\mathcal{C})|$. Hence, $|B| \geq |E(G)| - |E(\mathcal{C})|$. □

3.2 Computing Co-Path Set S_1

An obvious way to obtain S_1 is to transform $\mathcal{C}$ into a path set of G and then set $S_1 = E(G) - E(\mathcal{C})$. To transform $\mathcal{C}$ into a path set of G, we can simply remove one arbitrary edge e_C from each cycle C in $\mathcal{C}$. By Lemma 3.4, S_1 computed in this way has size $\leq |B| + k$ and is hence close to $|B|$ if k is small. Indeed, one can easily observe that $|B| \geq k$. So, computing S_1 in this simple way achieves an approximation ratio of 2.

To obtain a smaller S_1, our idea is to break the cycles in $\mathcal{C}$ more carefully. In other words, we cannot afford to break the cycles in $\mathcal{C}$ independently. Rather, we try to find a collection $\mathcal{K}$ of two or three connected components of $\mathcal{C}$ such that we can transform $\mathcal{K}$ into a collection $\mathcal{K}'$ of vertex-disjoint paths with $|E(\mathcal{K}')| \geq |E(\mathcal{K})| - (\kappa - 1)$, where κ is the number of cycles in $\mathcal{K}$. For example, if $\mathcal{K}$ is a collection of two cycle components C_1 and C_2 of $\mathcal{C}$ such that $E(G) - E(\mathcal{C})$ contains an edge $e = \{u_1, u_2\}$ with $u_1 \in V(C_1)$ and $u_2 \in V(C_2)$, then we can let $\mathcal{K}' = (\mathcal{K} - \{e_1, e_2\}) + \{e\}$, where e_1 (respectively, e_2) is one of the two edges of C_1 (respectively, C_2) incident to u_1 (respectively, u_2).

We next proceed to the details of the computation of S_1. Since we want to save $\mathcal{C}$ for later use, we first make a copy $\mathcal{D}$ of $\mathcal{C}$. We then compute S_1 by modifying $\mathcal{D}$ as follows:

1. Initialize a set $M = \emptyset$.
2. While G has a *good edge related to* $\mathcal{D}$, i.e., an edge $\{u, v\} \in E(G) - E(\mathcal{D})$ such that u and v appear in different cycle components in $\mathcal{D}$, add edge $\{u, v\}$ to M and modify $\mathcal{D}$ by removing one of the two edges in $\mathcal{D}$ incident to u, removing one of the two edges in $\mathcal{D}$ incident to v, and adding edge $\{u, v\}$.
3. Initialize a set $M_1 = M$. (*Comment:* M_1 will be used to compute S_2 later.)
4. While G has a *superb edge related to* $\mathcal{D}$, i.e., an edge $\{u, v\} \in E(G) - E(\mathcal{D})$ such that $d_{\mathcal{D}}(u) \leq 1$ and v appears in a cycle component of $\mathcal{D}$, add edge $\{u, v\}$ to M_1 and further modify $\mathcal{D}$ by deleting one of the two edges in $\mathcal{D}$ incident to v and adding $\{u, v\}$.
5. While G has a *superb 4-cycle related to* $\mathcal{D}$, i.e., a 4-cycle consisting of four edges $\{u_1, v_1\}$, $\{v_1, v_2\}$, $\{v_2, u_2\}$, and $\{u_2, u_1\}$ in G such that $\{v_1, v_2\}$ is an edge of a path component of $\mathcal{D}$ and $\{u_1, u_2\}$ is an edge of a cycle component of $\mathcal{D}$, add both edges $\{u_1, v_1\}$ and $\{u_2, v_2\}$ to M_1 and further modify $\mathcal{D}$ by deleting one of the two edges in $\mathcal{D}$ incident to u_1, deleting one of the two edges in $\mathcal{D}$ incident to u_2, deleting edge $\{v_1, v_2\}$, and adding edges $\{u_1, v_1\}$ and $\{u_2, v_2\}$. (*Comment:* No superb edge related to $\mathcal{D}$ will be created in this step because of Step 2.)
6. While G has a *good 3-path related to* $\mathcal{D}$, i.e., a path consisting of three edges $\{u_1, v_1\}$, $\{v_1, v_2\}$, and $\{v_2, u_2\}$ in G such that $\{v_1, v_2\}$ is an edge of a path component of $\mathcal{D}$ and u_1 and u_2 fall into different cycle components of $\mathcal{D}$, add both edges $\{u_1, v_1\}$ and $\{u_2, v_2\}$ to M_1 and further modify $\mathcal{D}$ by deleting one of the two edges in $\mathcal{D}$ incident to u_1, deleting one of the two edges in $\mathcal{D}$ incident to u_2, deleting edge $\{v_1, v_2\}$, and adding edges $\{u_1, v_1\}$ and $\{u_2, v_2\}$. (*Comment:* Neither superb edge nor superb 4-cycle related to $\mathcal{D}$ will be created in this step because of Step 2.)

7. For each cycle C in $\mathcal{D}$, delete an arbitrary edge from C.

8. Let $S_1 = E(G) - E(\mathcal{D})$.

In order to analyze $|S_1|$, we define the following notations :

- Let σ be the number of cycles broken in Step 4 or 5.
- Let 2γ be the number of cycles broken in Step 6.
- Let c' be the number of chord-sensitive cycles in $\mathcal{D}$ immediately before Step 7.
- Let b be the number of chord-free cycles in $\mathcal{D}$ with exactly two antennas immediately before Step 7.
- Let t be the number of chord-free cycles in $\mathcal{D}$ with at least three antennas immediately before Step 7.

The next lemma follows from Lemma 3.4 and the definitions of the notations immediately.

Lemma 3.5 $|S_1| = |E(G)| - |E(\mathcal{C})| + k - |M| - \sigma - \gamma \leq |B| + k - |M| - \sigma - \gamma$. *Moreover,* $k = 2|M| + \sigma + 2\gamma + c' + b + t$.

Lemma 3.6 *M_1 does not contain two edges $\{u_1, v_1\}$ and $\{u_2, v_2\}$ such that one connected component of $\mathcal{C}$ contains both u_1 and u_2 and another contains both v_1 and v_2. Moreover, the graph obtained by modifying $\mathcal{C} + M_1$ by merging each connected component of $\mathcal{C}$ into a super-vertex is a forest.*

Proof. Let e_1, e_2, ..., e_ℓ be the edges of M_1. We may assume that for each $i \in \{1, 2, \ldots, \ell - 1\}$, e_i was added to M_1 by our algorithm (in Steps 3 through 6) no later than e_{i+1}. Imagine that we construct the graph $K = \mathcal{C} + M_1$ by initializing $K = \mathcal{C}$ and then adding the edges e_1, e_2, ..., e_ℓ to K one by one in this order. Obviously, for each $i \in \{1, 2, \ldots, \ell - 1\}$, when we add e_i to K, at least one endpoint of e_i appears in a connected component of K that contains none of e_1, e_2, ..., e_{i-1}. This fact implies the lemma immediately. □

Lemma 3.7 *The above computation of S_1 can be implemented in linear time.*

Proof. Throughout this proof, the data structure for representing a graph H is always its adjacency list, where (i) for each $u \in V(H)$, the list L_u of neighbors of u in H is doubly-linked and (ii) for each $\{u, v\} \in E(H)$, the copy of v in L_u and the copy of u in L_v point to each other (via two pointers). This data structure enables us to delete a vertex u from H in time linear in $d_H(u)$.

It is easy to see that each step other than Steps 2, 4, 5, and 6 can be implemented in linear time. For convenience, for each $i \in \{2, 4, 5, 6\}$, let $\mathcal{D}_i$ be the graph $\mathcal{D}$ immediately before Step i. We may assume that the vertices in $\mathcal{D}_2$ are the integers 1, 2, ..., n, where for each cycle component

C in $\mathcal{D}_2$, if u is the smallest vertex of C, then the vertices of C are u, $u+1$, $\ldots$, $u+|V(C)|-1$ and they appear in C clockwise in this order. Let the cycles in $\mathcal{D}_2$ be C_1, C_2, $\ldots$, C_k.

To implement Step 2, we construct a graph $\mathcal{H}_2$ whose vertices are 1, 2, $\ldots$, k and whose edges are those $\{i,j\} \subseteq \{1,\ldots,k\}$ such that G has an edge $\{u,v\}$ with $u \in V(C_i)$ and $v \in V(C_j)$. For each edge $\{i,j\}$ of $\mathcal{H}_2$, we also record an (arbitrary) edge $\{u,v\} \in E(G)$ with $u \in V(C_i)$ and $v \in V(C_j)$ and call it the *good edge corresponding to* $\{i,j\}$. Now, to implement Step 2, we just repeat the following two steps in turn until $E(\mathcal{H}_2)$ becomes empty:

- Choose an arbitrary edge $\{i,j\}$ in $\mathcal{H}_2$ and use the good edge $\{u,v\}$ corresponding to $\{i,j\}$ to modify M and $\mathcal{D}$ as in Step 2.
- Remove vertices i and j from $\mathcal{H}_2$.

To implement Step 4, we construct a bipartite graph $\mathcal{H}_4 = (X_4 \cup Y_4, E_4)$, where X_4 consists of the endpoints of the path components in $\mathcal{D}_4$, Y_4 consists of all $i \in \{1,\ldots,k\}$ such that C_i is a cycle component in $\mathcal{D}_4$, and E_4 consists of those $\{u,i\}$ such that $u \in X_4$, $i \in Y_4$, and C_i has a vertex v with $\{u,v\} \in E(G)$. For each edge $\{u,i\}$ of $\mathcal{H}_4$, we also record an (arbitrary) edge $\{u,v\} \in E(G)$ with $v \in V(C_i)$ and call it the *superb edge corresponding to* $\{u,i\}$. Now, to implement Step 4, we just repeat the following two steps in turn until $E(\mathcal{H}_4)$ becomes empty:

- Choose an arbitrary edge $\{u,i\}$ in $\mathcal{H}_4$ and use the superb edge $\{u,v\}$ corresponding to $\{u,i\}$ to modify M and $\mathcal{D}$ as in Step 4.
- Remove vertices u and i from $\mathcal{H}_4$.

To implement Step 5, we construct a bipartite graph $\mathcal{H}_5 = (X_5 \cup Y_5, E_5)$, where X_5 consists of the inner edges of the path components in $\mathcal{D}_5$, Y_5 consists of all $i \in \{1,\ldots,k\}$ such that C_i is a cycle component in $\mathcal{D}_5$, and E_5 consists of those $\{\{v_1,v_2\},i\}$ such that $\{v_1,v_2\} \in X_5$, $i \in Y_5$, and G has two edges $\{u_1,v_1\}$ and $\{u_2,v_2\}$ with $\{u_1,u_2\} \in E(C_i)$. For each edge $\{\{v_1,v_2\},i\}$ of $\mathcal{H}_5$, we also record a pair of (arbitrary) edges $\{u_1,v_1\}$ and $\{u_2,v_2\}$ of G with $\{u_1,u_2\} \in E(C_i)$ and call it the *superb pair corresponding to* $\{\{v_1,v_2\},i\}$. Now, to implement Step 5, we just repeat the following two steps in turn until $E(\mathcal{H}_5)$ becomes empty:

- Choose an arbitrary edge $\{\{v_1,v_2\},i\}$ in $\mathcal{H}_5$ and use the superb pair $(\{u_1,v_1\},\{u_2,v_2\})$ corresponding to $\{\{v_1,v_2\},i\}$ to modify M and $\mathcal{D}$ as in Step 5.
- Remove vertices $\{v_1,v_2\}$ and i from $\mathcal{H}_5$.

We next explain how to construct $\mathcal{H}_5$ in linear time, because it does not look so obvious. Clearly, we can set up a (random-access) array $\mathcal{A}$ of size n such that for each $u \in \{1,2,\ldots,n\}$, $\mathcal{A}[u] = 0$ if vertex u does not appear in a cycle component of $\mathcal{D}_5$, and $\mathcal{A}[u] = i$ if C_i is the cycle component of $\mathcal{D}_5$ with $u \in V(C_i)$. We also set up a (random-access) array $\mathcal{A}'$ of size n whose elements are all 0. We further compute the minimum vertex n_i and the maximum vertex x_i in C_i for every $i \in Y_5$. Now, consider an arbitrary $\{v_1,v_2\} \in X_5$. It suffices to show that we can find all $i \in Y_5$ with

$\{\{v_1, v_2\}, i\} \in E_5$ in $d_G(v_1) + d_G(v_2)$ total time. To this end, for each neighbor u_1 of v_1 in G, we set $\mathcal{A}'[u_1] = 1$. We then check, for each neighbor u_2 of v_2 in G with $\mathcal{A}[u_2] > 0$, whether (1) $u_2 < x_{\mathcal{A}[u_2]}$ and $\mathcal{A}'[u_2 + 1] = 1$, (2) $u_2 > n_{\mathcal{A}[u_2]}$ and $\mathcal{A}'[u_2 - 1] = 1$, (3) $u_2 = x_{\mathcal{A}[u_2]}$ and $\mathcal{A}'[n_{\mathcal{A}[u_2]}] = 1$, or (4) $u_2 = n_{\mathcal{A}[u_2]}$ and $\mathcal{A}'[x_{\mathcal{A}[u_2]}] = 1$. If the checking succeeds for some neighbor u_2 of v_2 in G with $\mathcal{A}[u_2] > 0$, then $\{\{v_1, v_2\}, \mathcal{A}[u_2]\}$ is an edge of $\mathcal{H}_5$. After scanning the neighbors of v_2 in G in this way, we should have found the edges of $\mathcal{H}_5$ adjacent to vertex $\{v_1, v_2\}$ and then we reset $\mathcal{A}'[u_1] = 0$ for every neighbor u_1 of v_1 in G.

To implement Step 6, we construct a bipartite graph $\mathcal{H}_6 = (X_6 \cup Y_6, E_6)$, where X_6 consists of the inner vertices of the path components in $\mathcal{D}_6$, Y_6 consists of all $i \in \{1, \ldots, k\}$ such that C_i is a cycle component in $\mathcal{D}_6$, and E_6 consists of those $\{u, i\}$ such that $u \in X_6$, $i \in Y_6$, and C_i has a vertex v with $\{u, v\} \in E(G)$. For each edge $\{v, i\}$ of $\mathcal{H}_6$, we also record an (arbitrary) edge $\{u, v\}$ of G with $u \in V(C_i)$ and call it the *representative edge corresponding to* $\{v, i\}$. Now, to implement Step 5, we let $\mathcal{L}$ be the list of the inner edges of the path components of $\mathcal{D}_6$ and repeat the following three steps in turn until $\mathcal{L}$ becomes empty:

- Choose an arbitrary edge $\{v_1, v_2\}$ in $\mathcal{L}$.
- If either (1) one of $d_{\mathcal{H}_6}(v_1)$ and $d_{\mathcal{H}_6}(v_2)$ is larger than 1 and the other is larger than 0, or (2) $d_{\mathcal{H}_6}(v_1) = 1$ and $d_{\mathcal{H}_6}(v_2) = 1$ but the neighbor of v_1 in $\mathcal{H}_6$ is different from that of v_2 in $\mathcal{H}_6$, then perform the following three steps in turn:
  - Find different C_i and C_j with $\{v_1, i\} \in E(\mathcal{H}_6)$ and $\{v_2, j\} \in E(\mathcal{H}_6)$ (by checking at most two neighbors of v_h in $\mathcal{H}_6$ for each $h \in \{1, 2\}$).
  - Use edge $\{v_1, v_2\}$, the representative edge $\{u_1, v_1\}$ corresponding to $\{v_1, i\}$, and the representative edge $\{u_2, v_2\}$ corresponding to $\{v_2, j\}$ to modify M and $\mathcal{D}$ as in Step 6.
  - Remove vertices i and j from $\mathcal{H}_6$.
- Remove $\{v_1, v_2\}$ from $\mathcal{L}$.

Obviously, the above three steps can be done in $O(d_{\mathcal{H}_6}(i) + d_{\mathcal{H}_6}(j))$ time if C_i and C_j are found in the second step, while in $O(1)$ time otherwise. □

3.3 Computing Co-Path Set S_2

As mentioned in Section 3.2, $|S_1|$ is close to $|B|$ when k is small. So, we want to compute S_2 in such a way that $|S_2|$ is close to $|B|$ when k is large. Recall that we can compute a maximum path set in a given graph with bounded tree-width in linear time. With this in mind, we will compute S_2 roughly as follows:

- Compute a set F of edges in $E(G) - E(\mathcal{C})$ such that $\mathcal{C} + F$ is a connected graph with bounded tree-width. (*Comment:* Since $\mathcal{C} + F$ is connected, $|F| \geq k + p - 1$. So, $|F|$ is large when k is large.)

- Compute a maximum path set P in $\mathcal{C}+F$ and output $E(G)-E(P)$.

How to compute F? This is the crux of computing S_2. A simple but smart way is to let F be an arbitrary set of edges in $E(G)-E(\mathcal{C})$ such that modifying $\mathcal{C}+F$ by merging each connected component of $\mathcal{C}$ into a super-vertex yields a tree. Obviously, such a set F contains exactly $k+p-1$ edges and $\mathcal{C}+F$ is an outerplanar graph (and hence its tree-width is at most 2). We can prove that computing F in this way leads to an approximation ratio of 1.5. We omit the proof here because we can do even better by choosing the edges in F more carefully (as shown below).

Our construction of F can be sketched as follows. F will be the union of six sets M_1, M_2, ..., M_6, where M_1 is the set computed in Section 3.2. By Lemma 3.6, modifying $\mathcal{C}+M_1$ by merging each connected component of $\mathcal{C}$ into a super-vertex yields a forest. The three sets M_2, M_3, and M_4 will be carefully computed so that $\sum_{i=1}^{4}|M_i|=k+p-1$ and modifying $\mathcal{C}+(\bigcup_{i=1}^{4}M_i)$ by merging each connected component of $\mathcal{C}$ into a super-vertex yields a tree. This implies that $\mathcal{C}+(\bigcup_{i=1}^{4}M_i)$ is outerplanar. Of special interest is the construction of M_4 which will be randomized. To construct M_5, we will start with $M_5=\emptyset$ and then, for every chord-sensitive cycle C in $\mathcal{C}$, add one (arbitrary) chord of C in G to M_5. In this way, $\mathcal{C}+(\bigcup_{i=1}^{5}M_i)$ remains to be outerplanar. M_6 will contain at most one edge; its role is simply to help us achieve an approximation ratio of $\frac{10}{7}$ (instead of $\frac{10}{7}-\epsilon$ for some arbitrarily small constant $\epsilon>0$).

We next detail the computation of S_2. For convenience, we say that a cycle C in $\mathcal{C}$ is *bad* if it is a chord-free cycle with exactly two antennas in G and there is no edge $\{u,v\}\in M_1$ with $\{u,v\}\cap V(C)\neq\emptyset$. Note that each bad cycle in $\mathcal{C}$ is also a cycle component of $\mathcal{D}$ immediately before Step 7. We compute S_2 as follows:

9. Let Z be the set of vertices that appear in bad cycles in $\mathcal{C}$.
10. Construct an auxiliary graph H_1 by modifying $G-Z$ by merging each connected component of $(\mathcal{C}+M_1)-Z$ into a super-vertex.
11. Compute a spanning forest T_1 of H_1.
12. Let M_2 be the set obtained from $E(T_1)$ by replacing each edge $e\in E(T_1)$ with an edge $\{u,v\}\in E(G)$ corresponding to e.
13. Construct an auxiliary graph H_2 as follows:
    (a) The vertices of H_2 one-to-one correspond to the connected components of $(\mathcal{C}+(M_1\cup M_2))-Z$.
    (b) For each pair $\{s_1,s_2\}$ of vertices of H_2, $\{s_1,s_2\}$ is an edge of H_2 if and only if there is a bad cycle C in $\mathcal{C}$ such that the tip of one antenna of C in G appears in the connected component of $(\mathcal{C}+(M_1\cup M_2))-Z$ corresponding to s_1 and the tip of the other antenna of C in G appears in the connected component of $(\mathcal{C}+(M_1\cup M_2))-Z$ corresponding to s_2.

14. Compute a spanning tree T_2 of H_2. (*Comment:* Note that H_2 is connected because so is G.)

15. Let M_3 be the set obtained from $E(T_2)$ by replacing each edge $\{s_1, s_2\} \in E(T_2)$ with the two antennas e_1 and e_2 of the cycle C in $\mathcal{C}$ such that the tip of e_1 appears in the connected component of $(\mathcal{C} + (M_1 \cup M_2)) - Z$ corresponding to s_1 and the tip of e_2 appears in the connected component of $(\mathcal{C} + (M_1 \cup M_2)) - Z$ corresponding to s_2.

16. Construct a graph H_3 from $\mathcal{C}$ and T as follows:

    (a) Initially, set $H_3 = \mathcal{C} + (M_1 \cup M_2 \cup M_3)$.

    (b) For each bad cycle C of $\mathcal{C}$ that is also a connected component of H_3, choose one antenna of C in G uniformly at random and add it to H_3. (*Comment:* M_4 is the set of edges added to H_3 in this step.)

    (c) For each chord-sensitive cycle C in $\mathcal{C}$, add one chord of C in G to H_3. (*Comment:* After this step, H_3 is outerplanar and so its tree-width is at most 2. Moreover, M_5 is the set of edges added to H_3 in this step.)

17. If $E(G) - E(H_3) \neq \emptyset$, add one (arbitrary) edge of $E(G) - E(H_3)$ to H_3. (*Comment:* After this step, the tree-width of H_3 is at most 3. Moreover, if one edge is added to H_3 in this step, M_6 consists of this edge only; otherwise, M_6 is empty.)

18. Compute a maximum path set P in H_3. (*Comment:* Since the tree-width of H_3 is at most 3, this step can be done in linear time via a standard dynamic programming.)

19. Let $S_2 = E(G) - E(P)$.

Obviously, if no edge is added to H_3 in Step 17, then S_2 is a minimum co-path set of G and we are done. So, we hereafter assume that one edge is added to H_3 in Step 17. For a random variable x, let $\mathcal{E}[x]$ denote the expected value of x. Then, we have the following lemma:

Lemma 3.8 $\mathcal{E}[|S_2|] \leq |B| + |E(G)| - |E(\mathcal{C})| - k - c - p - \max\{0, \frac{1}{2}(b - |M| - p + 1)\}$.

Proof. Since $E(H_3) - B$ is a path set of H_3 and P is a maximum path set of H_3, $|E(H_3)| - |E(H_3) \cap B| \leq |E(P)| = |E(G)| - |S_2|$. So, $|E(H_3) \cap B| \geq |S_2| + |E(H_3)| - |E(G)| = |S_2| - |E(G)| + |E(\mathcal{C})| + k + c + p$. Hence, $\mathcal{E}[|E(H_3) \cap B|] \geq \mathcal{E}[|S_2|] - |E(G)| + |E(\mathcal{C})| + k + c + p$.

We claim that $\mathcal{E}[|(E(G) - E(H_3)) \cap B|] \geq \max\{0, \frac{1}{2}(b - |M| - p + 1)\}$. This claim together with the last inequality in the last paragraph implies the lemma. To see the claim, first observe that $\mathcal{D}$ has exactly $p + |M| + c' + b + t$ connected components immediately before Step 7. So, $(\mathcal{C} + (M_1 \cup M_2)) - Z$ has at most $p + |M|$ connected components because G is connected and there is no edge between any pair of cycles in $\mathcal{D}$ immediately before Step 7. Thus, $|E(T_2)| \leq p + |M| - 1$, implying that at least $\max\{0, b - (p + |M| - 1)\}$ edges were added to H_3 in Step 16b. By Lemma 3.3 and our choice of B, for each edge e added to H_3 in Step 16b, the probability that $e' \in B$ is at least $\frac{1}{2}$, where e' is the other antenna of the bad cycle one of whose antennas is e. This finishes the proof of the claim and hence the lemma. □

Obviously, the above computation of S_2 can be implemented in linear time. Note that each random choice in Step 16b uses one random bit. The proof of Lemma 3.8 remains valid, even if the random bits used in Step 16b are mutually dependent (i.e., only one random bit is generated and each random choice in Step 16b uses this random bit). So, we can easily derandomize the above computation of S_2. In more details, we modify Steps 16 through 19 as follows:

16. Construct two graphs H_3 and H_3' from $\mathcal{C}$ and T as follows:
    (a) Initially, set $H_3 = H_3' = \mathcal{C} + (M_1 \cup M_2 \cup M_3)$.
    (b) For each bad cycle C of $\mathcal{C}$ that is also a connected component of H_3, add one antenna of C in G to H_3 and add the other to H_3'.
    (c) For each chord-sensitive cycle C in $\mathcal{C}$, add one chord of C in G to both H_3 and H_3'.
17. If $E(G) - E(H_3) \neq \emptyset$, add one (arbitrary) edge of $E(G) - E(H_3)$ to H_3. Similarly, if $E(G) - E(H_3') \neq \emptyset$, add one (arbitrary) edge of $E(G) - E(H_3')$ to H_3'.
18. Compute a maximum path set P in H_3 and compute a maximum path set P' in H_3'.
19. If $|E(P)| \geq |E(P')|$, then let $S_2 = E(G) - E(P)$; otherwise, let $S_2 = E(G) - E(P')$.

Therefore, in the next section, we will assume that our algorithm has been derandomized.

3.4 Analysis of the Algorithm

We start by proving a crucial lower bound on $|B|$:

Lemma 3.9 $|B| \geq 4t + 3b + 3c'$.

PROOF. For convenience, let $\mathcal{D}_7$ be the graph $\mathcal{D}$ immediately before Step 7. The idea behind the proof is to find a set $\Gamma(C)$ of edges in G for every cycle C in $\mathcal{D}_7$ satisfying the following conditions:

(a) The sets $\Gamma(C)$ for the cycles C in $\mathcal{D}_7$ are disjoint.

(b) If C is chord-sensitive, then $|\Gamma(C) \cap B| \geq 3$.

(c) If C is chord-free and has exactly two antennas in G, then $|\Gamma(C) \cap B| \geq 3$.

(d) If C is chord-free and has at least three antennas in G, then $|\Gamma(C) \cap B| \geq 4$.

Because of Conditions (a) through (d), the existence of $\Gamma(C)$ implies the lemma immediately. So, it remains to show the existence of $\Gamma(C)$. For the cycles C of $\mathcal{D}_7$, we construct $\Gamma(C)$ by performing the following steps:

(1) For each cycle C in $\mathcal{D}_7$, compute the set $A(C)$ of antennas of C in G. (*Comment:* Because of Steps 2 and 4, the tip of each antenna in $A(C)$ appears in a path component of $\mathcal{D}_7$ and is not an endpoint of the path.)

(2) For each chord-free cycle C in $\mathcal{D}_7$, initialize $\Gamma(C) = E(C) \cup A(C)$.

(3) For each chord-sensitive cycle C in $\mathcal{D}_7$, initialize $\Gamma(C) = E(C) \cup A(C) \cup \{e\}$, where e is an arbitrary chord of C in G.

(4) Let $\mathcal{A} = \bigcup_C A(C)$, where C ranges over all cycles of $\mathcal{D}_7$.

(5) For each vertex u that appears in a path component of $\mathcal{D}_7$ and is incident to exactly one edge in $\mathcal{A} \cap (E(G) - B)$, we choose one (arbitrary) edge $e \in B \cap E(\mathcal{D}_7)$ incident to u and add e to $\Gamma(C)$, where C is the cycle in $\mathcal{D}_7$ such that the edge in $\mathcal{A} \cap (E(G) - B)$ is an antenna of C in G. (*Comment:* Since u is the tip of an antenna of some cycle C in $\mathcal{D}_7$, the degree of u in $\mathcal{D}_7$ is 2. So, $B \cap E(\mathcal{D}_7)$ contains at least one edge incident to u. Moreover, since G has no good 3-path related to $\mathcal{D}_7$, no edge is added to more than one $\Gamma(C)$ in this step.)

(6) For each vertex u that appears in a path component of $\mathcal{D}_7$ and is incident to exactly two edges in $\mathcal{A} \cap (E(G) - B)$, we find the two edges e_1 and e_2 of $\mathcal{D}_7$ incident to u, add e_1 to $\mathcal{E}_{C_1}$, and add e_2 to $\mathcal{E}_{C_2}$, where C_1 and C_2 are the cycles in $\mathcal{D}_7$ such that one edge in $\mathcal{A} \cap (E(G) - B)$ is an antenna of C_1 and the other is an antenna of C_2. (*Comment:* Since the degree of u in $G - B$ is 2, B must contain both e_1 and e_2. Moreover, since G has no good 3-path related to $\mathcal{D}_7$, no edge is added to more than one $\Gamma(C)$ in this step or the previous step. Furthermore, it is possible that C_1 and C_2 are the same cycle.)

By the above construction of $\Gamma(C)$ for the cycles C of $\mathcal{D}_7$, it is easy to see that Condition (a) is satisfied. To prove Condition (b), consider an arbitrary chord-sensitive cycle C in $\mathcal{D}_7$. Let $e = \{u, v\}$ be the chord of C in G added to $\Gamma(C)$ in Step (3). Let e' be an arbitrary antenna of C in G. To see that $\Gamma(C)$ satisfies Condition (b), we make the following simple observations:

- If B contains both e and e', then B contains at least one edge of C because C is a cycle and B is a co-path set of G.
- If B contains e' but does not contain e, then B contains at least one edge e_u of C incident to u and contains at least one edge e_v of C incident to v because B is a co-path set of G. Note that $e_u \neq e_v$.
- If B contains e but does not contain e', then at least one edge of B is added to $\Gamma(C)$ in Step (5) or (6) and B also contains at least one edge of C incident to the foot of antenna e'.
- If B contains neither e nor e', then at least one edge of B is added to $\Gamma(C)$ in Step (5) or (6) and B also contains at least one edge e_u of C incident to u and contains at least one edge e_v of C incident to v.

To prove Condition (c), consider an arbitrary chord-free cycle C in $\mathcal{D}_7$ with exactly two antennas in G. Recall that B contains at least one antenna of C in G because of Lemma 3.3 and our choice of B. $\Gamma(C)$ satisfies Condition (c) because of the following observations:

- If B contains both antennas of C in G, then B must contain at least one edge of C because C is a cycle.
- If one antenna e of C in G is in B but the other e' is not, then one edge of B is added to $\Gamma(C)$ in Step (4) or (5) and B contains at least one edge of C incident to the foot of e'.

To prove that Condition (d) holds, we choose three arbitrary antennas e_1, e_2, e_3 of C in G and make the following observations:

- If B contains e_1, e_2, and e_3, then B also contains at least one edge of C.
- If B contains both e_1 and e_2 but does not contain e_3, then at least one edge of B is added to $\Gamma(C)$ in Step (4) or (5) and at least one edge of C incident to the foot of e_3 is in B.
- Suppose that B contains e_1 but contains neither e_2 nor e_3. There are two cases depending on the relative locations of the foots of e_2 and e_3 in C. In case the foots are distinct and adjacent in C, the total number of edges of B added to $\Gamma(C)$ in Step (4) or (5) is at least 2 (because of Step 5) and at least one edge of C is in B. In case the foots are either the same or distinct but not adjacent in C, B contains at least two edges incident to the foots and at least one edge of B is added to $\Gamma(C)$ in Step (4) or (5).
- Suppose that B contains none of e_1, e_2, and e_3. Then, at most two of the foots of e_1, e_2, and e_3 are the same. We consider two cases depending on the relative locations of the foots in C. In case there are two distinct foots adjacent in C, the total number of edges of B added to $\Gamma(C)$ in Step (4) or (5) is at least 2 (because of Step 5) and at least two edges of C are in B. In case no two distinct foots are adjacent in C, B contains at least three of the edges of C incident to the foots and at least one edge of B is added to $\Gamma(C)$ in Step (4) or (5).

□

Theorem 3.10 *There is an $O(n^{1.5})$-time approximation algorithm for MCPS achieving an approximation ratio of $\frac{10}{7}$, where n is the number of vertices in the input graph.*

Proof. Let S be the smaller one between S_1 and S_2. First, consider the case where $b \leq p + |M|$. In this case, we have:

$$\begin{aligned}
p + |M| &\geq b \\
5|B| + 5|E(G)| &\geq 5|S| + 5k + 5c + 5p + 5|E(\mathcal{C})| \quad \text{(Lemma 3.8)} \\
5|B| + 5|E(\mathcal{C})| &\geq 5|E(G)| \quad \text{(Lemma 3.4)} \\
|B| &\geq 4t + 3b + 3c' \quad \text{(Lemma 3.9)} \\
4t + 4b + 4c' + 4\sigma + 8\gamma + 8|M| &\geq 4k \quad \text{(Lemma 3.5)} \\
9|B| + 9k &\geq 9|S| + 9\sigma + 9\gamma + 9|M| \quad \text{(Lemma 3.5)}
\end{aligned}$$

Adding up the left-hand sides and the right-hand sides of the above inequalities respectively, we have $20|B| \geq 14|S| + 5\sigma + \gamma + 5c - c' + 4p$. Since $\sigma \geq 0$, $\gamma \geq 0$, $p \geq 0$, and $c \geq c'$, we have $10|B| \geq 7|S|$.

Next, consider the case where $b \geq p + |M|$. In this case, we have:

$$\begin{aligned}
3b &\geq 3p + 3|M| \\
10|B| + 10|E(G)| + 5|M| &\geq 10|S| + 10k + 10c + 5p + 5b + 10|E(\mathcal{C})| \quad \text{(Lemma 3.8)} \\
10|B| + 10|E(\mathcal{C})| &\geq 10|E(G)| \quad \text{(Lemma 3.4)} \\
2|B| &\geq 8t + 6b + 6c' \quad \text{(Lemma 3.9)} \\
8t + 8b + 8c' + 8\sigma + 16\gamma + 16|M| &\geq 8k \quad \text{(Lemma 3.5)} \\
18|B| + 18k &\geq 18|S| + 18\sigma + 18\gamma + 18|M| \quad \text{(Lemma 3.5)}
\end{aligned}$$

Adding up the left-hand sides and the right-hand sides of the above inequalities respectively, we have $40|B| \geq 28|S| + 10\sigma + 2\gamma + 10c - 2c' + 8p$. Since $\sigma \geq 0$, $p \geq 0$, $\gamma \geq 0$, and $c \geq c'$, we have $10|B| \geq 7|S|$. □

4 Open Problems

An obvious open problem is to ask for better approximation algorithms for MCPS. It is worth noting that unless P=NP, MCPS has no polynomial-time approximation schemes. This is true because of the following theorem:

Theorem 4.1 *MCPS is Max-SNP-hard.*

Proof. The reduction is from the traveling salesman problem with distances one and two ($(1,2)$-TSP for short). Recall that an instance of $(1,2)$-TSP is an edge-weighted complete graph G where each weight is 1 or 2. Given such a graph G, $(1,2)$-TSP requires the computation of a minimum-weight Hamiltonian cycle of G. It is widely known that $(1,2)$-TSP is Max-SNP-hard [6]. Indeed, $(1,2)$-TSP remains to be Max-SNP-hard even if we require that each vertex of the input graph G be incident to at most four edges of weight 1. This result is implicitly shown in [5].

Given an instance G of $(1,2)$-TSP such that each vertex of G is incident to at most four edges of weight 1, we construct an instance H of MCPS from G by removing all edges of weight 2. To show that this reduction is an L-reduction, we prove the following two statements:

1. The minimum size $opt(H)$ of a co-path set of H is at most twice the minimum weight $opt(G)$ of a Hamiltonian cycle of G.

2. Given a co-path set S of H, we can construct a Hamiltonian cycle C of G in polynomial time such that $|opt(G) - w(C)| \leq |opt(H) - |S||$, where $w(C)$ is the total weight of edges in C.

To prove Statement 1, let n (respectively, m) be the number of vertices (respectively, edges) in H. Then, $m \leq 2n$ because the degree of each vertex in H is at most 4. Moreover, $opt(G) =$

$2n - (m - opt(H))$. So, $opt(H) = opt(G) + m - 2n \leq 2n \leq 2 \cdot opt(G)$ because $n \leq opt(G) \leq 2n$ and $m \leq 2n$.

To prove Statement 2, we construct C from S by deleting the edges of S from H and then arbitrarily connecting the resulting paths into a cycle. Obviously, $w(C) = 2n - (m - |S|)$. So, $|S| = w(C) + m - 2n$. Consequently, $w(C) - |S| = opt(G) - opt(H)$, or equivalently $opt(G) - w(C) = opt(H) - |S|$. This completes the proof. □

It would be also interesting to consider more general optimization versions of the radiation hybrid map construction problem than MCPS, where some given clusters of markers can be of size larger than 2. The most general problem is the following: Given a marker set $S = \{1, 2, \ldots, n\}$ and a collection $\mathcal{C} = \{C_1, C_2, \ldots, C_m\}$ of clusters (i.e., subsets of S), we want to remove as few clusters as possible from $\mathcal{C}$ so that there is a linear ordering of the markers in S in which the markers in each leftover cluster C_i appear consecutively. By putting a fixed upper bound k (say, 3) on the sizes of clusters, we can obtain a less general problem. To our knowledge, no approximation algorithms are known for these problems when $k \geq 3$.

We can also consider maximization counterparts of the above problems. Namely, given a marker set $S = \{1, 2, \ldots, n\}$ and a collection $\mathcal{C} = \{C_1, C_2, \ldots, C_m\}$ of clusters (i.e., subsets of S), we want to choose as many clusters as possible from $\mathcal{C}$ so that there is a linear ordering of the markers in S in which the markers in each chosen cluster C_i appear consecutively. By putting a fixed upper bound k on the sizes of clusters, we can obtain a less general problem. Clearly, when $k = 2$, the problem is exactly MPC. Again, to our knowledge, no approximation algorithms are known for these problems when $k \geq 3$.

Acknowledgments

We thank the referees for insightful comments. The first author was supported in part by the Grant-in-Aid for Scientific Research of the Ministry of Education, Science, Sports and Culture of Japan, under Grant No. 20500021. The second was supported in part by NSERC. The third author was fully supported by a grant from City University of Hong Kong (Project No. 7002452).

References

[1] P. Berman and M. Karpinski. 8/7-approximation algorithm for (1,2)-TSP. In *ACM-SIAM Proceedings of the Seventeenth Annual Symposium on Discrete Algorithms (SODA'06)*, pages 641–648, 2006.

[2] H.L. Bodlaender. Planar graphs with bounded treewidth. Technical Report RUU-CS-88-14, Department of Computer Science, University of Utrecht, March 1988.

[3] Y. Cheng, Z. Cai, R. Goebel, G. Lin, and B. Zhu. The radiation hybrid map construction problem: recognition, hardness, and approximation algorithms. *Unpublished Manuscript*, 2008.

[4] D. R. Cox, M. Burmeister, E. R. Price, S. Kim, and R. M. Myers. Radiation hybrid mapping: a somatic cell genetic method for constructing high resolution maps of mammalian chromosomes. *Science*, 250:245–250, 1990.

[5] L. Engebretsen. An Explicit Lower Bound for TSP with Distances One and Two. *Algorithmica*, 35(4): 301–318, 2003.

[6] C. H. Papadimitriou and M. Yannakakis. The traveling salesman problem with distances one and two. *Mathematics of Operations Research*, 18(1):1–11, 1993.

[7] C. W. Richard, D. A. Withers, T. C. Meeker, S. Maurer, G. A. Evans, R. M. Myers, and D. R. Cox. A radiation hybrid map of the proximal long arm of human chromosome 11, containing the multiple endocrine neoplasia type 1 (MEN-l) and bcl-1 disease loci. *American Journal of Human Genetics*, 49:1189–1196, 1991.

[8] D. Slonim, L. Kruglyak, L. Stein, and E. Lander. Building human genome maps with radiation hybrids. *Journal of Computational Biology*, 4:487–504, 1997.

[9] S. Vishwanathan. An approximation algorithm for the asymmetric travelling salesman problem with distances one and two. *Information Processing Letters*, 44:297-302, 1992.