An Approximation Algorithm for Covering Vertices by 4^+-Paths

Mingyang Gong[1], Zhi-Zhong Chen[2](✉), Guohui Lin[1], and Lusheng Wang[3]

[1] University of Alberta, Edmonton, Canada
{mgong4,guohui}@ualberta.ca
[2] Tokyo Denki University, Saitama, Japan
zzchen@mail.dendai.ac.jp
[3] City University of Hong Kong, Hong Kong SAR, China
lusheng.wang@cityu.edu.hk

Abstract. This paper deals with the problem of finding a collection of vertex-disjoint paths in a given graph $G = (V, E)$ such that each path has at least four vertices and the total number of vertices in these paths is maximized. The problem is NP-hard and admits an approximation algorithm which achieves a ratio of 2 and runs in $O(|V|^8)$ time. The known algorithm is based on time-consuming local search, and its authors ask whether one can design a better approximation algorithm by a completely different approach. In this paper, we answer their question in the affirmative by presenting a new approximation algorithm for the problem. Our algorithm achieves a ratio of 1.874 and runs in $O(\min\{|E|^2|V|^2, |V|^5\})$ time. Unlike the previously best algorithm, ours starts with a maximum matching M of G and then tries to transform M into a solution by utilizing a maximum-weight path-cycle cover in a suitably constructed graph.

1 Introduction

Throughout this paper, a graph always means a simple undirected graph without parallel edges or self-loops, and an approximation algorithm always means one running in polynomial time. Let k be a positive integer. Given a graph $G = (V, E)$, MPC_v^{k+} is the problem of finding a collection of vertex-disjoint paths each with at least k vertices in G so that the total number of vertices in these paths is maximized. Note that we can assume that each path in the output collection has at most $2k-1$ vertices. This is because we can split a path having $2k$ or more vertices into two or more paths each having at least k and at most $2k-1$ vertices. MPC_v^{k+} has numerous real-life applications such as transportation networks [9]. In this paper, we mainly focus on MPC_v^{4+}.

On the one hand, MPC_v^{k+} is related to many important optimization problems. For example, Berman and Karpinski [3] consider *the maximum path cover problem*, which is the problem of finding a collection of vertex-disjoint paths in a given graph so that the total number of edges in the paths is maximized. For other related path cover problems with different objectives, the reader is referred

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2024
W. Wu and J. Guo (Eds.): COCOA 2023, LNCS 14461, pp. 459–470, 2024.
https://doi.org/10.1007/978-3-031-49611-0_33

to [1–5,8,15,16] for more details. On the other hand, MPC_v^{k+} can be viewed as a special case of *the maximum-weight* $(2k-1)$*-set packing problem* [10,14] because the former can be easily reduced to the latter as follows. Recall that an instance of the latter problem is a collection $\mathcal{C}$ of sets each having a non-negative weight and at most $2k-1$ elements. The objective is to select a collection of pairwise-disjoint sets in $\mathcal{C}$ so that the total weight of the selected sets is maximized. To reduce MPC_v^{k+} to the maximum-weight $(2k-1)$-set packing problem, it suffices to construct an instance $\mathcal{C}$ of the latter problem from a given instance graph G of MPC_v^{k+}, where $\mathcal{C}$ is the collection of all paths of G with at least k and at most $2k-1$ vertices and the weight of each path P in $\mathcal{C}$ is the number of vertices in P. This reduction leads to an approximation algorithm for MPC_v^{k+} achieving a ratio of k because the maximum-weight $(2k-1)$-set packing problem can be approximated within a ratio of k [10] or within a slightly better ratio of $k - \frac{1}{63,700,992} + \epsilon$ [14] for any $\epsilon > 0$.

MPC_v^{k+} can be solved in polynomial time if $k \le 3$ [5], but is NP-hard otherwise [11]. Kobayashi *et al.* [11] design an approximation algorithm for MPC_v^{4+} achieving a ratio of 4. Afterwards, Gong et al. [9] give the formal definition of MPC_v^{k+} and present an approximation algorithm for MPC_v^{k+} which achieves a ratio of $\rho(k) \le 0.4394k + 0.6576$ and runs in $O(|V|^{k+1})$ time. The core of their algorithm is three local improvement operations, each of which increases the number of vertices in the current solution by at least 1 if it is applicable. The algorithm stops when none of the three operations is applicable. They employ an amortization scheme to analyze the approximation ratio of their algorithm by assigning the vertices in the optimal solution to the vertices of the solution outputted by their algorithm. For the special case where $k = 4$, they design two more local improvement operations to increase the number of vertices or the number of paths with exactly 4 vertices in the current solution, and then use a more careful amortization scheme to prove that the approximation ratio of their algorithm is bounded by 2 although the running time jumps to $O(|V|^8)$. As an open question, they ask whether one can design better approximation algorithms for the problem by completely different approaches.

1.1 Our Contribution and Design Highlights

In this paper, we answer the open question in the affirmative for the case where $k = 4$. Motivated by the approaches in [5,6,12] for similar problems, one may want to design an approximation algorithm for MPC_v^{k+} by first computing a maximum path-cycle cover $\mathcal{C}$ of the input graph G and then transforming $\mathcal{C}$ into a solution for G. Unfortunately, this approach to maximizing the number of edges does not seem to work. Our new idea for designing a better approximation algorithm for MPC_v^{4+} is to let the algorithm start by computing a maximum matching M in the input graph G. The intuition behind this idea is that the paths in an optimal solution for G can cover at most $\frac{5}{2}|M|$ vertices. So, it suffices to find a solution for G of which the paths cover a large fraction of the endpoints of the edges in M. To this purpose, our algorithm then constructs a maximum-weight path-cycle cover C in an auxiliary graph suitably constructed from M

and G. Our algorithm further tries to use the edges in C to connect a large fraction of the edges of M into paths with at least four vertices. If the algorithm fails to do so, then it will be able to reduce the problem to a smaller problem and in turn uses recursion to get a good solution.

Due to lack of space, the proofs of most lemmas are omitted here and will be shown in the journal version.

2 Basic Definitions

Throughout the remainder of this paper, we fix an instance G of MPC_v^{4+} for discussion. Let $n = |V(G)|$ and $m = |E(G)|$. For the graph G, let $V(G)$ and $E(G)$ be the vertex and edge set of G.

For a subset F of $E(G)$, we use $V(F)$ to denote the set $\{v \in V(G) \mid v$ is an endpoint of an edge in $F\}$. A *spanning subgraph* of G is a subgraph H of G with $V(H) = V(G)$. For a set F of edges in G, $G - F$ denotes the spanning subgraph $(V(G), E(G) \setminus F)$. In contrast, for a set F of edges with $V(F) \subseteq V(G)$ and $F \cap E(G) = \emptyset$, $G + F$ denotes the graph $(V(G), E(G) \cup F)$. The *degree* of a vertex v in G, denoted by $d_G(v)$, is the number of edges incident to v in G. A vertex v of G is *isolated* in G if $d_G(v) = 0$. The *subgraph induced by* a subset U of $V(G)$, denoted by $G[U]$, is the graph (U, E_U), where $E_U = \{\{u, v\} \in E(G) \mid u, v \in U\}$. Two vertex-disjoint subgraphs of G are *adjacent* in G if G has an edge between them.

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 of G or a connected subgraph of G in which exactly two vertices (called the *endpoints*) are of degree 1 and the others (called the *internal vertices*) are of degree 2. A *path component* of G is a connected component of G that is a path. If a path component is an edge, then it is called an *edge component*. The *order* of a cycle or path C, denoted by $|C|$, is the number of vertices in C. A *triangle* of G is a cycle of order 3 in G. A *k-path* of G is a path of order k in G, while a *k^+-path* of G is a path of order k or more in G. A *matching* of G is a (possibly empty) set of edges of G in which no two edges share an endpoint. A *maximum matching* of G is a matching of G whose size is maximized over all matchings of G. A *path-cycle cover* of G is a set F of edges in G such that in the spanning subgraph $(V(G), F)$, the degree of each vertex is at most 2. A *star* is a connected graph in which exactly one vertex is of degree ≥ 2 and each of the remaining vertices is of degree 1. The vertex of degree ≥ 2 is called the *center*, while the other vertices are the *satellites* of the star.

Notation 1. *For a graph G,*

- *$OPT(G)$ denotes an optimal solution for the instance graph G of MPC_v^{4+}, and $opt(G)$ denotes the total number of vertices in $OPT(G)$;*
- *$ALG(G)$ denotes the solution for G outputted by a specific algorithm, and $alg(G)$ denotes the total number of vertices in $ALG(G)$.*

3 The Algorithm for MPC_v^{4+}

Our algorithm for MPC_v^{4+} consists of multiple phases. In the first phase, it computes a maximum matching M in G in $O(\sqrt{n}m)$ time [13], initializes a subgraph $H = (V(M), M)$, and then repeatedly modifies H and M as described in Sect. 3.1. With a small loss of vertices, OPT can be transferred to a matching (by moving edges) and thus we have the following lemma.

Lemma 1. *$|V(M)| \geq \frac{4}{5} opt(G)$.*

3.1 Modifying H and M

We here describe a process for modifying H and M iteratively. The process consists of two steps. During these two steps, the following will be an invariant, which will be proved in Lemma 2.

Invariant 1. *M is both a maximum matching of G and a subset of $E(H)$. Each connected component K of H is an edge, a triangle, a star, or a 5-path. Moreover, if K is a 5-path, then the two edges of $E(K)$ incident to the endpoints of K are in M; otherwise, exactly one edge of K is in M.*

Initially, Invariant 1 clearly holds. Since M is a maximum matching, any two vertices of $V(G) \setminus V(H)$ cannot be adjacent to each other. Moreover, for any vertex $u_0 \in V(G) \setminus V(H)$, either it is incident to two different edge components e_0, e_1; incident to an unique edge component e_0 or not incident to any edge components of H. Generally speaking, for the first case, we present an operation to generate a 5-path by connecting u_0 with e_0, e_1. For the second case, u_0 and e_0 form a triangle or a star with other vertices of $V(G) \setminus V(H)$. Lastly, if no vertex of $V(G) \setminus V(H)$ is incident to e_0, then e_0 remains an edge component of H.

Definition 1. *An* augmenting triple *with respect to H is a triple $(u_0, e_0 = \{v_0, w_0\}, e_1 = \{v_1, w_1\})$ such that $u_0 \in V(G) \setminus V(H)$, both e_0 and e_1 are edge components of H. We modify H and M as follows:*

C1. If $\{u_0, v_0\}, \{u_0, v_1\} \in E(G)$, then add u_0 and the edges $\{u_0, v_0\}$, $\{u_0, v_1\}$ to H.
C2. If $\{u_0, v_0\}, \{w_0, v_1\} \in E(G)$, then add u_0 and the edges $\{u_0, v_0\}$, $\{w_0, v_1\}$ to H and then modify M by replacing e_0 with $\{u_0, v_0\}$.

Clearly, once the above modification is executed, a 5-path is generated. We next give the first two steps for H and M as follows.

Step 1.1 Repeatedly modify H and M with an augmenting triple until it is not applicable;
Step 1.2 Add all those edges $\{u, v\} \in E(G)$ such that $u \in V(G) \setminus V(H)$ and v is an endpoint of an edge component of H, as well as their endpoints u, to H.

We have the next lemma on H and M at the end of Step 1.2.

Lemma 2. *The above Steps 1.1–1.2 can be done in $O(\min\{m^2 n, n^4\})$ time such that Invariant 1 always holds.*

3.2 Bad Components and Rescuing Them

We consider the subgraph H and the maximum matching M at the end of Step 1.2. In the sequel, a component always means a *connected* component. Note that a 5-path of H can be contained in the solution, but we can not form a 4^+-path from any other components of H, which are defined as *bad.*

Definition 2. *A* bad component *of H is a connected component that is not a 5-path. By Invariant 1, a bad component is an edge, a triangle or a star.*

Clearly, moving all bad components of H will lead to a large loss of the vertices in $V(M)$. So, in this subsection, we construct a maximum-weighted path-cycle cover to connect a bad component of H to another bad component or a 5-path as many as possible such that we are able to form more 4^+-paths from bad components. We call it a *rescue* process of bad components.

> **Step 2.1** Construct a spanning subgraph G_1 of G of which the edge set consists of all the edges $\{v_1, v_2\}$ of G such that v_1 and v_2 appear in different components of H and at least one of the components is bad.

Definition 3. *A set F of edges in G_1* saturates a bad component *K of H if at least one edge in F is incident to a vertex of K. The* weight *of F is the number of bad components saturated by F.*

Lemma 3. *A maximum-weighted path-cycle cover in G_1 can be found in $O(mn \log n)$ time.*

> **Step 2.2.** Compute a maximum-weighted path-cycle cover C of G_1 (as in the proof of Lemma 3).
> **Step 2.3.** As long as C contains an edge e such that $C \setminus e$ has the same weight as C, repeatedly remove e from C, that is, C is updated to $C \setminus e$.

Notation 2. – *C denotes the maximum-weighted path-cycle cover of G_1 computed at the end of Step 2.3.*
– *M_C denotes the subset of the maximum matching M containing those edges in 5-paths of H or in bad components of H saturated by C.*

Intuitively, the maximum-weighted path-cycle cover C connects as many bad components as possible with each other or 5-paths. So, the vertices of M_C is relatively larger than $opt(G)$ since it only removes the vertices (exactly two vertices in $V(M)$ by Invariant 1) of each bad component not saturated by C. The following lemma shows this fact.

Lemma 4. $|V(M_C)| \geq \frac{4}{5} opt(G)$.

3.3 Structure of Composite Components of $H + C$

By Lemma 4, in order to obtain a good approximate solution for G, it suffices to focus on M_C instead of its superset M. By Step 2.3, we remove the edges of C such that the weight of C is unchanged. So, it gives us simpler structures of each connected component of $H + C$.

Notation 3. – *$H+C$ denotes the spanning subgraph $(V(G), E(H) \cup C)$. In the sequel, we use K to refer to a component in $H + C$.*
– *$(H+C)_m$ denotes the graph obtained from $H+C$ by contracting each component of H into a single node. In other words, the nodes of $(H+C)_m$ one-to-one correspond to the components of H and two nodes are adjacent in $(H + C)_m$ if and only if C contains an edge between the two corresponding components. We use $(K)_m$ to refer to the component of $(H + C)_m$ corresponding to the component K in $H + C$.*

We next show the structures of each component $(K)_m$.

Lemma 5. *For each component $(K)_m$ of $(H + C)_m$ (see Notation 3), the following statements hold:*

1. *$(K)_m$ is an isolated node, an edge, or a star.*
2. *If $(K)_m$ is an edge, then at least one endpoint of $(K)_m$ corresponds to a bad component of H.*
3. *If $(K)_m$ is a star, then each satellite of $(K)_m$ corresponds to a bad component of H.*

If $(K)_m$ is isolated, then K is defined as *isolated* as well. Otherwise, K is a *composite component* and it contains two or more components of H, which are connected through the edges of C. If $(K)_m$ is isolated, K is a 5-path or a bad component of H, not saturated by C. Recall that we only focus on the vertices of M_C. So, we can assume $(K)_m$ is a 5-path if $(K)_m$ is isolated. We next discuss the case that $(K)_m$ is an edge or a star. By the second statement in Lemma 5, when $(K)_m$ is an edge, we choose an endpoint corresponding to a bad component of H as the satellite, while the other endpoint as the center.

Definition 4. *For each composite component K of $H + C$, its* center element *is the component of H corresponding to the center of $(K)_m$, and it is denoted as K_c in the sequel; the other components of H contained in K are the* satellite elements *of K.*

Every vertex v of K_c is defined as an anchor*. The edge connecting v to a satellite element S in C is called the* rescue-edge *for S and v is called the* supporting *anchor for S. For a nonnegative integer j, an anchor v is a j-*anchor *if v is the supporting anchor for exactly j satellite elements of $H + C$.*

Since C is a path-cycle cover of G_1, each anchor is a 0-, 1-, or 2-anchor. When $(K)_m$ is isolated, then K is a 5-path and thus $opt(K) = 5$. One might ask whether the solution $OPT(K)$ can be easily computed for a composite component K. The following lemma answers this question in the affirmative.

Lemma 6. *For each component K of $H + C$, an $OPT(K)$ can be computed in $O(1)$ time.*

Definition 5. *For each composite component K of $H + C$, let $s(K) = |V(K) \cap V(M_C)|$. A* critical component *of $H + C$ is a component K with $\frac{s(K)}{opt(K)} \geq \frac{14}{11}$.*

Generally speaking, by computing an $OPT(K)$ for every K of $H + C$ and outputting their union as an approximate solution for G, we obtain an approximation algorithm for MPC_v^{4+} achieving a ratio of $\frac{5}{4} \max_K \frac{s(K)}{opt(K)}$ because of Lemma 4, unless K is *critical* (Definition 4) and *responsible* (Definition 8). If K is an isolated 5-path, then by Invariant 1, we have $\frac{s(K)}{opt(K)} = \frac{4}{5}$. But if K is a composite component, $\frac{s(K)}{opt(K)}$ is not necessarily small (smaller than our target value which is about 1.4992). Next, we show the possible structures of a critical component in Fact 1.

Fact 1. *A critical component K of $H + C$ has one 2-anchor or two 2-anchors. Moreover, K_c is an edge or a 5-path. If K_c is an edge, then $s(K) = 8$; if K_c is a 5-path, then $s(K) \in \{8, 10, 14, 16, 18\}$. Figure 1 shows all possible structures of a critical component.*

Proof. We can prove the fact by discussing the number of 2-anchors in K_c.

Remark 1. Even if a satellite element S of K can be a star or triangle, we almost always draw only one edge of S in Fig. 1 for simplicity.

Definition 6. *A 2-anchor of $H + C$ is* critical *if it appears in a critical component of $H + C$. A satellite element of $H + C$ is* critical *if its rescue-anchor is critical in $H + C$.*

By Fact 1, every critical component must have one or two critical 2-anchor.

Definition 7. *Suppose that v is a 0- or 1-anchor in $H + C$ and S is a satellite element in $H + C$ such that S has a vertex w with $\{v, w\} \in E(G)$. Then,* moving *S to v in $H + C$ is the operation of modifying C by replacing the rescue-edge of S with the edge $\{v, w\}$.*

Suppose a critical component K has exactly one 2-anchor. Then if we move one of critical satellite element to an isolated 5-path (if possible), K and the 5-path will be both not critical since they do not have 2-anchor. So, such moving decreases the number of critical components by one. However, we cannot guarantee that every moving will reduce critical components. In Fact 2, we discuss the different movings and their effects.

Fact 2. *For each critical component K of $H+C$ and its critical satellite element S, the following statements hold:*

1. *If we move S to another component (not K), then K is no longer critical and will not become isolated.*

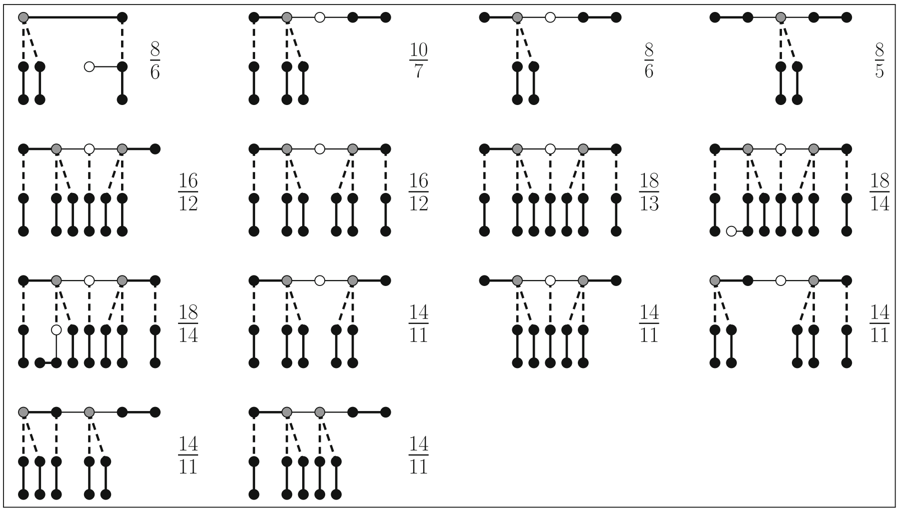

Fig. 1. The possible structures for a critical component K of $H + C$, where thick (respectively, dashed) edges are in the matching M (respectively, the path-cycle cover C), thin edges are not in $M \cup C$, the filled (respectively, blank) vertices are in (respectively, not in) $V(M)$, gray vertices are 2-anchors, and the fraction on the right side of each structure is $\frac{s(K)}{opt(K)}$.

2. *If v is a 0-anchor of K such that v is adjacent to S in G, then moving S to v in $H + C$ makes K no longer critical.*
3. *If v is a 1-anchor in K, then moving S to v makes K remain critical only if K has the first structure of Fig. 1, or K has the last or the second last structure in the bottom row of Fig. 1 and the rescue-anchor of S is the leftmost 2-anchor in K.*

Roughly speaking, we might need to pay more attention to 1-anchors since if a critical satellite element is moved to a 1-anchor v, then v is possible to become a critical 2-anchor. So, we introduce the following definition of 1-anchors.

Definition 8. *Let K be a composite component of $H + C$. If K has a 1-anchor v such that G has an edge between v and some critical satellite-element S of $H + C$ in G and moving S to v in $H + C$ makes K critical in $H + C$, then we call K a* responsible component *of $H + C$ and call v a* responsible 1-anchor *of $H + C$.*

By the third statement in Fact 2, a component of $H + C$ can be both critical and responsible only if it has the first or one of the last two structures in Fig. 1.

Lemma 7. *Suppose that a component K of $H + C$ is both critical and responsible. If K has the first structure in Fig. 1, then $s(K) = 8$ and we find a feasible solution with at least 7 vertices in $O(1)$ time; otherwise, $s(K) = 14$ and we find a feasible solution with at least 12 vertices in $O(1)$ time.*

By the above lemma, we know for each critical and responsible component K, we can find a feasible solution for K in constant time, which is still denoted as $OPT(K)$ for ease of presentation, with $\frac{s(K)}{opt(K)} < \frac{14}{11}$. Now, we can regard each critical and responsible component K as a non-critical component. So, any critical component cannot be responsible or vice versa. Hereafter, a critical component always refers to a critical but not responsible component and a responsible component always refers to a responsible but not critical component.

By Definition 8, the structure for a responsible component of $H+C$ can only be obtained by deleting a critical satellite-element from one of the structures in Fig. 1. So, by Fig. 1, we can easily list all possible structures for responsible components of $H+C$.

3.4 Operations for Modifying Critical Components

In this subsection, we define three operations for modifying C (and accordingly one or more critical components of $H+C$) so that after the modification, $H+C$ will hopefully have fewer critical components. Let v be a vertex of a satellite element S in a critical component K and v' be a vertex of K' in $H+C$. We remark that K and K' may be the same. Suppose $\{v, v'\} \in E(G) \setminus C$ and we design the following three operations.

Operation 1. *Suppose v' is a 0-anchor of K or v' is a non-responsible 1-anchor. Then, the operation modifies C by replacing the rescue-edges of S with $\{v, v'\}$.*

Clearly, Operation 1 does not change the weight of C by the first statement in Fact 2. Suppose v' is a 0-anchor. If $K = K'$, then after Operation 1, K is no longer critical by the second statement of Fact 2. Then, we suppose $K \neq K'$. Obviously, K is no longer critical but K' may become critical after Operation 1. So, Operation 1 may not necessarily decrease but does not increase the number of critical components in $H+C$. Fortunately, Operation 1 changes v' from a 0-anchor to a 1-anchor. Similarly, it is not hard to check if v' is a non-responsible 1-anchor, then Operation 1 makes K, K' both not critical. So, Operation 1 decreases the number of 0-anchors in $H+C$ by 1 or the number of critical components in $H+C$ by 1. Obviously, Operation 1 does not change the number of components in $H+C$.

Operation 2. *Suppose v' is in a satellite-element S' of K' and the center element K'_c of K' is an edge or a star to which no satellite element other than S' is adjacent in $H+C$. Then, the operation modifies C by replacing the rescue-edge of S with $\{v, v'\}$.*

Obviously, Operation 2 does not change the weight of C by the first statement of Fact 2. Note that K' has no 2-anchor and hence K' is not critical by Fact 1. So, $K \neq K'$ since K is critical. Moreover, after Operation 2, S' becomes the center element of K' and hence Lemma 5 still holds. Furthermore, by the first statement in Fact 2 and Fact 1, K, K' are not critical after Operation 2 and thus Operation 2 decreases the number of critical components in $H + C$ by 1. Clearly, Operation 2 does not change the number of components in $H + C$. Before Operation 2, K' may have one 0-anchor x. After Operation 2, x will be in a satellite element of $H + C$ and hence will not be a 0-anchor, but S' will become a center element with two satellite elements adjacent to it in $H + C$, implying that one vertex of S' may become a 0-anchor in $H + C$ (or not an anchor, if S' is a star). In summary, Operation 2 does not increase the number of 0-anchors in $H + C$.

Operation 3. *Suppose v' appears in a satellite-element S' of K' and K'_c is a 5-path , or K'_c is an edge or a star to which at least one more satellite element other than S' is adjacent in $H + C$. Then, the operation modifies C by replacing the rescue-edges of S and S' with the edge $\{v, v'\}$.*

By the first statement of Fact 2, Operation 3 does not change the weight of C since K, K' will not be an isolated bad component of H. Operation 3 uses the edge $\{v, v'\}$ to connect S and S' into a new composite component K_{new} of $H + C$. By Fact 1, K_{new} is not critical. If $K = K'$, then clearly Operation 3 does not increase the number of critical components in $H + C$. Otherwise, Operation 3 makes K not critical because of the first statement in Fact 2, but it is possible that Operation 3 makes K' critical. In any case, Operation 3 does not increase the number of critical components in $H + C$. Luckily, Operation 3 always increases the number of components in $H + C$ by 1.

Lemma 8. *Operations 1–3 can be repeatedly performed at most $O(n^2)$ times.*

Suppose that G has an edge $\{v, v'\}$ such that v is in a critical satellite-element S of $H + C$ and $v' \notin V(S)$ when none of Operations 1–3 is applicable. Then v' is a 2-anchor or a responsible 1-anchor.

3.5 Bounding *opt*(G)

Let R denote the set of vertices $v \in V(H)$ such that v is a 2-anchor or a responsible 1-anchor in $H + C$. By Lemma 8, once none of Operations 1–3 is applicable, any critical satellite element can be only incident to the vertices of R. Clearly, $|R \cap V(K)|$ is bounded by the total number of 1- and 2-anchors in K_c (K_c is not a triangle). Thus, if K_c is an edge, then $|R \cap V(K)| \in \{0, 1, 2\}$; if K_c is a star, then $|R \cap V(K)| \in \{0, 1\}$; if K_c is a 5-path, then $|R \cap V(K)| \in \{0, 1, 2, 3, 4, 5\}$. By Fact 1 and Lemma 7, each critical component K has one or two 2-anchors and no responsible 1-anchor. That is, $|R \cap V(K)| \in \{1, 2\}$.

Notation 4. *For the components in $H + C$, we define the notations as follows.*

- *Let $\mathcal{K}$ be the set of composite components or isolated 5-paths of $H + C$.*
- *For each $i \in \{0, 1, 2, 3, 4, 5\}$, let $\mathcal{K}_i \subseteq \mathcal{K}$ be a subset of $\mathcal{K}$ such that $|R \cap V(K)| = i$.*
- *For each $i \in \{1, 2\}$, let $\mathcal{K}_{i,c}$ be the set of critical components in $\mathcal{K}_i$.*
- *Let R_c be the set of 2-anchors in the critical components of $H + C$.*

– *$U_c = \bigcup_{v \in R_c} \{w \in V(H) \mid w$ is in a critical satellite-element whose rescue-anchor is $v\}$.*
– *Let $G_c = G[V(G) \setminus (R_c \cup U_c)]$.*

Lemma 9. *$opt(G) \le opt(G_c) + 7\sum_{i=1}^{5} i|\mathcal{K}_i|$.*

The above lemma indicates that though there are many critical components in $H+C$, after we "destroyed" all the critical components, the problem is reduced to a smaller problem on G_c and $opt(G_c)$ is not far away from $opt(G)$. So, it is possible to get a good solution of G by recursively solving the problem on G_c.

4 Summary of the Algorithm

Let $r = \frac{15+\sqrt{505}}{20} \approx 1.874$ be the positive root to the quadratic equation $10r^2 - 15r - 7 = 0$. Our algorithm proceeds as follows.

0. If $|V(G)| \le 4$, find an optimal solution by brute-force search, output it, and then halt.
1. Construct the graph H as follows:
   (a) Compute a maximum matching M in G and initialize $H = (V(M), M)$.
   (b) Modify M and H by performing Steps 1.1 and 1.2 in Sect. 3.1.
2. Compute a maximum path-cycle cover C and modify it as follows:
   (a) Perform Steps 2.1, 2.2, and 2.3 in Sect. 3.2 to compute a maximum path-cycle cover C in an auxiliary graph G_1.
3. Repeatedly perform Operations 1, 2, and 3 in Sect. 3.4 to modify C, until none of them is applicable.
4. If no component of $H+C$ is critical, or $\frac{\sum_{i=1}^{5} i|\mathcal{K}_i|}{|\mathcal{K}_{1,c}|+2|\mathcal{K}_{2,c}|} > \frac{5}{7}r$, then
   (a) compute $OPT(K)$ for each component K of $H+C$ that is not an isolated bad component of H by Lemma 6;
   (b) output their union as a solution for G, and then halt.
5. Otherwise, there is at least one critical component and $\frac{\sum_{i=1}^{5} i|\mathcal{K}_i|}{|\mathcal{K}_{1,c}|+2|\mathcal{K}_{2,c}|} \le \frac{5}{7}r$.
   (a) Recursively call the algorithm on the graph G_c to obtain a solution $ALG(G_c)$.
   (b) For each $v \in R_c$, compute a 5^+-path P_v since v is a 2-anchor.
   (c) Output the union of $ALG(G_c)$ and $\cup_{v \in R_c} P_v$, and halt.

Theorem 1. *The running time of the algorithm is bounded by $O(\min\{m^2n^2, n^5\})$ and the approximation ratio is at most $r = \frac{15+\sqrt{505}}{20} < 1.874$.*

Due to the page limitation, the proof of Theorem 1 is omitted here.

References

1. Asdre, K., Nikolopoulos, S.D.: A linear-time algorithm for the k-fixed-endpoint path cover problem on cographs. Networks **50**, 231–240 (2007)
2. Asdre, K., Nikolopoulos, S.D.: A polynomial solution to the k-fixed-endpoint path cover problem on proper interval graphs. Theoret. Comput. Sci. **411**, 967–975 (2010)
3. Berman, P., Karpinski, M.: 8/7-approximation algorithm for (1,2)-TSP. In: Proceedings of ACM-SIAM SODA 2006, pp. 641–648 (2006)
4. Cai, Y., et al.: Approximation algorithms for two-machine flow-shop scheduling with a conflict graph. In: Wang, L., Zhu, D. (eds.) COCOON 2018. LNCS, vol. 10976, pp. 205–217. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-94776-1_18
5. Chen, Y., et al.: Path cover with minimum nontrivial paths and its application in two-machine flow-shop scheduling with a conflict graph. J. Comb. Optim. **43**, 571–588 (2022)
6. Chen, Z.-Z., Konno, S., Matsushita, Y.: Approximating maximum edge 2-coloring in simple graphs. Discret. Appl. Math. **158**, 1894–1901 (2010)
7. Gabow, H.N.: An efficient reduction technique for degree-constrained subgraph and bidirected network flow problems. In: Proceedings of ACM STOC 1983, pp. 448–456 (1983)
8. Gomez, R., Wakabayashi, Y.: Nontrivial path covers of graphs: existence, minimization and maximization. J. Comb. Optim. **39**, 437–456 (2020)
9. Gong, M., Fan, J., Lin, G., Miyano, E.: Approximation algorithms for covering vertices by long paths. In: Proceedings of MFCS 2022. LIPIcs, vol. 241, pp. 53:1–53:14 (2022)
10. Hochbaum, D.S.: Efficient bounds for the stable set, vertex cover and set packing problems. Discret. Appl. Math. **6**, 243–254 (1983)
11. Kobayashi, K., et al.: Path cover problems with length cost. In: Mutzel, P., Rahman, M.S., Slamin (eds.) WALCOM 2022. LNCS, vol. 13174, pp. 396–408. Springer, Cham (2022). https://doi.org/10.1007/978-3-030-96731-4_32
12. Kosowski, A.: Approximating the maximum 2- and 3-edge-colorable subgraph problems. Discret. Appl. Math. **157**, 3593–3600 (2009)
13. Micali, S., Vazirani, V.V.: An $O(\sqrt{|V|}|E|)$ algorithm for finding maximum matching in general graphs. In: Proceedings of IEEE FOCS 1980, pp. 17–27 (1980)
14. Neuwohner, M.: An improved approximation algorithm for the maximum weight independent set problem in d-claw free graphs. In: Proceedings of STACS 2021, pp. 53:1–53:20 (2021)
15. Pao, L.L., Hong, C.H.: The two-equal-disjoint path cover problem of matching composition network. Inf. Process. Lett. **107**, 18–23 (2008)
16. Rizzi, R., Tomescu, A.I., Mäkinen, V.: On the complexity of minimum path cover with subpath constraints for multi-assembly. BMC Bioinform. **15**, S5 (2014)