A Local Search 4/3-approximation Algorithm for the Minimum 3-path Partition Problem

Yong Chen[1], Randy Goebel[2], Guohui Lin[2(✉)], Longcheng Liu[2,3], Bing Su[4], Weitian Tong[5], Yao Xu[2(✉)], and An Zhang[1]

[1] Department of Mathematics, Hangzhou Dianzi University, Hangzhou, China
{chenyong,anzhang}@hdu.edu.cn
[2] Department of Computing Science, University of Alberta, Edmonton, Canada
{rgoebel,guohui,xu2}@ualberta.ca
[3] School of Mathematical Sciences, Xiamen University, Xiamen, China
longchengliu@xmu.edu.cn
[4] School of Economics and Management, Xi'an Technological University, Xi'an, China
subing684@sohu.com
[5] Department of Computer Science, Georgia Southern University, Statesboro, USA
wtong@georgiasouthern.edu

Abstract. Given a graph $G = (V, E)$, the 3-path partition problem is to find a minimum collection of vertex-disjoint paths each of order at most 3 to cover all the vertices of V. It is different from but closely related to the well-known 3-set cover problem. The best known approximation algorithm for the 3-path partition problem was proposed recently and has a ratio 13/9. Here we present a local search algorithm and show, by an amortized analysis, that it is a 4/3-approximation. This ratio matches up to the best approximation ratio for the 3-set cover problem.

Keywords: k-path partition · Path cover · k-set cover · Approximation algorithms · Local search · Amortized analysis

1 Introduction

Motivated by the data integrity of communication in wireless sensor networks and several other applications, the k-PATH PARTITION (kPP) problem was first considered by Yan et al. [14]. Given a simple graph $G = (V, E)$ (we consider only simple graphs), with $n = |V|$ and $m = |E|$, the *order* of a simple path in G is the number of vertices on the path and it is called a *k-path* if its order is k. The kPP problem is to find a minimum collection of vertex-disjoint paths each of order at most k such that every vertex is on some path in the collection.

Clearly, the 2PP problem is exactly the MAXIMUM MATCHING problem, which is solvable in $O(m\sqrt{n}\log(n^2/m)/\log n)$-time [7]. For each $k \geq 3$, kPP is NP-hard [6]. We point out the key phrase "at most k" in the definition, that

Y. Chen et al. (Eds.): FAW 2019, LNCS 11458, pp. 14–25, 2019.
https://doi.org/10.1007/978-3-030-18126-0_2

ensures the existence of a feasible solution for any given graph; on the other hand, if one asks for a path partition in which every path has an order exactly k, the problem is called *P_k-partitioning* and is also NP-complete for any fixed constant $k \geq 3$ [6], even on bipartite graphs of maximum degree three [11]. To the best of our knowledge, there is no approximation algorithm with proven performance for the general kPP problem, except the trivial k-approximation using all 1-paths. For 3PP, Monnot and Toulouse [11] proposed a 3/2-approximation, based on two maximum matchings; recently, Chen et al. [2] presented an improved 13/9-approximation.

The kPP problem is a generalization to the Path Cover problem [5] (also called Path Partition), which is to find a minimum collection of vertex-disjoint paths which together cover all the vertices in G. Path Cover contains the Hamiltonian Path problem [6] as a special case, and thus it is NP-hard and it is outside APX unless P = NP.

The kPP problem is also closely related to the well-known Set Cover problem. Given a collection of subsets $\mathcal{C} = \{S_1, S_2, \ldots, S_m\}$ of a finite ground set $U = \{x_1, x_2, \ldots, x_n\}$, an element $x_i \in S_j$ is said to be *covered* by the subset S_j, and a *set cover* is a collection of subsets which together cover all the elements of the ground set U. The Set Cover problem asks to find a minimum set cover. Set Cover is one of the first problems proven to be NP-hard [6], and is also one of the most studied optimization problems for the approximability [8] and inapproximability [4,12,13]. The variant of Set Cover in which every given subset has size at most k is called k-Set Cover, which is APX-complete and admits a 4/3-approximation for $k = 3$ [3] and an $(H_k - \frac{196}{390})$-approximation for $k \geq 4$ [10].

To see the connection between kPP and k-Set Cover, we may take the vertex set V of the given graph as the ground set, and an ℓ-path with $\ell \leq k$ as a subset; then the kPP problem is the same as asking for a minimum *exact* set cover. That is, the kPP problem is a special case of the *minimum* Exact Cover problem [9], for which unfortunately there is no approximation result that we may borrow. Existing approximations for (non-exact) k-Set Cover do not readily apply to kPP, because in a feasible set cover, an element of the ground set could be covered by multiple subsets. There is a way to enforce the *exactness* requirement in the Set Cover problem, by expanding $\mathcal{C}$ to include all the proper subsets of each given subset $S_j \in \mathcal{C}$. But in an instance graph of kPP, not every subset of vertices on a path is traceable, and so such an expanding technique does not apply. In summary, kPP and k-Set Cover share some similarities, but none contains the other as a special case.

In this paper, we study the 3PP problem. The authors of the 13/9-approximation [2] first presented an $O(nm)$-time algorithm to compute a k-path partition with the least 1-paths, for any $k \geq 3$; then they applied an $O(n^3)$-time greedy approach to merge three 2-paths into two 3-paths whenever possible. We aim to design better approximations for 3PP with provable performance, and we achieve a 4/3-approximation. Our algorithm starts with a 3-path partition with the least 1-paths, then it applies a local search scheme to repeatedly search

for an *expected collection* of 2- and 3-paths and replace it by a strictly smaller *replacement collection* of new 2- and 3-paths.

The rest of the paper is organized as follows. In Sect. 2 we present the local search scheme searching for all the expected collections of 2- and 3-paths. The performance of the algorithm is proved through an amortized analysis in Sect. 3. We conclude the paper in Sect. 4.

2 A Local Search Approximation Algorithm

The 13/9-approximation proposed by Chen et al. [2] applies only one replacement operation which is to merge three 2-paths into two 3-paths. In order to design approximation for 3PP with better performance, we examine four more replacement operations each transfers three 2-paths to two 3-paths with the aid of a few other 2- or 3-paths. Starting with a 3-path partition with the least 1-paths, our approximation algorithm repeatedly finds a certain expected collection of 2- and 3-paths and replaces it by a replacement collection of one less new 2- and 3-paths, in which the net gain is exactly one.

In Sect. 2.1 we present all the replacement operations to perform on the 3-path partition with the least 1-paths. The complete algorithm, denoted as Approx, is summarized in Sect. 2.2.

2.1 Local Operations and Their Priorities

Throughout the local search, the 3-path partitions are maintained to have the least 1-paths. Our four local operations are designed so not to touch the 1-paths, ensuring that the final 3-path partition still contains the least 1-paths. These operations are associated with different priorities, that is, one operation applies only when all the other operations of higher priorities (labeled by smaller numbers) fail to apply to the current 3-path partition. We remind the reader that the local search algorithm is iterative, and every iteration ends after executing a designed local operation. The algorithm terminates when none of the designed local operations applies.

Definition 1. *With respect to the current* 3*-path partition* $\mathcal{Q}$*, a local* Operation i_1*-*i_2*-By-*j_1*-*j_2*, where* $j_1 = i_1 - 3$ *and* $j_2 = i_2 + 2$*, replaces an* expected collection *of* i_1 2*-paths and* i_2 3*-paths of* $\mathcal{Q}$ *by a* replacement collection *of* j_1 2*-paths and* j_2 3*-paths on the same subset of* $2i_1 + 3i_2$ *vertices.*

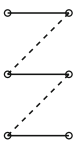
Operation 3-0-By-0-2**, highest priority 1:** When three 2-paths of $\mathcal{Q}$ can be connected into a 6-path in the graph G (see Fig. 1a for an illustration), they form into an expected collection. By removing the middle edge on the 6-path, we achieve two 3-paths on the same six vertices and they form the replacement collection. This is the only local operation executed in the 13/9-approximation [2].

In each of the following operations, we need the aid of one or two 3-paths to transfer three 2-paths to two 3-paths. We first note that for a 3-path u-w-$v \in \mathcal{Q}$,

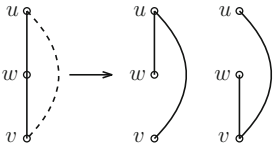
if $(u, v) \in E$ too, then if desired, we may *rotate* u-w-v into another 3-path with w being an endpoint (see Fig. 1b for an illustration). In the following, any 3-path in an expected collection can be either the exact one in $\mathcal{Q}$ or the one rotated from a 3-path in $\mathcal{Q}$.

Operation 3-1-By-0-3**, priority 2:** We identify two classes of configurations for the expected collection in this operation. Consider an expected collection of three 2-paths P_1, P_2, P_3 and a 3-path $P_4 = u$-w-v in $\mathcal{Q}$.

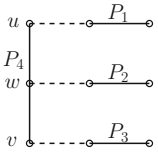
In the first class, which has priority 2.1, u, w, v are adjacent to an endpoint of P_1, P_2, P_3 in G, respectively (see Fig. 1c for an illustration). The operation breaks the 3-path u-w-v into three singletons and connects each of them to the respective 2-path to form the replacement collection of three new 3-paths.



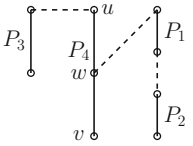
(a) The configuration of the expected collection for Operation 3-0-By-0-2.

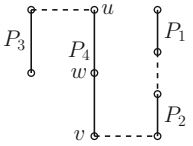


(b) A 3-path u-w-$v \in \mathcal{Q}$ can be rotated so that w is an endpoint if $(u, v) \in E$.

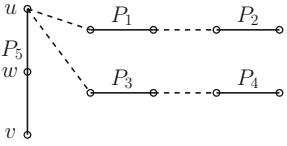


(c) The first class of configuration of the expected collection for Operation 3-1-By-0-3.





(d) The second class of configurations of the expected collection in Operation 3-1-By-0-3.



(e) The configuration of the expected collection for Operation 4-1-By-1-3.

Fig. 1. (a), (c–e) illustrate the configurations of the expected collections for the first three operations, where solid edges are in $\mathcal{Q}$ and dashed edges are in E but outside of $\mathcal{Q}$. (b) illustrates a rotated 3-path in $\mathcal{Q}$.

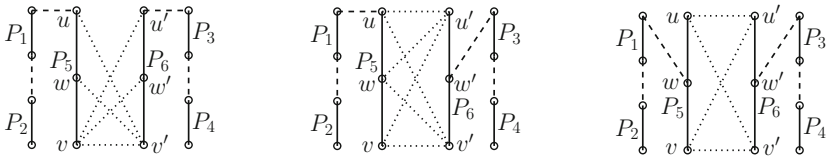
In the second class, which has priority 2.2, two of the three 2-paths, say P_1 and P_2, are adjacent and thus they can be replaced by a new 3-path and a singleton. We distinguish two configurations in this class (see Fig. 1d for illustrations). In the first configuration, the singleton is adjacent to the midpoint w and P_3 is adjacent to one of u and v; in the second configuration, the singleton and P_3 are adjacent to u and v, respectively. For an expected collection of either configuration, the operation replaces it by three new 3-paths.

Operation 4-1-By-1-3**, priority 3:** Consider an expected collection of four 2-paths P_1, P_2, P_3, P_4 and a 3-path $P_5 = u$-w-v in $\mathcal{Q}$. These four 2-paths can be separated into two pairs, each of which are adjacent in the graph G, thus we can replace them by two new 3-paths while leaving two singletons. In the

configuration for the expected collection in this operation, the two singletons are adjacent to a common endpoint, say u, of P_5 (see Fig. 1e for an illustration), and they can be replaced by a new 2-path v-w and a new 3-path. Overall, the operation replaces the expected collection by three new 3-paths and a new 2-path.

Operation 4-2-By-1-4**, lowest priority 4:** Consider an expected collection of four 2-paths P_1, P_2, P_3, P_4 and two 3-paths $P_5 = u$-w-v, $P_6 = u'$-w'-v' in $\mathcal{Q}$. These four 2-paths can be separated into two pairs, each of which are adjacent in the graph G, thus we can replace them by two new 3-paths while leaving two singletons. Each of P_5 and P_6 should be adjacent to at least one of the two singletons. We distinguish three classes of configurations for the expected collection in this operation, for which the replacement collection consists of four new 3-paths and a new 2-path.

In the first class, the two singletons are adjacent to P_5 and P_6 at endpoints, say u and u', respectively; additionally, one of the five edges (u, v'), (v, u'), (w, v'), (v, w'), (v, v') is in E (see Fig. 2a for an illustration).



(a) The first class. (b) The second class. (c) The third class.

Fig. 2. The three classes of configurations of the expected collections for an Operation 4-2-By-1-4, where solid edges are in $\mathcal{Q}$, dashed and dotted edges are in E but outside of $\mathcal{Q}$. In every class, each dotted edge between P_5 and P_6 corresponds to one configuration.

In the second class, one singleton is adjacent to an endpoint of a 3-path, say u on P_5, and the other singleton is adjacent to the midpoint w' of P_6; additionally, one of the six edges (u, u'), (u, v'), (w, u'), (w, v'), (v, u'), (v, v'), is in E (see Fig. 2b for an illustration).

In the third class, the two singletons are adjacent to the midpoints of the two 3-paths, w and w', respectively; additionally, one of the four edges (u, u'), (u, v'), (v, u'), (v, v') is in E (see Fig. 2c for an illustration).

In each of the above three classes of configurations, the operation replaces P_5, P_6, and the two singletons by two new 3-paths and one new 2-path.

2.2 The Complete Local Search Algorithm Approx

A high-level description of the complete algorithm Approx is depicted in Fig. 3. For the running time, Step 1 takes in $O(nm)$ time [2]. Note that there are $O(n)$ 2-paths and $O(n)$ 3-paths in $\mathcal{Q}$ at the beginning of Step 2, and therefore there

Algorithm Approx on $G = (V, E)$:

Step 1. compute a 3-path partition $\mathcal{Q}$ with the least 1-paths in G;
Step 2. Iteratively perform:
2.1. if Operation 3-0-By-0-2 applies, update $\mathcal{Q}$ and break;
2.2. if Operation 3-1-By-0-3 with priority 2.1 applies, update $\mathcal{Q}$ and break;
2.3. if Operation 3-1-By-0-3 with priority 2.2 applies, update $\mathcal{Q}$ and break;
2.4. if Operation 4-1-By-1-3 applies, update $\mathcal{Q}$ and break;
2.5. if Operation 4-2-By-1-4 applies, update $\mathcal{Q}$ and break;
Step 3. Return $\mathcal{Q}$.

Fig. 3. A high-level description of the algorithm Approx, where each "break" ends the current iteration of Step 2.

are $O(n^6)$ *original* candidate collections to be examined, since a candidate collection has a maximum size of 6. When a local operation applies, an iteration ends and the 3-path partition $\mathcal{Q}$ reduces its size by 1, while introducing at most 5 new 2- and 3-paths. These new 2- and 3-paths give rise to $O(n^5)$ *new* candidate collections to be examined in the subsequent iterations. Since there are at most n iterations in Step 2, we conclude that the total number of original and new candidate collections examined in Step 2 is $O(n^6)$. Determining whether a candidate collection is an expected collection, and if so, deciding the corresponding replacement collection, can be done in $O(1)$ time. We thus prove that the overall running time of Step 2 is $O(n^6)$, and consequently prove the following theorem.

Theorem 1. *The running time of the algorithm* Approx *is in* $O(n^6)$.

3 Analysis of the Approximation Ratio 4/3

In this section, we show that our local search algorithm Approx is a 4/3-approximation for 3PP. The performance analysis is done through amortization.

The 3-path partition produced by the algorithm Approx is denoted as $\mathcal{Q}$; let $\mathcal{Q}_i$ denote the sub-collection of i-paths in $\mathcal{Q}$, for $i = 1, 2, 3$, respectively. Let $\mathcal{Q}^*$ be an optimal 3-path partition, *i.e.*, it achieves the minimum total number of paths, and let $\mathcal{Q}_i^*$ denote the sub-collection of i-paths in $\mathcal{Q}^*$, for $i = 1, 2, 3$, respectively. Since our $\mathcal{Q}$ contains the least 1-paths among all 3-path partitions for G, we have $|\mathcal{Q}_1| \leq |\mathcal{Q}_1^*|$. Since both $\mathcal{Q}$ and $\mathcal{Q}^*$ cover all the vertices of V, we have $|\mathcal{Q}_1| + 2|\mathcal{Q}_2| + 3|\mathcal{Q}_3| = n = |\mathcal{Q}_1^*| + 2|\mathcal{Q}_2^*| + 3|\mathcal{Q}_3^*|$.

Next, we prove the following inequality which gives an upper bound on $|\mathcal{Q}_2|$, through an amortized analysis:

$$|\mathcal{Q}_2| \leq |\mathcal{Q}_1^*| + 2|\mathcal{Q}_2^*| + |\mathcal{Q}_3^*|. \tag{1}$$

It follows that $3|\mathcal{Q}_1| + 3|\mathcal{Q}_2| + 3|\mathcal{Q}_3| \leq 4|\mathcal{Q}_1^*| + 4|\mathcal{Q}_2^*| + 4|\mathcal{Q}_3^*|$, that is, $|\mathcal{Q}| \leq \frac{4}{3}|\mathcal{Q}^*|$, and consequently the following theorem holds.

Theorem 2. *The algorithm* Approx *is an* $O(n^6)$*-time 4/3-approximation for the* 3PP *problem, and the performance ratio 4/3 is tight for* Approx.

In the amortized analysis, each 2-path of $\mathcal{Q}_2$ has one token (*i.e.*, $|\mathcal{Q}_2|$ tokens in total) to be distributed to the paths of $\mathcal{Q}^*$. The upper bound in Eq. (1) will immediately follow if we prove the following lemma.

Lemma 1. *There is a distribution scheme in which*

1. *every* 1*-path of* $\mathcal{Q}_1^*$ *receives at most* 1 *token;*
2. *every* 2*-path of* $\mathcal{Q}_2^*$ *receives at most* 2 *tokens;*
3. *every* 3*-path of* $\mathcal{Q}_3^*$ *receives at most* 1 *token.*

In the rest of the section we present the distribution scheme that satisfies the three requirements stated in Lemma 1.

Denote $E(\mathcal{Q}_2)$, $E(\mathcal{Q}_3)$, $E(\mathcal{Q}_2^*)$, $E(\mathcal{Q}_3^*)$ as the set of all the edges on the paths of $\mathcal{Q}_2$, $\mathcal{Q}_3$, $\mathcal{Q}_2^*$, $\mathcal{Q}_3^*$, respectively, and $E(\mathcal{Q}^*) = E(\mathcal{Q}_2^*) \cup E(\mathcal{Q}_3^*)$. In the subgraph of $G\big(V, E(\mathcal{Q}_2) \cup E(\mathcal{Q}^*)\big)$, only the midpoint of a 3-path of $\mathcal{Q}_3^*$ may have degree 3, *i.e.*, incident with two edges of $E(\mathcal{Q}^*)$ and an edge of $E(\mathcal{Q}_2)$, while all the other vertices have degree at most 2 since each is incident with at most one edge of $E(\mathcal{Q}_2)$ and at most one edge of $E(\mathcal{Q}^*)$.

Our distribution scheme consists of two phases. We define two functions $\tau_1(P)$ and $\tau_2(P)$ to denote the fractional amount of token received by a path $P \in \mathcal{Q}^*$ in Phase 1 and Phase 2, respectively; we also define the function $\tau(P) = \tau_1(P) + \tau_2(P)$ to denote the total amount of token received by the path $P \in \mathcal{Q}^*$ at the end of our distribution process. Then, we have $\sum_{P \in \mathcal{Q}^*} \tau(P) = |\mathcal{Q}_2|$.

3.1 Token Distribution Phase 1

In Phase 1, we distribute all the $|\mathcal{Q}_2|$ tokens to the paths of $\mathcal{Q}^*$ (*i.e.*, $\sum_{P \in \mathcal{Q}^*} \tau_1(P) = |\mathcal{Q}_2|$) such that a path $P \in \mathcal{Q}^*$ receives some token from a 2-path u-$v \in \mathcal{Q}_2$ only if u or v is (or both are) on P, and the following three requirements are satisfied:

1. $\tau_1(P_i) \le 1$ for $\forall P_i \in \mathcal{Q}_1^*$;
2. $\tau_1(P_j) \le 2$ for $\forall P_j \in \mathcal{Q}_2^*$;
3. $\tau_1(P_\ell) \le 3/2$ for $\forall P_\ell \in \mathcal{Q}_3^*$.

In this phase, the one token held by each 2-path of $\mathcal{Q}_2$ is breakable but can only be broken into two halves. Thus for every path $P \in \mathcal{Q}^*$, $\tau_1(P)$ is a multiple of 1/2.

For each 2-path u-$v \in \mathcal{Q}_2$, at most one of u and v can be a singleton of $\mathcal{Q}^*$. If $P_1 = v \in \mathcal{Q}_1^*$, then the whole 1 token of the path u-v is distributed to v, that is, $\tau_1(v) = 1$ (see Fig. 4a for an illustration). This way, we have $\tau_1(P) \le 1$ for $\forall P \in \mathcal{Q}_1^*$.

For a 2-path u-$v \in \mathcal{Q}_2$, we consider the cases when both u and v are incident with an edge of $E(\mathcal{Q}^*)$. If one of u and v, say v, is incident with an edge of $E(\mathcal{Q}_2^*)$, that is, v is on a 2-path $P_1 = v$-$w \in \mathcal{Q}_2^*$, then the 1 token of the path u-v

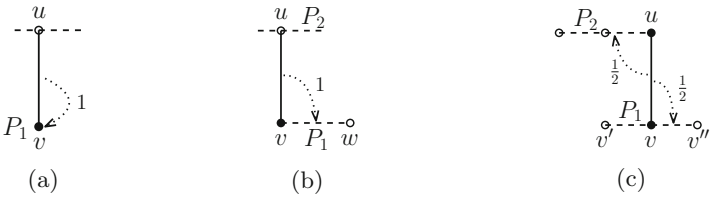


Fig. 4. Illustrations of the token distribution scheme in Phase 1, where solid edges are in $E(\mathcal{Q}_2)$ and dashed edges are in $E(\mathcal{Q}^*)$. In (c), u or v can be either an endpoint or the midpoint of the corresponding 3-path of $\mathcal{Q}_3^*$.

is given to the path $P_1 \in \mathcal{Q}_2^*$ (see Fig. 4b for an illustration). Note that if u is also on a 2-path $P_2 \in \mathcal{Q}_2^*$ and $P_2 \neq P_1$, then the path P_2 receives no token from the path u-v. The choice of which of the two vertices u and v comes first does not matter. This way, we have $\tau_1(P) \leq 2$ for $\forall P \in \mathcal{Q}_2^*$ since the 2-path $P_1 \in \mathcal{Q}_2^*$ might receive another token from a 2-path of $\mathcal{Q}_2$ incident at w.

Next, we consider the cases for a 2-path u-$v \in \mathcal{Q}_2$ in which each of u and v is incident with an edge of $E(\mathcal{Q}_3^*)$. Consider a 3-path $P_1 \in \mathcal{Q}_3^*$: v'-v-v''. We distinguish two cases for a vertex of P_1 to determine the amount of token received by P_1 (see Fig. 4c for an illustration). In the first case, either the vertex, say v', is not on any path of $\mathcal{Q}_2$ or it is on a path of $\mathcal{Q}_2$ with 0 token left, then P_1 receives no token *through vertex* v'. In the second case, the vertex, say v (the following argument also applies to the other two vertices v' and v''), is on a path u-$v \in \mathcal{Q}_2$ holding 1 token, and consequently u must be on a 3-path $P_2 \in \mathcal{Q}_3^*$, then the 1 token of u-v is broken into two halves, with 1/2 token distributed to P_1 through vertex v and the other 1/2 token distributed to P_2 through vertex u. This way, we have $\tau_1(P) \leq 3/2$ for $\forall P \in \mathcal{Q}_3^*$ since the 3-path $P_1 \in \mathcal{Q}_3^*$ might receive another 1/2 token through each of v' and v''.

3.2 Token Distribution Phase 2

In Phase 2, we will transfer the extra 1/2 token from every 3-path $P \in \mathcal{Q}_3^*$ with $\tau_1(P) = 3/2$ to some other paths of $\mathcal{Q}^*$ in order to satisfy the three requirements of Lemma 1. In this phase, each 1/2 token can be broken into two quarters, thus for a path $P \in \mathcal{Q}^*$, $\tau_2(P)$ is a multiple of 1/4.

Consider a 3-path $P_1 = v''$-v'-$v \in \mathcal{Q}_3^*$. We observe that if $\tau_1(P_1) = 3/2$, then each of v, v', and v'' must be incident with an edge of $E(\mathcal{Q}_2)$, the other endpoint of which must also be on a 3-path of $\mathcal{Q}_3^*$. One of the three vertices, say v, on an edge $(u, v) \in E(\mathcal{Q}_2)$, must have its corresponding u outside of P_1. Denote P_2 as the 3-path of $\mathcal{Q}_3^*$ where u is on. Let w be a vertex adjacent to u on P_2. We can verify that due to $\mathcal{Q}$ being a partition with the least 1-paths and by Operation 3-0-By-0-2, w cannot be a singleton of $\mathcal{Q}_1$ or on any 2-path of $\mathcal{Q}_2$, and thus it must be on a 3-path of $\mathcal{Q}_3$, being either an endpoint or the midpoint (see Fig. 5 for an illustration). We thus conclude that $\tau_1(P_2) \leq 1$, and we have the following lemma.

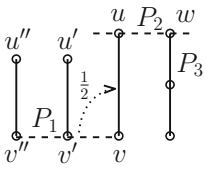


Fig. 5. An illustration of a 3-path $P_1 = v\text{-}v'\text{-}v'' \in \mathcal{Q}_3^*$ with $\tau_1(P_1) = 3/2$, where u-v, u'-v', u''-$v'' \in E(\mathcal{Q}_2)$, $P_3 \in \mathcal{Q}_3$, with w being either the midpoint or an endpoint of P_3, and $P_2 \in E(\mathcal{Q}_3^*)$ is represented by dashed edges, on which w is adjacent to u.

Lemma 2. *For any 3-path $P_1 \in \mathcal{Q}_3^*$ with $\tau_1(P_1) = 3/2$, there must be another 3-path $P_2 \in \mathcal{Q}_3^*$ with $\tau_1(P_2) \leq 1$ such that*

1. *u-v is a 2-path of $\mathcal{Q}_2$, where v is on P_1 and u is on P_2, and*
2. *any vertex adjacent to u on P_2 must be on a 3-path P_3 of $\mathcal{Q}_3$.*

The first step of Phase 2 is to transfer this extra 1/2 token back from P_1 to the 2-path u-v through vertex v (see Fig. 5 for an illustration). Thus, we have $\tau_2(P_1) = -1/2$ and $\tau(P_1) = 3/2 - 1/2 = 1$.

Using Lemma 2 and its notation, let x_1 and y_1 be the other two vertices on P_3 ($P_3 = w$-x_1-y_1 or $P_3 = x_1$-w-y_1). Denote $P_4 \in \mathcal{Q}^*$ ($P_5 \in \mathcal{Q}^*$, respectively) as the path where x_1 (y_1, respectively) is on. Next, we will transfer the 1/2 token from u-v to the paths P_4 or/and P_5 through some *pipe* or *pipes*.

We define a *pipe* $r \to s \to t$, where r is an endpoint of a 2-path of $\mathcal{Q}_2$ which receives 1/2 token in the first step of Phase 2, (r, s) is an edge on a 3-path $P' \in \mathcal{Q}_3^*$ with $\tau_1(P') \leq 1$ ($P' = P_2$ here), s and t are both on a 3-path of $\mathcal{Q}_3$ (P_3 here), and t is a vertex on our destination path of $\mathcal{Q}^*$ (P_4 or P_5 here) which will receive token from the 2-path of $\mathcal{Q}_2$. r and t are called the *head* and *tail* of the pipe, respectively. For example, in Fig. 6a, there are four possible pipes $u \to w \to x_1$, $u \to w \to y_1$, $u'' \to w \to x_1$, and $u'' \to w \to y_1$. We distinguish the cases, in which the two paths P_4 and P_5 belong to different combinations of $\mathcal{Q}_1^*$, $\mathcal{Q}_2^*$, $\mathcal{Q}_3^*$, to determine how they receive more token through some pipe or pipes.

Recall that u can be either an endpoint or the midpoint of P_2. We discuss the cases with u being an endpoint of P_2 (the cases for u being the midpoint can be discussed the same), that is, $P_2 = u$-w-u''. The following is a summary of our discussion and results ([1] contains the full details).

Case 1. At least one of P_4 and P_5 is a singleton of $\mathcal{Q}_1^*$, say $P_4 = x_1 \in \mathcal{Q}_1^*$ (see Fig. 6 for illustrations). In this case, $\tau_1(P_4) = 0$, and we transfer the 1/2 token from u-v to P_4 through pipe $u \to w \to x_1$. We prove in [1] that there are at most two pipes with tail x_1 through each of which could P_4 receive 1/2 token, due to Operation 3-1-By-0-3 and Operation 4-1-By-1-3. That is, we have $\tau_2(P_4) \leq 1/2 \times 2 = 1$, implying $\tau(P_4) \leq 0 + 1 = 1$.

Case 2. Both P_4 and P_5 are paths of $\mathcal{Q}_2^* \cup \mathcal{Q}_3^*$. In this case, if w is an endpoint of $P_3 = w$-x_1-y_1, with y_1 on P_5, we transfer the 1/2 token from u-v to P_5 through

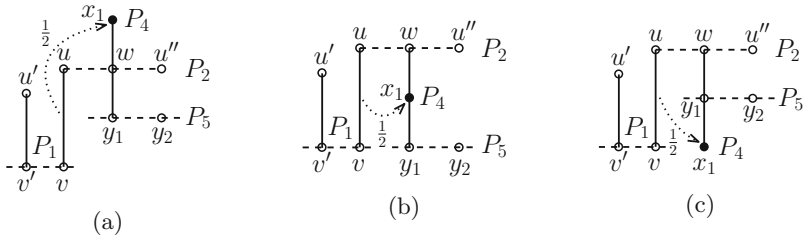


Fig. 6. The cases when P_4 is a singleton of $\mathcal{Q}_1^*$, where solid edges are in $E(\mathcal{Q}_2)$ or $E(\mathcal{Q}_3)$ and dashed edges are in $E(\mathcal{Q}^*)$. x_1 is the tail of the pipe through which P_4 could receive 1/2 token from the 2-path of u-v.

pipe $u \to w \to y_1$ (see Fig. 7a for an illustration); if w is the midpoint of P_3, we transfer 1/4 token from u-v to P_4 through pipe $u \to w \to x_1$ and the other 1/4 token to P_5 through pipe $u \to w \to y_1$ (see Fig. 7b for an illustration). We prove in [1] that for any $P \in \{P_4, P_5\}$, if $\tau_2(P) > 0$, then we have $\tau_1(P) \leq 1/2$ and $\tau_2(P) \leq 1$, and it falls into one of the following four cases:

1. If w is an endpoint of P_3 and $\tau_1(P) = 0$, then there are at most two pipes through each of which could P receive 1/2 token. That is, $\tau_2(P) \leq 1/2 \times 2 = 1$, implying $\tau(P) \leq 0 + 1 = 1$.
2. If w is an endpoint of P_3 and $\tau_1(P) = 1/2$, then only through one pipe could P receive the 1/2 token. That is, $\tau_2(P) \leq 1/2$, implying $\tau(P) \leq 1/2 + 1/2 = 1$.
3. If w is the midpoint of P_3 and $\tau_1(P) = 0$, then there are at most four pipes through each of which could P receive 1/4 token. That is, $\tau_2(P) \leq 1/4 \times 4 = 1$, implying $\tau(P) \leq 0 + 1 = 1$.
4. If w is the midpoint of P_3 and $\tau_1(P) = 1/2$, then there are at most two pipes through each of which could P receive 1/4 token. That is, $\tau_2(P) \leq 1/4 \times 2 = 1/2$, implying $\tau(P) \leq 1/2 + 1/2 = 1$.

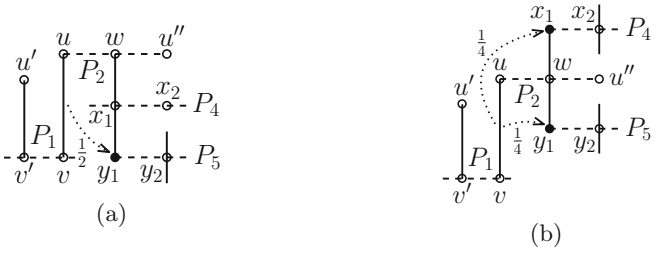


Fig. 7. The cases when both P_4 and P_5 are in $\mathcal{Q}_2^* \cup \mathcal{Q}_3^*$, where solid edges are in $E(\mathcal{Q}_2)$ or $E(\mathcal{Q}_3)$ and dashed edges are in $E(\mathcal{Q}^*)$. In (a), y_1 is the tail of the pipe through which P_5 receives 1/2 token from the 2-path u-v; in (b), x_1 is the tail of the pipe through which P_4 receives 1/4 token from the 2-path u-v and y_1 is the tail of the pipe through which P_5 receives 1/4 token from the 2-path u-v.

In summary, for any $P_1 \in \mathcal{Q}^*$ with $\tau_1(P_1) = 3/2$, we have $\tau_2(P_1) = -1/2$; for any $P \in \mathcal{Q}^*$ with $\tau_2(P) > 0$, we have $\tau_1(P) = 0$ if $\tau_2(P) \leq 1$, or $\tau_1(P) \leq 1/2$ if $\tau_2(P) \leq 1/2$. Therefore, at the end of Phase 2, we have

1. $\tau(P_i) \leq 1$ for $\forall P_i \in \mathcal{Q}_1^*$,
2. $\tau(P_j) \leq 2$ for $\forall P_j \in \mathcal{Q}_2^*$,
3. $\tau(P_\ell) \leq 1$ for $\forall P_\ell \in \mathcal{Q}_3^*$.

This proves Lemma 1.

3.3 A Tight Instance for Approx

Figure 8 illustrates a tight instance, in which our solution 3-path partition $\mathcal{Q}$ contains nine 2-paths and three 3-paths (solid edges) and an optimal 3-path partition $\mathcal{Q}^*$ contains nine 3-paths (dashed edges). Each 3-path of $\mathcal{Q}^*$ receives 1 token from the 2-paths in $\mathcal{Q}$ in our distribution process. This instance shows that the performance ratio of 4/3 is tight for Approx.

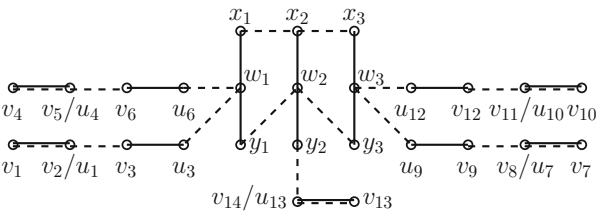


Fig. 8. A tight instance of 27 vertices, where solid edges represent a 3-path partition produced by Approx and dashed edges represent an optimal 3-path partition. The edges (u_{3i+1}, v_{3i+1}), $i = 0, 1, \ldots, 4$, are in $E(\mathcal{Q}_2) \cap E(\mathcal{Q}^*)$, shown in both solid and dashed. The vertex u_{3i+1} collides into v_{3i+2}, $i = 0, 1, \ldots, 4$. In our distribution process, each of the nine 3-paths in $\mathcal{Q}^*$ receives 1 token from the 2-paths in $\mathcal{Q}$.

Lemma 1 and the tight instance shown above together prove Theorem 2.

4 Conclusions

We studied the 3PP problem and designed a 4/3-approximation algorithm Approx. Approx first computes a 3-path partition $\mathcal{Q}$ with the least 1-paths in $O(nm)$-time, then iteratively applies four local operations with different priorities to reduce the total number of paths in $\mathcal{Q}$. The overall running time of Approx is $O(n^6)$. The performance ratio 4/3 of Approx is proved through an amortization scheme, using the structure properties of the 3-path partition returned by Approx. We also show that the performance ratio 4/3 is tight for our algorithm.

The 3PP problem is closely related to the 3-Set Cover problem, but none is a special case of the other. The best 4/3-approximation for 3-Set Cover has stood there for more than three decades; our algorithm Approx for 3PP has the approximation ratio matches up to this best approximation ratio 4/3. We leave it open to better approximate 3PP.

Acknowledgement. YC and AZ were supported by the NSFC Grants 11771114 and 11571252; YC was also supported by the China Scholarship Council Grant 201508330054. RG, GL and YX were supported by the NSERC Canada. LL was supported by the China Scholarship Council Grant No. 201706315073, and the Fundamental Research Funds for the Central Universities Grant No. 20720160035. WT was supported in part by funds from the College of Engineering and Computing at the Georgia Southern University.

References

1. Chen, Y., et al.: A local search 4/3-approximation algorithm for the minimum 3-path partition problem. arXiv:1812.09353 (2018)
2. Chen, Y., Goebel, R., Lin, G., Su, B., Xu, Y., Zhang, A.: An improved approximation algorithm for the minimum 3-path partition problem. J. Comb. Optim. (2018). https://doi.org/10.1007/s10878-018-00372-z
3. Duh, R., Fürer, M.: Approximation of k-set cover by semi-local optimization. In: Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing (STOC 1997), pp. 256–264 (1997)
4. Feige, U.: A threshold of for approximating set cover. J. ACM **45**, 634–652 (1998)
5. Franzblau, D.S., Raychaudhuri, A.: Optimal hamiltonian completions and path covers for trees, and a reduction to maximum flow. ANZIAM J. **44**, 193–204 (2002)
6. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-completeness. W. H. Freeman and Company, San Francisco (1979)
7. Goldberg, A.V., Karzanov, A.V.: Maximum skew-symmetric flows and matchings. Math. Program. **100**, 537–568 (2004)
8. Johnson, D.S.: Approximation algorithms for combinatorial problems. J. Comput. Syst. Sci. **9**, 256–278 (1974)
9. Karp, R.M.: Reducibility among combinatorial problems. In: Miller, R.E., Thatcher, J.W., Bohlinger, J.D. (eds.) Complexity of Computer Computations, pp. 85–103. Springer, Boston (1972). https://doi.org/10.1007/978-1-4684-2001-2_9
10. Levin, A.: Approximating the unweighted k-set cover problem: greedy meets local search. In: Erlebach, T., Kaklamanis, C. (eds.) WAOA 2006. LNCS, vol. 4368, pp. 290–301. Springer, Heidelberg (2007). https://doi.org/10.1007/11970125_23
11. Monnot, J., Toulouse, S.: The path partition problem and related problems in bipartite graphs. Oper. Res. Lett. **35**, 677–684 (2007)
12. Raz, R., Safra, S.: A sub-constant error-probability low-degree test, and sub-constant error-probability PCP characterization of NP. In: Proceedings of the 29th Annual ACM Symposium on Theory of Computing (STOC 1997), pp. 475–484 (1997)
13. Vazirani, V.: Approximation Algorithms. Springer, Heidelberg (2001). https://doi.org/10.1007/978-3-662-04565-7
14. Yan, J.-H., Chang, G.J., Hedetniemi, S.M., Hedetniemi, S.T.: k-path partitions in trees. Discrete Appl. Math. **78**, 227–233 (1997)