Journal of Combinatorial Optimization (2022) 44:3595–3610
https://doi.org/10.1007/s10878-022-00915-5

Check for updates

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[3] · Bing Su[4] · Weitian Tong[5] · Yao Xu[5] · An Zhang[1]

Accepted: 19 September 2022 / Published online: 28 September 2022

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. The previous best approximation algorithm for the 3-path partition problem has a performance ratio $13/9$, which is based on a simple local search strategy. We propose a more involved local search and show by an amortized analysis that it is a $4/3$-approximation; we also design an instance to illustrate that the approximation ratio is tight.

Keywords Path partition · Path cover · Local search · Approximation algorithms · 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. (1997). Given a simple graph $G = (V, E)$ (we consider only simple graphs and we drop "simple" hereafter), 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.

An extended abstract (Chen et al. 2019a) of this paper appears in the *Proceedings of 13th International Frontiers of Algorithmics Workshop (FAW 2019)*, LNCS 11458, pages 14–25.

✉ Guohui Lin
guohui@ualberta.ca

✉ Bing Su
subing684@sohu.com

Extended author information available on the last page of the article

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 (Goldberg and Karzanov 2004). For each $k \geq 3$, kPP is NP-hard (Garey and Johnson 1979). We point out the key phrase "at most k" in the problem definition, that 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 $k \geq 3$ (Garey and Johnson 1979), even on bipartite graphs of maximum degree three (Monnot and Toulouse 2007). To the best of our knowledge, there is no prior approximation algorithm with proven performance for the general kPP problem, except the trivial k-approximation using all 1-paths. For 3PP, Monnot and Toulouse (2007) proposed a 3/2-approximation, based on two maximum matchings; recently, Chen et al. (2019b) presented an improved 13/9-approximation, based on a simple local search strategy (which is briefly reviewed below).

The kPP problem is a generalization to the PATH COVER problem (Franzblau and Raychaudhuri 2002) (also called PATH PARTITION), which is to find a minimum collection of vertex-disjoint paths that together cover all the vertices in the graph. PATH COVER contains the HAMILTONIAN PATH problem (Garey and Johnson 1979) 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 (Garey and Johnson 1979), and is also one of the most studied optimization problems for the approximability (Johnson 1974) and inapproximability (Raz and Safra 1997; Feige 1998; Vazirani 2003). 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 $k = 3$ (Duh and Fürer 1997) and an $(H_k - \frac{196}{390})$-approximation for $k \geq 4$ (Levin 2006), where H_k is the k-th harmonic number.

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 (Karp 1972), 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 the given collection $\mathcal{C}$ to include all the proper subsets of each given subset $S_j \in \mathcal{C}$. However, in an instance graph of kPP, not every subset of vertices on a path is traceable, and hence 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 (Chen et al. 2019b) 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 local search strategy to find three 2-paths that can be replaced by two 3-paths, until not 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 more involved local search scheme to repeatedly search for a *collection* of 2- and 3-paths that can be replaced by a *strictly smaller collection* of new 2- and 3-paths, so as to reduce the size of the 3-path partition, until not possible. One thus may view our local search as a refinement of the previous work; using a slightly more complex amortized analysis, we are able to prove the performance ratio $4/3$.

The rest of the paper is organized as follows. In Sect. 2 we present the local search scheme for searching a candidate collection of 2- and 3-paths. The amortized analysis and the proof of the performance of the algorithm are done in Sect. 3, and at the end we design an instance to show that the approximation ratio $4/3$ is tight. We conclude the paper in Sect. 4.

2 A local search approximation algorithm

The $13/9$-approximation proposed by Chen et al. (2019b) applies a simple local search strategy to iteratively find three 2-paths in the current solution that can be replaced by two 3-paths (Operation 3- 0- By- 0- 2 in Definition 1). Such a replacement operation looks at only those six vertices covered by the three 2-paths, but no more. In our more involved local search, we examine three more possibilities where three 2-paths can be *transferred* into two 3-paths, either alone or with the help of a few other 2-paths and/or 3-paths. Specifically, we look for the help either from a single 3-path (Operation 3- 1- By- 0- 3 in Definition 1), or from a combination of a 2-path and a 3-path (Operation 4- 1- By- 1- 3 in Definition 1), or from a combination of a 2-path and two 3-paths (Operation 4- 2- By- 1- 4 in Definition 1). We show that these three additional replacement operations are sufficient to achieve the performance ratio of 4/3.

Starting with a 3-path partition with the least 1-paths, our approximation algorithm repeatedly finds a certain collection of 2- and 3-paths in the current solution and replaces it by another collection of one less new 2- and 3-paths. This way, the new solution is better and the algorithm continues on until further reduction is impossible.

In Sect. 2.1 we present all the four replacement operations to be executed 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

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. We remind the reader that the local search algorithm is iterative, and every iteration ends after executing a local

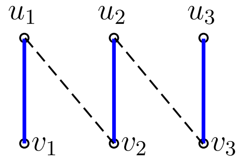

Fig. 1 A configuration of a candidate collection for the OPERATION 3- 0- BY- 0- 2, where blue solid edges are in $\mathcal{Q}$ and black dashed edges are in E but outside of $\mathcal{Q}$ (Color figure online)

operation. The algorithm terminates when none of the designed local operations is applicable.

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 a collection of i_1 2-paths and i_2 3-paths in $\mathcal{Q}$ by a collection of j_1 2-paths and j_2 3-paths on the same subset of $2i_1 + 3i_2$ vertices.

For ease of presentation, the collection of i_1 2-paths and i_2 3-paths in the current 3-path partition $\mathcal{Q}$ is referred to as a *candidate* collection; while the latter collection of j_1 2-paths and j_2 3-paths on the same subset of $2i_1 + 3i_2$ vertices is referred to as a *replacement* collection.

We remark that by such a local OPERATION i_1-i_2- BY- j_1-j_2, three 2-paths in the candidate collection are transferred into two 3-paths, with the help of the other 2-paths and the 3-paths in the candidate collection. One clearly sees that the achieved 3-path partition contains exactly one less path than $\mathcal{Q}$ (since $j_1 + j_2 = i_1 + i_2 - 1$).

In the rest of this section we determine the configurations for all the local operations.

2.1.1 OPERATION **3- 0**- BY- **0- 2**

When three 2-paths of $\mathcal{Q}$ can be connected into a 6-path in the graph G (see Fig. 1 for an illustration), they form a candidate collection satisfying Definition 1. By removing the middle edge on the 6-path, we achieve two 3-paths on the same six vertices which replace the original three 2-paths. In the example illustrated in Fig. 1, using the two edges $(u_1, v_2), (u_2, v_3) \in E$ outside of $\mathcal{Q}$ (shown as black dashed), the OPERATION 3- 0- BY- 0- 2 replaces the three 2-paths u_1-v_1, u_2-v_2, and u_3-v_3 of $\mathcal{Q}$ by two new 3-paths v_1-u_1-v_2 and u_2-v_3-u_3.

Another way to look at this candidate collection $\{u_1\text{-}v_1, u_2\text{-}v_2, u_3\text{-}v_3\}$ is, supposing the two 2-paths u_2-v_2 and u_3-v_3 are connected via the edge (u_2, v_3), the 2-path u_1-v_1 *directly* reaches it via the edge (u_1, v_2).

Assume a 3-path u-w-$v \in \mathcal{Q}$. Clearly, if $(u, v) \in E$, then one can replace u-w-v by w-v-u or v-u-w in $\mathcal{Q}$ so that the three vertices remain covered by the same path. In this sense, we say that these three paths are *equivalent* to each other, and if needed, any one of them can replace the other to present in $\mathcal{Q}$ (see Fig. 2 for an illustration).

2.1.2 OPERATION **3- 1**- BY- **0- 3**

Consider a collection of three 2-paths u_1-v_1, u_2-v_2, u_3-v_3, and a 3-path u-w-v in $\mathcal{Q}$. If one of the 2-paths, say u_1-v_1, is adjacent to an endpoint of the 3-path, say $(u_1, u) \in E$ (see Fig. 3 for an illustration), then we can remove the edge (u, w) while add the edge

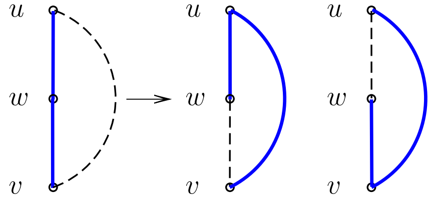

Fig. 2 If $(u, v) \in E$, then the 3-path u-w-$v \in \mathcal{Q}$ can be replaced by either w-v-u or v-u-w when needed, where blue solid edges are in $\mathcal{Q}$ and black dashed edges are in E but outside of $\mathcal{Q}$ (Color figure online)

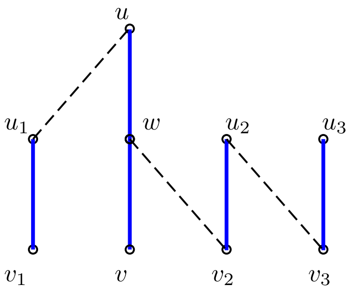

Fig. 3 A configuration of a candidate collection for the Operation 3- 1- By- 0- 3, where blue solid edges are in $\mathcal{Q}$ and black dashed edges are in E but outside of $\mathcal{Q}$ (Color figure online)

(u, u_1) to transfer the 2-path u_1-v_1 and the 3-path u-w-v into a new 3-path u-u_1-v_1 and a new 2-path w-v. If follows that, when an Operation 3- 0- By- 0- 2 is applicable to the three 2-paths u_2-v_2, u_3-v_3, and w-v, then the original collection $\{u_1$-v_1, u_2-v_2, u_3-v_3, u-w-$v\}$ is a candidate collection for applying an Operation 3- 1- By- 0- 3. In the example illustrated in Fig. 3, the replacement collection is $\{u$-u_1-v_1, v-w-v_2, u_2-v_3-$u_3\}$.

Similarly, another way to look at this candidate collection $\{u_1$-v_1, u_2-v_2, u_3-v_3, u-w-$v\}$ is, supposing the two 2-paths u_2-v_2 and u_3-v_3 are connected via the edge (u_2, v_3), the 2-path u_1-v_1 reaches it *indirectly* to the 3-path via the edge (u_1, u) first and then via the edge (w, v_2). That is, the 3-path u-w-v offers help for the operation.

2.1.3 Operation 4- 1- By- 1- 3

Assume the two 2-paths u_1-v_1 and u_2-v_2 of $\mathcal{Q}$ are connected via the edge (v_1, u_2), and the other two 2-paths u_3-v_3 and u_4-v_4 of $\mathcal{Q}$ are connected via the edge (v_3, u_4). If both u_1 and u_3 are adjacent to an endpoint of a 3-path, say the vertex u of u-w-v (see Fig. 4 for an illustration), then these four 2-paths and the 3-path form a candidate collection. By removing the three edges (u, w), (u_1, v_1), (u_3, v_3) while adding the four edges (u_1, u), (u, u_3), (v_1, u_2), (v_3, u_4), an Operation 4- 1- By- 1- 3 transfers the candidate collection into the replacement collection $\{w$-v, u_1-u-u_3, v_1-u_2-v_2, v_3-u_4-$v_4\}$.

In such an operation, the 3-path u-w-v offers help by breaking itself into a 2-path w-v and a singleton u, the latter of which connects the two 4-paths into a 9-path (which is broken down into three 3-paths afterwards).

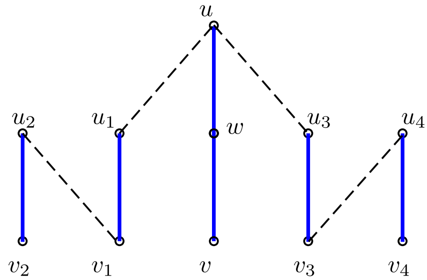

Fig. 4 A configuration of a candidate collection for the Operation 4-1-By-1-3, where blue solid edges are in $\mathcal{Q}$ and black dashed edges are in E but outside of $\mathcal{Q}$ (Color figure online)

2.1.4 Operation 4-2-By-1-4

Assume the two 2-paths u_1-v_1 and u_2-v_2 of $\mathcal{Q}$ are connected via the edge (v_1, u_2), and the other two 2-paths u_3-v_3 and u_4-v_4 of $\mathcal{Q}$ are connected via the edge (v_3, u_4). If u_1 is adjacent to a vertex on a 3-path u-w-v and u_3 is adjacent to a vertex on another 3-path u'-w'-v', then we examine whether there is an edge connecting these two 3-paths so that an Operation 4-2-By-1-4 transfers the collection $\{u_1$-v_1, u_2-v_2, u_3-v_3, u_4-v_4, u-w-v, u'-w'-$v'\}$ into a replacement collection of a 2-path and four 3-paths.

Depending on how the vertices u_1 and u_3 are adjacent to the vertices on the 3-paths u-w-v and u'-w'-v', respectively, there are three possible classes of configurations. In the first class, both u_1 and u_3 are adjacent to endpoints, say $(u_1, u), (u_3, u') \in E$. Then, the existence of one of the five edges (u, v'), (v, u'), (w, v'), (v, w'), (v, v') enables the Operation 4-2-By-1-4 (see Fig. 5a for an illustration). For example, when $(u, v') \in E$, removing the four edges (u, w), (w', v'), (u_1, v_1), (u_3, v_3) while adding the five edges (u_1, u), (u_3, u'), (u, v'), (v_1, u_2), (v_3, u_4), transfer the candidate collection into the replacement collection $\{w$-v, u_1-u-v', w'-u'-u_3, v_1-u_2-v_2, v_3-u_4-$v_4\}$.

In the second class, u_1 and u_3 are adjacent to an endpoint and an midpoint, respectively, say $(u_1, u), (u_3, w') \in E$. Then, the existence of one of the six edges (u, u'), (v, v'), (u, v'), (v, u'), (w, u'), (w, v') enables the Operation 4-2-By-1-4 (see Fig. 5b for an illustration). For example, when $(u, v') \in E$, removing the four edges (u, w), (w', v'), (u_1, v_1), (u_3, v_3) while adding the five edges (u_1, u), (u_3, w'), (u, v'), (v_1, u_2), (v_3, u_4), transfer the candidate collection into the replacement collection $\{w$-v, u_1-u-v', u'-w'-u_3, v_1-u_2-v_2, v_3-u_4-$v_4\}$.

In the last class, both u_1 and u_3 are adjacent to midpoints, that is, $(u_1, w), (u_3, w') \in E$. Then, the existence of one of the four edges (u, u'), (v, v'), (u, v'), (v, u') enables the Operation 4-2-By-1-4 (see Fig. 5c for an illustration). For example, when $(u, v') \in E$, removing the four edges (u, w), (w', v'), (u_1, v_1), (u_3, v_3) while adding the five edges (u_1, w), (u_3, w'), (u, v'), (v_1, u_2), (v_3, u_4), transfer the candidate collection into the replacement collection $\{u$-v', u_1-w-v, u'-w'-u_3, v_1-u_2-v_2, v_3-u_4-$v_4\}$.

2.2 The complete local search algorithm Approx

The first step of our local search algorithm Approx is to compute a 3-path partition $\mathcal{Q}$ with the least 1-paths. The second step is iterative, and in each iteration the algorithm

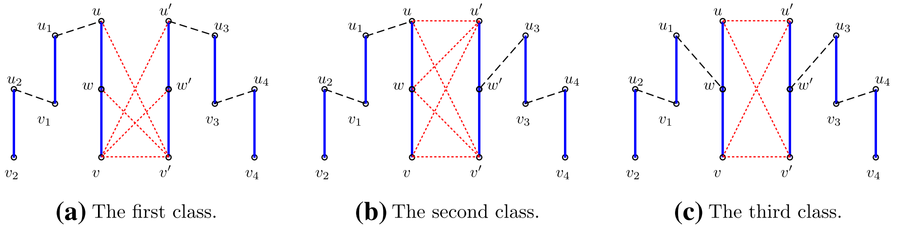

Fig. 5 The three classes of configurations of a candidate collection for the OPERATION 4-2-BY-1-4, where blue solid edges are in $\mathcal{Q}$, black dashed edges are in E but outside of $\mathcal{Q}$, and for each class at least one red dotted edge is in E but outside of $\mathcal{Q}$ (Color figure online)

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, if an OPERATION i_1-i_2-BY-j_1-j_2 is applicable, update $\mathcal{Q}$;

Step 3. return $\mathcal{Q}$.

Fig. 6 A high-level description of the local search algorithm APPROX

tries to apply one of the four local operations, by finding a candidate collection and determining the subsequent replacement collection. When no candidate collection can be found, the second step terminates. The algorithm outputs the achieved 3-path partition $\mathcal{Q}$ as the solution. A high-level description of the complete algorithm APPROX is depicted in Fig. 6.

Step 1 runs in $O(nm)$ time (Chen et al. 2019b), where $n = |V|$ and $m = |E|$. Note that there are $O(n)$ 2-paths and $O(n)$ 3-paths in the $\mathcal{Q}$ at the beginning of Step 2, and therefore there are $O(n^6)$ *original* collections to be examined, since a candidate collection has a maximum size of 6. When a local operation is applied, the iteration ends and the 3-path partition $\mathcal{Q}$ reduces its size by 1, while introducing at most 5 new 2-paths and 3-paths. These new 2-paths and 3-paths give rise to $O(n^5)$ *new* collections (each contains at least one of the new paths) 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 collections examined in Step 2 is $O(n^6)$. Determining whether a collection is a candidate collection, and if so, deciding the corresponding replacement collection, can be done in $O(1)$ time. We thus conclude that the overall running time of Step 2 is $O(n^6)$, and consequently have proved 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 all the 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 all the 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

$$|\mathcal{Q}_1| \le |\mathcal{Q}_1^*|. \tag{1}$$

Since both $\mathcal{Q}$ and $\mathcal{Q}^*$ cover all the vertices of V, we have

$$|\mathcal{Q}_1| + 2|\mathcal{Q}_2| + 3|\mathcal{Q}_3| = n = |\mathcal{Q}_1^*| + 2|\mathcal{Q}_2^*| + 3|\mathcal{Q}_3^*|. \tag{2}$$

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

$$|\mathcal{Q}_2| \le |\mathcal{Q}_1^*| + 2|\mathcal{Q}_2^*| + |\mathcal{Q}_3^*|. \tag{3}$$

Combining Eqs. (1, 2, 3), it follows that

$$3|\mathcal{Q}_1| + 3|\mathcal{Q}_2| + 3|\mathcal{Q}_3| \le 4|\mathcal{Q}_1^*| + 4|\mathcal{Q}_2^*| + 4|\mathcal{Q}_3^*|, \tag{4}$$

that is, $|\mathcal{Q}| \le \frac{4}{3}|\mathcal{Q}^*|$, 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. (3) will immediately follow if we prove the following lemma.

Lemma 1 *There is a token 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.*, it is incident with two edges of $E(\mathcal{Q}^*)$ and one 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; then $\tau(P) = \tau_1(P) + \tau_2(P)$ is the total amount of token received by the path $P \in \mathcal{Q}^*$ at the end of our distribution process. Recall that $\sum_{P \in \mathcal{Q}^*} \tau(P) = |\mathcal{Q}_2|$.

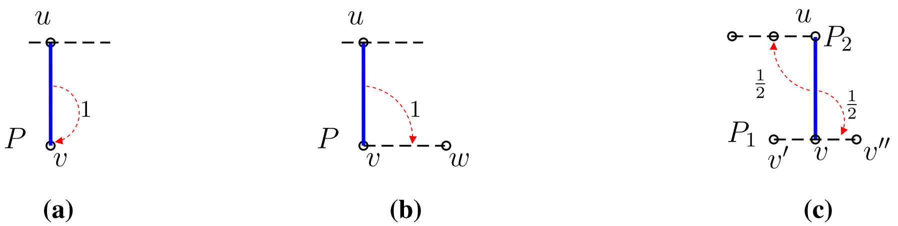

Fig. 7 Illustrations of the token distribution scheme in Phase 1, where blue solid edges are in $E(\mathcal{Q}_2)$, black dashed edges are in $E(\mathcal{Q}^*)$, and red dotted arrows indicate how tokens are distributed. In (**a**), $P = v \in \mathcal{Q}_1^*$; in (**b**), $P = v\text{-}w \in \mathcal{Q}_2^*$; in (**c**), both v and u are on a 3-path of $\mathcal{Q}_3^*$ (Color figure online)

3.1 Distribution process 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 $u\text{-}v \in \mathcal{Q}_2$ if and only if u or v is (or both are) on P, and the following three requirements are satisfied:

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

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. So for every path $P \in \mathcal{Q}^*$, $\tau_1(P)$ is a multiple of $1/2$.

For each 2-path $u\text{-}v \in \mathcal{Q}_2$, if one vertex v or u lies on a singleton ($P = v$ in Fig. 7a) or a 2-path ($P = v\text{-}w$ in Fig. 7b) of $\mathcal{Q}^*$, then $u\text{-}v$ gives its whole token to the singleton or the 2-path; otherwise, both v and u lie on a 3-path (P_1 and P_2, respectively, in Fig. 7c) of $\mathcal{Q}^*$, and then the whole token of $u\text{-}v$ is broken into two halves, one for each of the two 3-paths. This way, we distribute the tokens to the paths of $\mathcal{Q}^*$ satisfying the above three requirements.

3.2 Distribution process 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 $P_1 = v''\text{-}v'\text{-}v \in \mathcal{Q}_3^*$. We observe that if $\tau_1(P_1) = 3/2$, then each of v, v', and v'' must be incident with an edge $e \in E(\mathcal{Q}_2)$, such that the other endpoint of the edge e is also on a 3-path of $\mathcal{Q}_3^*$ — see Phase 1. Since there are three of them, we assume without loss of generality that $(u, v) \in E(\mathcal{Q}_2)$ and the vertex u lies on a 3-path $P_2 \in \mathcal{Q}_3^*$ distinct from P_1, that is, $P_2 \neq P_1$, and furthermore (u, w) is an edge on the 3-path P_2 (see Fig. 8 for an illustration). We can verify the following claim that the vertex w is on a 3-path of $\mathcal{Q}_3$, and consequently $\tau_1(P_2) \le 1$.

Claim 1 *The vertex w is on a 3-path of $\mathcal{Q}_3$.*

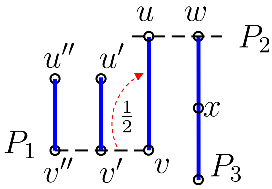

Fig. 8 An illustration of a 3-path $P_1 = v\text{-}v'\text{-}v'' \in \mathcal{Q}_3^*$ with $\tau_1(P_1) = 3/2$, where $u\text{-}v$, $u'\text{-}v'$, $u''\text{-}v'' \in \mathcal{Q}_2$ shown in blue solid edges; the vertex u is on a distinct 3-path $P_2 \in \mathcal{Q}_3^*$ and the edge (u, w) is on P_2 shown black dashed. Then, w must lie on a 3-path $P_3 \in \mathcal{Q}_3$. The red dotted arrow indicates how tokens are redistributed out of P_1 (Color figure online)

Proof See Fig. 8 for an illustration, where we denote the 2-paths of $\mathcal{Q}_2$ incident at v' and v'' as $u'\text{-}v'$ and $u''\text{-}v''$, respectively. We remark that u' and v'' (and likewise u'' and v', respectively) are not necessarily distinct.

Firstly, w cannot collide into any of u', u'' since otherwise the three 2-paths $u\text{-}v$, $u'\text{-}v'$, $u''\text{-}v''$ form a candidate collection for an Operation 3- 0- By- 0- 2. Secondly, suppose w is on a 2-path $w\text{-}x$ of $\mathcal{Q}_2$, then the three 2-paths $u\text{-}v$, $u'\text{-}v'$, $w\text{-}x$ also form a candidate collection for an Operation 3- 0- By- 0- 2. These contradictions show that w does not lie on any 2-path of $\mathcal{Q}_2$.

Lastly, suppose w is a singleton of $\mathcal{Q}_1$, then w and the 2-path $u\text{-}v$ can be merged to a 3-path, contradicting to the fact that $\mathcal{Q}$ is a partition with the least 1-paths. This proves the claim. $\square$

We summarize the above into the following lemma.

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* $\tau_1(P_2) \le 1$ *such that*

(1) *$u\text{-}v$ 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 is on a 3-path P_3 of $\mathcal{Q}_3$.*

From Lemma 2 and seeing Fig. 8 for an illustration, the first step of Phase 2 is to transfer the extra 1/2 token held by P_1 from P_1 to the 2-path $u\text{-}v$ through vertex v. This way, we have $\tau_2(P_1) = -1/2$ and $\tau(P_1) = 3/2 - 1/2 = 1$.

Let us continue using the notations in Lemma 2, and let x_1 and y_1 be the other two vertices on P_3 (*i.e.*, $P_3 = w\text{-}x_1\text{-}y_1$ or $P_3 = x_1\text{-}w\text{-}y_1$). Denote the path in $\mathcal{Q}^*$ on which x_1 (y_1, respectively) lies as P_4 (P_5, respectively). Next, we will transfer the 1/2 token from $u\text{-}v$ to the paths P_4 or/and P_5 through some *pipes*, to be defined below.

We define a *pipe* $r \to s \to t$, where r is an endpoint of a *source* 2-path of $\mathcal{Q}_2$ (the path $u\text{-}v$ in the current consideration) 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 $\tau_1(P) \le 1$ (the path P_2 in the current consideration), s and t are both on a 3-path of $\mathcal{Q}_3$ (the path P_3 in the current consideration), and t is a vertex on the *destination* path of $\mathcal{Q}^*$ (the path P_4 or P_5 in the current consideration) which will receive token from the source 2-path of $\mathcal{Q}_2$. That is, the pipe $r \to s \to t$ will transfer some token from the source 2-path of $\mathcal{Q}_2$ on which r lies, to the destination path of $\mathcal{Q}^*$ on which t lies; and we call r and t the *head* and the *tail* of the pipe, respectively.

For example, in the configuration shown in Fig. 9a, 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 by the orders of the two destination paths P_4 and P_5, to determine how they receive token from source 2-paths through pipes.

Note that u can be either an endpoint or the midpoint of P_2. Nevertheless, when u is the midpoint of P_2, there are at most two possible pipes passing through w; consequently, it can be treated as a subcase of the more general case where u is an endpoint of P_2. Below we consider the more general case, and assume that $P_2 = u$-w-u''; then there are two possible pipes headed by u and two possible pipes headed by u'' passing through w. The second step of Phase 2 is to transfer the $1/2$ token held by the 2-path u-v to the paths P_4 or/and P_5, separated into the following three cases:

Case 1. One of P_4 and P_5 is a singleton.
We assume without loss of generality $P_4 = x_1 \in \mathcal{Q}_1^*$ (see Fig. 9 for illustrations). In this case, we transfer the $1/2$ token from u-v to P_4 through the pipe $u \to w \to x_1$.

Case 2. Both P_4 and P_5 are paths of $\mathcal{Q}_2^* \cup \mathcal{Q}_3^*$, and w is an endpoint of $P_3 = w$-x_1-y_1.
In this case, we transfer the $1/2$ token from u-v to P_5 through the pipe $u \to w \to y_1$ (see Fig. 10a for an illustration).

Case 3. Both P_4 and P_5 are paths of $\mathcal{Q}_2^* \cup \mathcal{Q}_3^*$, and w is the midpoint of $P_3 = x_1$-w-y_1.
In this case, we transfer $1/4$ token from u-v to P_4 through the pipe $u \to w \to x_1$ and transfer the other $1/4$ token from u-v to P_5 through the pipe $u \to w \to y_1$ (see Fig. 10b for an illustration).

Claim 2 *The first item of Lemma 1 holds, that is, for any* 1*-path* $P \in \mathcal{Q}_1^*$, $\tau(P) \le 1$.

Proof Note that if the vertex v on the singleton $P \in \mathcal{Q}_1^*$ does not lie on a 3-path of $\mathcal{Q}_3$, then its token is not changed in Phase 2 and thus $\tau(P) = \tau_1(P) \le 1$. In other words, if the token of P is increased during Phase 2, then $\tau_1(P) = 0$ and $\tau_2(P)$ is assigned in Case 1 during the second step of Phase 2. We thus consider the path P_4 in Case 1, and refer to the configurations in Fig. 9.

Firstly, if P_5 is a singleton or a 2-path, then there are at most 2 pipes ending at x_1, which are $u \to w \to x_1$ and $u'' \to w \to x_1$. Therefore, $\tau_2(P_4) \le 2 \times 1/2 = 1$.

Next, we assume that P_5 is a 3-path so that there could be two more pipes passing through y_1 to x_1 (that is, y_1 takes up the role of w); let (y_1, y_2) be an edge on P_5, then $y_2 \to y_1 \to x_1$ is one of the two possible pipes if y_2 has the same role as u.

When w is the midpoint of $P_3 = x_1$-w-y_1 (see Fig. 9a for an illustration), y_2 cannot have the same role as u since y_2 cannot be on a 2-path of $\mathcal{Q}_2$; or otherwise $\{u'$-v', u-v, P', $P_3\}$ would be a candidate collection for an Operation 3- 1- By- 0- 3, where P' is 2-path of $\mathcal{Q}_2$ on which y_2 lies. Therefore, we again have $\tau_2(P_4) \le 2 \times 1/2 = 1$.

When w is an endpoint of P_3, *i.e.*, either $P_3 = w$-x_1-y_1 (see Fig. 9b for an illustration) or $P_3 = w$-y_1-x_1 (see Fig. 9c for an illustration), u'' cannot have the same role as u; or otherwise $\{u'$-v', u-v, P', P'', $P_3\}$ would be a candidate collection for an Operation 4- 1- By- 1- 3, where P' and P'' are the two 2-paths of $\mathcal{Q}_2$ associated with u'' (just like the two 2-paths u'-v' and u-v associated with u). That is, $u'' \to w \to x_1$ is not a pipe. Additionally, for the same reason as in the last paragraph, y_2 cannot have the same role as u when either $P_3 = w$-x_1-y_1 or $P_3 = w$-y_1-x_1. Therefore, we have $\tau_2(P_4) \le 1/2$.

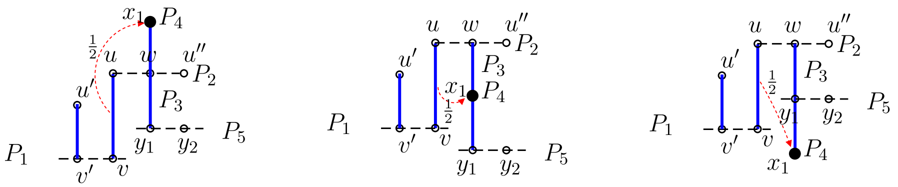

Fig. 9 The configurations in which $P_4 = x_1$ is a singleton of $\mathcal{Q}_1^*$, where blue solid edges are in $E(\mathcal{Q})$ and black dashed edges are in $E(\mathcal{Q}^*)$. The vertex x_1 is the tail of a pipe through which P_4 receives 1/2 token from the source 2-path u-v, indicated by red dotted arrows (Color figure online)

In summary, we have showed that $\tau_2(P_4) \le 1$; it follows from $\tau_1(P_4) = 0$ that $\tau(P_4) \le 1$. □

Claim 3 *The second item of Lemma 1 holds, that is, for any 2-path $P \in \mathcal{Q}_2^*$, $\tau(P) \le 2$.*

Proof Note that for a 2-path $P = u$-$v \in \mathcal{Q}_2^*$, if the vertex v does not lie on a 3-path of $\mathcal{Q}_3$, then its token received through v is not changed in Phase 2, and we denote this portion of token as $\tau(P(v)) = \tau_1(P(v)) \le 1$. In other words, if $\tau_2(P(v)) > 0$, then $\tau_1(P(v)) = 0$ and $\tau_2(P(v))$ is assigned in Cases 2 and 3 during the second step of Phase 2. We thus consider the path P_5 in these two cases, and refer to the configurations in Fig. 10.

In Case 2 where w is an endpoint of $P_3 = w$-x_1-y_1 (see Fig. 10a), u'' cannot have the same role as u; or otherwise $\{u'$-v', u-v, P', P'', $P_3\}$ would be a candidate collection for an Operation 4-1-By-1-3, where P' and P'' are the two 2-paths of $\mathcal{Q}_2$ associated with u'' (just like the two 2-paths u'-v' and u-v associated with u). That is, $u'' \to w \to y_1$ is not a pipe. Assume (x_1, x_2) is an edge on the path P_4; for the same reason x_2 cannot have the same role as u, since otherwise $\{u$-v, P', P'', $P_3\}$ would be a candidate collection for an Operation 3-1-By-0-3, where P' and P'' are the two 2-paths of $\mathcal{Q}_2$ associated with x_2 (just like the two 2-paths u'-v' and u-v associated with u). That is, $x_2 \to x_1 \to y_1$ is not a pipe either. Therefore, we have $\tau_2(P_5(y_1)) \le 1/2$.

In Case 3 where w is the midpoint of $P_3 = x_1$-w-y_1 (see Fig. 10b), we assume (x_1, x_2) is an edge on the path P_4; for the same reason as in the last paragraph x_2 cannot have the same role as u. That is, $x_2 \to x_1 \to y_1$ is not a pipe. Therefore, there are at most two pipes ending with y_1 and consequently $\tau_2(P_5(y_1)) \le 2 \times 1/4 = 1/2$.

In summary, we have showed that the token received through the vertex y_1 is at most $\tau_2(P_5(y_1)) \le 1/2$; it follows from $\tau_1(P_5(y_1)) = 0$ that $\tau(P_5(y_1)) \le 1/2$. One can apply exactly the same argument on the other vertex y_2 and show that if $\tau_2(P_5(y_2)) > 0$ then $\tau(P_5(y_2)) \le 1/2$. These together prove that for every $P \in \mathcal{Q}_2^*$, $\tau(P) \le \max\{2, 1 + 1/2, 1/2 + 1/2\} = 2$. □

Claim 4 *The third item of Lemma 1 holds, that is, for any 3-path $P \in \mathcal{Q}_3^*$, $\tau(P) \le 1$.*

Proof Recall that for every 3-path $P \in \mathcal{Q}_3^*$, P receives 1/2 token through its vertex v in Phase 1 if and only if v lies on a 2-path v-$v' \in \mathcal{Q}_2$ and v' is also on a 3-path of $\mathcal{Q}_3^*$. We denote this portion of token as $\tau_1(P(v))$.

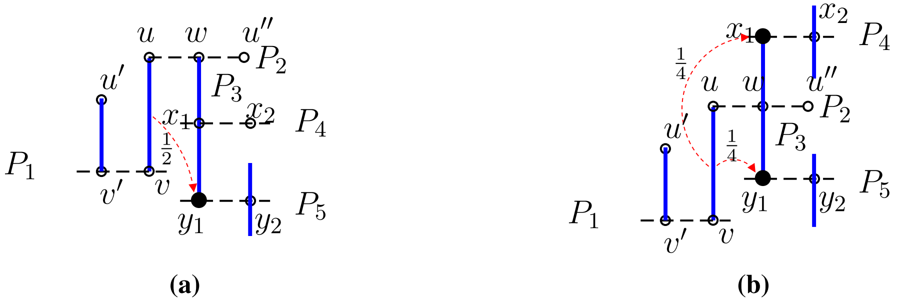

Fig. 10 The configurations in which both P_4 and P_5 are in $\mathcal{Q}_2^* \cup \mathcal{Q}_3^*$, where blue solid edges are in $E(\mathcal{Q})$ and black dashed edges are in $E(\mathcal{Q}^*)$. In (**a**), y_1 is the tail of a pipe through which P_5 receives $1/2$ token from the source 2-path u-v; in (**b**), x_1 is the tail of a pipe through which P_4 receives $1/4$ token from the source 2-path u-v and y_1 is the tail of a pipe through which P_5 receives $1/4$ token from the source 2-path u-v. The red dotted arrows indicate how tokens are moved (Color figure online)

Furthermore, if $\tau_1(P) = 3/2$ then $\tau_2(P) = -1/2$ in the first step of Phase 2 and P never receives any token in the second step of Phase 2; therefore, $\tau(P) = 1$. If $\tau_1(P) \le 1$ and $\tau_2(P) > 0$, then P receives token through pipes with their tails on P and a 3-path of $\mathcal{Q}_3$, and $\tau_2(P)$ is assigned in Cases 2 and 3 during the second step of Phase 2. We thus consider the paths P_4 and P_5 in these two cases, and refer to the configurations in Fig. 10.

In Case 2 where w is an endpoint of $P_3 = w$-x_1-y_1 (see Fig. 10a), the same as in the proof of Claim 3, u'' cannot have the same role as u; that is, $u'' \to w \to y_1$ is not a pipe. Assume (x_1, x_2) is an edge on the path P_4; for the same reason x_2 cannot have the same role as u, that is, $x_2 \to x_1 \to y_1$ is not a pipe either. Therefore, we have $\tau_2(P_5(y_1)) \le 1/2$.

Additionally, one sees that y_2 cannot lie on a 2-path $P_3' \in \mathcal{Q}_2$, since otherwise $\{u'$-v', u-$v, P_3', P_3\}$ would be a candidate collection for an Operation 3- 1- By- 0- 3. Therefore, $\tau_1(P_5(y_2)) = 0$. Note that y_2 cannot be a singleton in $\mathcal{Q}$ either, due to the fact that $\mathcal{Q}$ has the least singletons; we conclude that y_2 lies on a 3-path $P_3' \in \mathcal{Q}_3$. If y_2 is the tail of a pipe, say $z_1 \to w' \to y_2$, that is, both y_2 and w' are on P_3' and z_1 has the same role as u (see Fig. 11 for an illustration), then $\{u'$-v', u-$v, P', P'', P_3, P_3'\}$ would be a candidate collection for an Operation 4- 2- By- 1- 4, where P' and P'' are the two 2-paths of $\mathcal{Q}_2$ associated with z_1 (just like the two 2-paths u'-v' and u-v associated with u). In Fig. 11, P' and P'' are shown as z_1-z_2 and z_3-z_4. This proves that y_2 cannot be the tail of any pipe, and thus $\tau_2(P_5(y_2)) = 0$ too.

In summary, in Case 2 we have $\tau(P_5(y_1)) \le 1/2$ and $\tau(P_5(y_2)) = 0$.

In Case 3 where w is the midpoint of $P_3 = x_1$-w-y_1 (see Fig. 10b), we assume (x_1, x_2) is an edge on the path P_4; the same as in the proof of Claim 3, x_2 cannot have the same role as u. That is, $x_2 \to x_1 \to y_1$ is not a pipe. Therefore, there are at most two pipes ending with y_1 and consequently $\tau_2(P_5(y_1)) \le 2 \times 1/4 = 1/2$.

Additionally and the same as in the above argument for Case 2, one sees that y_2 cannot lie on a 2-path or a singleton of $\mathcal{Q}$; therefore, y_2 lies on a 3-path $P_3' \in \mathcal{Q}_3$ and $\tau_1(P_5(y_2)) = 0$. The exactly same argument for Case 2 above proves that y_2 cannot be the tail of any pipe, and thus $\tau_2(P_5(y_2)) = 0$ too.

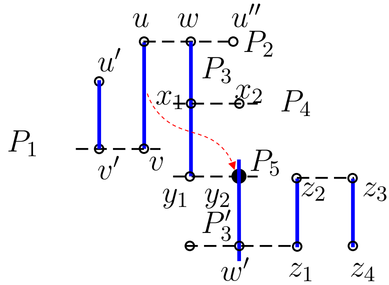

Fig. 11 A configuration where y_2 in Fig. 10a is the tail of a pipe, say $z_1 \to w' \to y_2$ (no matter y_2 and w' are adjacent or not on P_3'), which could never happen due to Operation 4- 2- By- 1- 4. The blue solid edges are in $E(\mathcal{Q})$ and the black dashed edges are outside of $E(\mathcal{Q})$ (Color figure online)

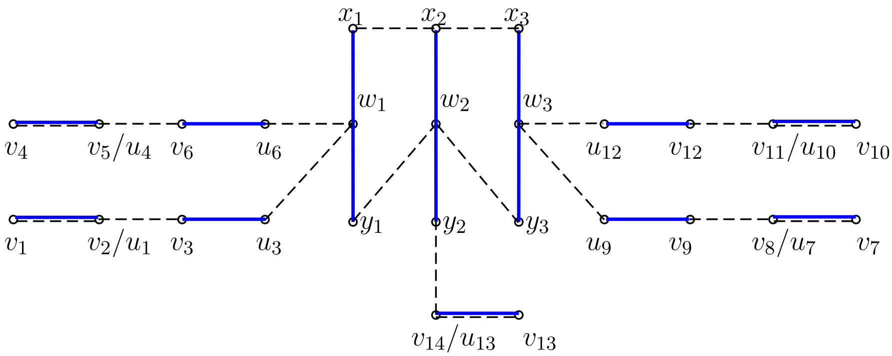

Fig. 12 A tight instance of 27 vertices, where blue solid edges represent a 3-path partition $\mathcal{Q}$ produced by Approx and black dashed edges represent an optimal 3-path partition $\mathcal{Q}^*$. The edges (u_{3i+1}, v_{3i+1}), $i = 0, 1, \ldots, 4$, are in $E(\mathcal{Q}_2) \cap E(\mathcal{Q}^*)$, shown both blue solid and black dashed. The vertex u_{3i+1} collides into v_{3i+2} (denoted as v_{3i+2}/u_{3i+1}), $i = 0, 1, \ldots, 4$. Applying our token distribution scheme, each of the nine 3-paths in $\mathcal{Q}^*$ receives exactly 1 token (Color figure online)

In summary, in Case 3 we have $\tau(P_5(y_1)) \le 1/2$ and $\tau(P_5(y_2)) = 0$.

From the above, we conclude that the token received through the vertex y_1 is at most $\tau(P_5(y_1)) \le 1/2$ and the token received through the vertex y_2 is $\tau(P_5(y_2)) = 0$. For the other vertex y_3 on P_5, if $\tau_1(P_5(y_3)) = 1/2$, then $\tau_2(P_5(y_3)) = 0$; otherwise $\tau_1(P_5(y_3)) = 0$, and the above argument applies again to show that $\tau_2(P_5(y_3)) \le 1/2$. Together, we have $\tau(P_5) = \tau(P_5(y_1)) + \tau(P_5(y_2)) + \tau(P_5(y_3)) \le 2 \times 1/2 = 1$. This proves the claim. □

From Claims 2, 3 and 4, Lemma 1 holds.

3.3 A tight instance for the algorithm Approx

Figure 12 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). When applying our token distribution scheme, each 3-path of $\mathcal{Q}^*$ receives exactly 1 token from the 2-paths in $\mathcal{Q}$. This instance shows that the performance ratio of 4/3 is tight for Approx, thus Theorem 2 is proved.

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 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 showed 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 of them 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 (during the preparation of this journal version, Weitian led the group to design a completely new and improved $21/16$-approximation algorithm for 3PP (Chen et al. 2019c)).

Acknowledgements The authors are grateful to the reviewers for their comments and suggestions to improve the presentation.

Funding YC and AZ are supported by the NSFC Grants 11771114, 11971139 and the Zhejiang Provincial NSFC Grant LY21A010014; YC and AZ are also supported by the CSC Grants 201508330054 and 201908330090, respectively. RG and GL are 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. BS is supported by the National Social Science Foundation of China (Grant No. 20XGL023) and Science and Technology Department of Shaanxi Province (Grant No. 2020JQ-654).

Data availability Not applicable.

Declarations

Conflict of interest The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

Chen Y, Goebel R, Lin G, Liu L, Su B, Tong W, Xu Y, Zhang A (2019a) A local search 4/3-approximation algorithm for the minimum 3-path partition problem. In: Proceedings of FAW 2019, LNCS, vol 11458, pp 14–25

Chen Y, Goebel R, Lin G, Su B, Xu Y, Zhang A (2019b) An improved approximation algorithm for the minimum 3-path partition problem. J Comb Optim 38:150–164

Chen Y, Goebel R, Su B, Tong W, Xu Y, Zhang A (2019c) A 21/16-approximation for the minimum 3-path partition problem. In: Proceedings of ISAAC 2019, LIPIcs, vol 149, pp 46:1–46:20

Duh R, Fürer M (1997) Approximation of k-set cover by semi-local optimization. In: Proceedings of the twenty-ninth annual ACM symposium on theory of computing, STOC'97, pp 256–264

Feige U (1998) A threshold of for approximating set cover. J ACM 45:634–652

Franzblau DS, Raychaudhuri A (2002) Optimal Hamiltonian completions and path covers for trees, and a reduction to maximum flow. ANZIAM J 44:193–204

Garey MR, Johnson DS (1979) Computers and intractability: a guide to the theory of NP-completeness. W. H. Freeman and Company, San Francisco

Goldberg AV, Karzanov AV (2004) Maximum skew-symmetric flows and matchings. Math Program 100:537–568

Johnson DS (1974) Approximation algorithms for combinatorial problems. J Comput Syst Sci 9:256–278

Karp RM (1972) Reducibility among combinatorial problems. In: Proceedings of a symposium on the complexity of computer computations. Springer, Boston

Levin A (2006) Approximating the unweighted k-set cover problem: greedy meets local search. In: Proceedings of the 4th international workshop on approximation and online algorithms (WAOA 2006), LNCS, vol 4368, pp 290–301

Monnot J, Toulouse S (2007) The path partition problem and related problems in bipartite graphs. Oper Res Lett 35:677–684

Raz R, Safra S (1997) 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'97), pp 475–484

Vazirani V (2003) Approximation algorithms. Springer, Berlin

Yan J-H, Chang GJ, Hedetniemi SM, Hedetniemi ST (1997) k-path partitions in trees. Discret Appl Math 78:227–233

Authors and Affiliations

Yong Chen[1] · Randy Goebel[2] · Guohui Lin[2] · Longcheng Liu[3] · Bing Su[4] · Weitian Tong[5] · Yao Xu[5] · An Zhang[1]

Yong Chen
chenyong@hdu.edu.cn

Randy Goebel
rgoebel@ualberta.ca

Longcheng Liu
longchengliu@xmu.edu.cn

Weitian Tong
wtong@georgiasouthern.edu

Yao Xu
yxu@georgiasouthern.edu

An Zhang
anzhang@hdu.edu.cn

1 Department of Mathematics, Hangzhou Dianzi University, Hangzhou, China

2 Department of Computing Science, University of Alberta, Edmonton, AB T6G 2E8, Canada

3 School of Mathematical Sciences, Xiamen University, Xiamen, China

4 School of Economics and Management, Xi'an Technological University, Xi'an, China

5 Department of Computer Science, Georgia Southern University, Statesboro, USA