An Approximate Algorithm for the Weighted Hamiltonian Path Completion Problem on a Tree

Q. S. Wu[1], C. L. Lu[2], and R. C. T. Lee[3]

[1] Dept. of Computer Science, National Tsing Hua University, Hsinchu, Taiwan 300, R.O.C. Email: solomon@cs.nthu.edu.tw

[2] National Center for High-Performance Computing, P.O. Box 19-136, Hsinchu, Taiwan 300, R.O.C. Email: cllu@nchc.gov.tw

[3] National Chi-Nan University, Puli, Nantou Hsien, Taiwan 545, R.O.C. Email: rctlee@ncnu.edu.tw

Abstract. Given a graph, the Hamiltonian path completion problem is to find an augmenting edge set such that the augmented graph has a Hamiltonian path. In this paper, we show that the Hamiltonian path completion problem will unlikely have any constant ratio approximation algorithm unless NP = P. This problem remains hard to approximate even when the given subgraph is a tree. Moreover, if the edge weights are restricted to be either 1 or 2, the Hamiltonian path completion problem on a tree is still NP-hard. Then it is shown that this problem will unlikely have any fully polynomial-time approximation scheme (FPTAS) unless NP=P. When the given tree is a k-tree, we give an approximation algorithm with performance ratio 1.5.

1 Introduction

Given a graph G, a Hamiltonian path is a simple path on G which traverses each vertex exactly once. Finding a Hamiltonian path is often required in problems involving routing and the periodic updating of data structures. In the past, the Hamiltonian path completion problem was defined on unweighted graphs. For a given unweighted graph $G = (V, E_0)$, the Hamiltonian path completion problem is to find an augmenting edge set E_2 such that $G' = (V, E_0 \cup E_2)$ has a Hamiltonian path. Such an edge set E_2 is called an *augment*. It was shown that to find an edge set E_2 with minimum cardinality is an NP-complete problem [9]. If the given graph G is a tree [10, 12, 14], a forest [18], an interval graph [1], a circular-arc graph [7], a bipartite permutation graph [19] or a block graph [19, 20, 22], it was shown that there exist polynomial-time algorithms for this problem by computing the *path covering number*, which is the minimum number of vertex-disjoint paths covering all vertices. In general, for a complete bipartite graph $K_{1,n}$, the path covering number is $n-1$, and at least $n-2$ edges must be added to make it have a Hamiltonian path. In other words, for $K_{1,n}$ to have a Hamiltonian path, any optimal augment must have exactly $n-2$ edges [6]. For

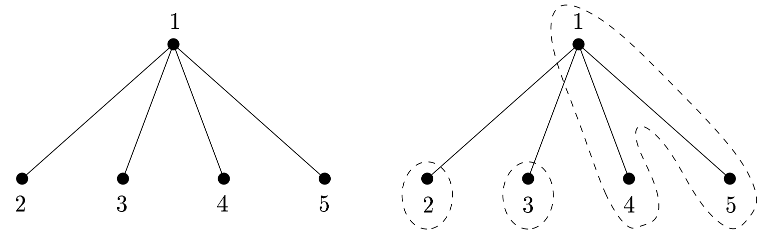


Fig. 1. A bipartite graph $K_{1,4}$ which can be covered by 3 paths.

example, in Figure 1, the graph can be covered by three paths $\{(4,1,5),(2),(3)\}$, and two edges, say (2,3) and (3,4), may be added to make the original graph have a Hamiltonian path.

In this paper, we shall discuss the weighted Hamiltonian path completion problem, whose formal definition is given below.

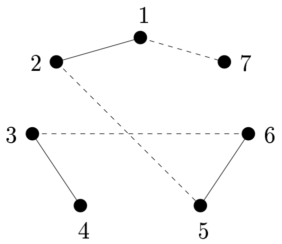
Problem 1 (Weighted Hamiltonian Path Completion Problem). Given a complete graph $G=(V,E)$ with edge costs $w: E \to R^+$, and an edge subset $E_0 \subseteq E$, find an augment $E_2 \subseteq E$ such that $G'=(V, E_0 \cup E_2)$ has a Hamiltonian path and $\sum_{e \in E_2} w(e)$ is minimized.

For example, in Figure 2, given $E_0 = \{(1,2),(3,4),(5,6)\}$ and the weight of each edge assigned as in the table aside, the optimal augment is $E_2 = \{(1,7),(2,5),(3,6)\}$ with cost 10.

However, this problem is hard to solve if E_0 is arbitrary. Throughout this paper, we shall assume that E_0 constitutes a tree. The formal definition of the problem is given as below.

Problem 2 (Weighted Hamiltonian Path Completion Problem on a Tree). Given a complete graph $G=(V,E)$ with edge costs $w: E \to R^+$, and an edge subset E_0 constituting a spanning tree on G, find an augment $E_2 \subseteq E$ such that $G'=(V, E_0 \cup E_2)$ has a Hamiltonian path and the cost $\sum_{e \in E_2} w(e)$ is minimized.

In this paper, we shall first show that the weighted Hamiltonian path completion problem on a tree (HPCT) will unlikely have any constant ratio ap-



	1	2	3	4	5	6	7
1	∞	0	10	10	10	10	2
2		∞	10	10	3	10	10
3			∞	0	10	5	10
4				∞	10	10	10
5					∞	0	10
6						∞	10
7							∞

Fig. 2. An example for the weighted Hamiltonian path completion problem, where the given edge set E_0 is represented by solid lines and augment E_2 by dashed lines.

proximation algorithm and it is still NP-hard even when the edge weights are restricted to be either 1 or 2. We then prove that this problem has no fully polynomial-time approximation scheme (FPTAS) unless NP=P. Furthermore, when the given tree has only one internal node, we give an approximation algorithm with performance ratio $\frac{3}{2}$ and then extend this algorithm to trees with k internal nodes.

2 Non-Approximability of HPCT

To prove that HPCT is hard to approximate, we adopt a technique which is similar to the one applied to prove that the traveling salesperson problem will unlikely have any α-approximate algorithm as in [17]. In our proof, we shall reduce the Hamiltonian path problem, which is a well-known NP-complete problem [9], to the weighted Hamiltonian path completion problem. Let us define the problem first.

Problem 3 (Hamiltonian Path Problem). Given $G = (V, E)$, where $|V| = n$, determine whether there exists a path $(v_1, v_2, v_3, \cdots, v_n)$ such that (v_i, v_{i+1}) in E, for all $i = 1, 2, \cdots, n-1$.

An approximation algorithm $\mathcal{A}$ is said to be α-*approximate* if for any problem instance, the cost of the approximate solution obtained by $\mathcal{A}$ is bounded by α times the cost of the optimal solution. We then have the following theorem.

Theorem 1. *For any $\alpha > 1$, if there exists an α-approximate algorithm for HPCT, then NP = P.*

Proof. Suppose there exists an approximation algorithm $\mathcal{A}$ of HPCT that will find an approximation solution of ratio α in polynomial time. We then reduce HP (Hamiltonian Path problem) to HPCT as follows. Given an instance of HP, say $G = (V, E)$ with $V = \{v_1, v_2, \cdots, v_n\}$, we construct an instance of HPCT, say $G' = (V', E')$ and E_0, as follows.

- $V' = V \cup \{v_0, v_{n+1}\}$.
- $E' = \{(v_i, v_j) \mid 0 \leq i \leq n+1, 0 \leq j \leq n+1 \text{ and } i \neq j\}$.
- $E_0 = \{(v_0, v_i) \mid 1 \leq i \leq n+1\}$.
- For each $e \in E'$, $w(e) = \begin{cases} 1, & \text{if } e \in E, \\ \alpha|E|(n-1), & \text{otherwise.} \end{cases}$

Let us see Figure 3 for an example of the reduction.

First, we claim that G has a Hamiltonian path if and only if G' has an optimal augment with cost $n-1$. It is easy to see that if G has a Hamiltonian path, then by the construction of G', G' has an augment with $n-1$ edges, and each of these edges has weight 1. Conversely, according to the discussion in the previous section, since the given E_0 constitutes a tree $K_{1,n+1}$, any optimal augment for G' must have exactly $n-1$ edges, and only edges with weight 1 corresponds to an edge in G. Therefore, G has a Hamiltonian path if and only if G' has an optimal augment with cost $n-1$.

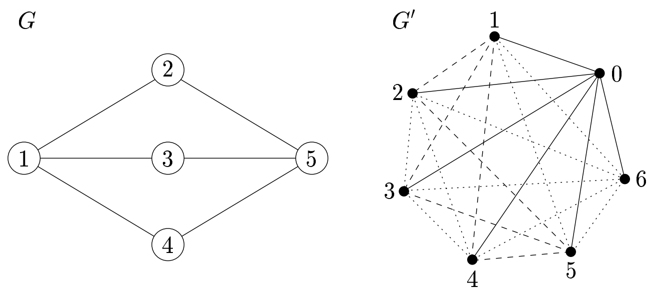


Fig. 3. Reduction from G in the Hamiltonian path problem to G' in HPCT. In G', the dashed edges have weight 1 and other edges have weight 24α.

Second, we claim that if there is an optimal augment for G' with cost $n-1$, $\mathcal{A}$ cannot generate a solution containing any edge with weight $\alpha|E|(n-1)$. If the approximate solution contains any such edge, the cost of the augment would be at least $\alpha|E|(n-1)$. Since $\frac{\alpha|E|(n-1)}{n-1} = \alpha|E| > \alpha$, this violates the assumption that $\mathcal{A}$ is an α-approximate algorithm.

Thus, if G has a Hamiltonian path, then the optimal augment for G' has cost $n-1$, and thus $\mathcal{A}$ will generate an augment with cost less than or equal to $|E|$, since it will only contain edges with weight 1. Conversely, if $\mathcal{A}$ generates an augment with cost less than or equal to $|E|$, then we know this augment does not contain any edge with cost $\alpha|E|(n-1)$. In other words, all the edges in this augment are contained in E. Since such a subset in E constitutes a Hamiltonian path, it lead to the conclusion that the graph G has a Hamiltonian path.

By the above discussion, G has a Hamiltonian path if and only if $\mathcal{A}$ generates an augment with cost less than $|E|$ for G'. Hence we can use $\mathcal{A}$ to solve the Hamiltonian path problem in polynomial time, by examining the cost of the solution returned by $\mathcal{A}$. Thus, if HPCT has an α-approximate algorithm, then NP=P. □

3 NP-Hardness of (1,2)-HPCT

In this section, we shall discuss the (1,2)-HPCT problem, which is the HPCT problem whose edge weights are restricted to be either 1 or 2.

Problem 4 ((1,2)-Hamiltonian Path Completion Problem on a Tree). Given a complete graph $G=(V,E)$ with edge costs $w: E \rightarrow \{1,2\}$, and an edge subset E_0 constituting a spanning tree on G, find an augment $E_2 \subseteq E$ such that $G' = (V, E_0 \cup E_2)$ has a Hamiltonian path and the cost $\sum_{e \in E_2} w(e)$ is minimized.

Theorem 2. *The (1,2)-Hamiltonian path completion problem on a tree is NP-hard.*

Proof. To show that the (1,2)-Hamiltonian path completion problem on a tree ((1,2)-HPCT) is NP-hard, we shall reduce the Hamiltonian path problem (HP) to it. The reduction from HP to (1,2)-HPCT is described as follows.

For $G = (V, E)$ with $V = \{v_1, v_2, \cdots, v_n\}$ in HP, we construct a corresponding instance $G' = (V', E')$ and E_0 in (1,2)-HPCT, such that

- $V' = V \cup \{v_0, v_{n+1}\}$.
- $E' = \{(v_i, v_j) \mid 0 \le i \le n+1, 0 \le j \le n+1 \text{ and } i \ne j\}$.
- For each $e \in E', w(e) = \begin{cases} 1, & \text{if } e \in E, \\ 2, & \text{otherwise.} \end{cases}$
- $E_0 = \{(v_0, v_i) \mid 1 \le i \le n+1\}$.

Now we are going to show that G has a Hamiltonian path if and only if G' has an augment of cost $n-1$. Let $(v_1, v_2, \cdots, v_n)$ be a Hamiltonian path in G. According to our construction, $\{(v_1, v_2), (v_2, v_3), \cdots, (v_{n-1}, v_n)\}$ forms an augment for E_0 in G', which yields a Hamiltonian path $(v_{n+1}, v_0, v_1, v_2, \cdots, v_n)$. For all $i = 1, 2, \cdots, n-1$, we have $w((v_i, v_{i+1})) = 1$ in G' since $(v_i, v_{i+1}) \in E$. Therefore, the cost of this augment is $n-1$.

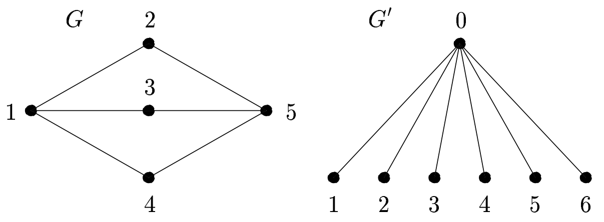
On the contrary, suppose E_0 has an augment E_2 of cost $n-1$. Since E_0 consists of a root vertex and $n+1$ leaves, at least $n-1$ edges must be added to make it Hamiltonian. Therefore, each of these $n-1$ edges must have weight exactly equal to 1 and $E_2 \subseteq E$.

Let $P \subseteq E_0 \cup E_2$ be a Hamiltonian path in $G'' = (V', E_0 \cup E_2)$. Since V' has $n+2$ vertices, $|P| = n+1$. Because the path is Hamiltonian, the vertex v_{n+1} must be covered by the path. Since $w((v_i, v_{n+1})) = 2$ for all $1 \le i \le n$, P must pass through v_0 to visit v_{n+1} and then stop. In other words, if we delete v_0 and v_{n+1} from P, we can obtain a path P' which visits all vertices in $\{v_1, v_2, \cdots, v_n\}$. Such a path $P' \subseteq E_2$ is a Hamiltonian path for G. Hence G has a Hamiltonian path, since $E_2 \subseteq E$.

Therefore, (1,2)-HPCT has an augment of cost $n-1$ if and only if G has a Hamiltonian path. Figure 4 illustrates such a reduction. □

4 (1,2)-HPCT Will Unlikely Have Any FPTAS

A *polynomial-time approximation scheme* (PTAS) is a family of algorithms, which for any rational value $r > 1$, there is a corresponding approximation



	0	1	2	3	4	5	6
0	∞	-	-	-	-	-	-
1		∞	1	1	1	2	2
2			∞	2	2	1	2
3				∞	2	1	2
4					∞	1	2
5						∞	2
6							∞

Fig. 4. A Hamiltonian path $(2, 1, 3, 5, 4)$ in G corresponds to an augment $\{(2, 1), (1, 3), (3, 5), (5, 4)\}$ with weight $= 4$. (The dash in the above table indicates that the edge is in the given edge set E_0 and will cause no cost to the augment, no matter it is equal to 1 or 2.)

algorithm whose solution is within ratio r, and the time complexity of this approximation algorithm is polynomial in the size of its input. When the running time of a PTAS is polynomial both in the size of the input and in $1/(r-1)$, the scheme is called a *fully polynomial-time approximation scheme* (FPTAS) [3]. Some problems, like maximum independent set problem on planar graphs and Euclidean traveling salesperson problem are found to have PTASs [2, 4], while 0-1 knapsack problem admits an FPTAS [11]. On the contrary, some problems, such as maximum 3-satisfiability problem, maximum leaves spanning tree problem, superstring problem, and traveling salesperson problem with distances one and two, are proved by a reduction from the MAX SNP-complete class that they unlikely have any PTAS unless NP=P [5, 8, 15, 16]. In this section, we shall use a simple method to show that (1,2)-HPCT will unlikely have any FPTAS unless NP=P.

Theorem 3. *If (1,2)-HPCT has an FPTAS, then NP=P.*

Proof. Suppose (1,2)-HPCT has an FPTAS. That is, for any $\epsilon > 0$, there exists a $(1+\epsilon)$-approximate algorithm for the (1,2)-HPCT problem such that its time complexity is polynomial in the size of the input and $\frac{1}{\epsilon}$. Then let us consider any problem instance with $G = (V, E)$.

Since (1,2)-HPCT has an FPTAS, we may choose $\epsilon = \frac{1}{2|E|}$. Then the fact that the performance ratio is less than or equal to $1 + \epsilon$ implies that

$$\frac{w_{apx}}{w_{opt}} \leq 1 + \frac{1}{2|E|} \Rightarrow w_{apx} \leq w_{opt} + \frac{w_{opt}}{2|E|} \Rightarrow w_{apx} - w_{opt} \leq \frac{w_{opt}}{2|E|}.$$

Notice that the optimal augment E^* will not contain all edges in E, i.e., $|E^*| < |E|$. Since the weight on each edge is either 1 or 2, we have

$$\frac{w_{opt}}{2|E|} = \frac{w(E^*)}{2|E|} \leq \frac{2|E^*|}{2|E|} < 1.$$

Therefore, $w_{apx} - w_{opt} < 1$. This inequality and the fact that the weight on each edge is greater than or equal to 1 guarantee that $w_{apx} - w_{opt} = 0$. That is, our approximation algorithm will generate an optimal solution with time complexity which is polynomial in both $|E|$ and $\frac{1}{\epsilon} = 2|E|$ (i.e., polynomial in the size of the input). The (1,2)-HPCT problem, which is shown to be NP-hard in Section 3, can be solved in polynomial time. This implies NP=P. □

5 A 1.5-Approximate Algorithm for (1,2)-Hamiltonian Path Completion Problem on 1-Star

A *1-star* is a tree with only one internal node. Since it is a bipartite graph, we shall use the notation $K_{1,n}$ to denote a 1-star with n leaf nodes. In this section, we shall give a 1.5-approximate algorithm for the (1,2)-Hamiltonian path completion problem on a 1-star.

Suppose $G = (V, E)$, where $V = \{v_0, v_1, v_2, \cdots, v_n\}$ and v_0 is the root of the given tree. Our approximation algorithm will first find a minimum-weight maximal matching [13] on $V' = V \setminus \{v_0\}$, and then add edges to concatenate these matching pairs. For example, in Figure 5, the matching may find $\{(v_1, v_2), (v_3, v_4)\}$, then in the second step (v_4, v_5) is added to form a Hamiltonian path. The formal description of our algorithm is as follows.

Algorithm 1
1. **if** $n \leq 2$, **then** return $E_2 = \emptyset$;
2. Perform a minimum-weight maximal matching algorithm on V';
   Suppose the matching is $\{(v_{i_1}, v_{j_1}), (v_{i_2}, v_{j_2}), \cdots, (v_{i_{\lfloor \frac{n}{2} \rfloor}}, v_{j_{\lfloor \frac{n}{2} \rfloor}})\}$.
3. Concatenate these edges serially by adding $(v_{j_2}, v_{i_3}), \cdots, (v_{j_{\lfloor \frac{n}{2} \rfloor - 1}}, v_{i_{\lfloor \frac{n}{2} \rfloor}})$;
4. Return $E_2 = \left(\bigcup_{k=1}^{\lfloor \frac{n}{2} \rfloor} \{(v_{i_k}, v_{j_k})\}\right) \cup \left(\bigcup_{k=2}^{\lceil \frac{n}{2} \rceil - 1} \{(v_{j_k}, v_{i_{k+1}})\}\right)$;

Lemma 1. *If the optimal augment E_2^* contains k edges with weight 1, then our algorithm finds at least $\frac{k}{2}$ of edges with weight 1 in the solution.*

Proof. It can be seen that for $K_{1,n}$, any optimal augment E_2^* always has exactly $n - 2$ edges. Moreover, in $G' = (V, E_0 \cup E_2^*)$, there exist two edges (v_0, v_s) and (v_0, v_t) in E_0 such that they together with these $n-2$ edges in E_2^* will constitute a Hamiltonian path, as illustrated in Figure 6.

By deleting the vertex v_0 from the path, we obtain two vertex-disjoint paths to cover $V' = \{v_1, v_2, \cdots, v_n\}$. (Please note that a single vertex is also regarded as a degenerated path here.) Connect these two paths with "head to head" and "tail to tail" by adding the two corresponding edges, we obtain a cycle C. Obviously $E_2^* \subset C$. Without lost of generality, let us label this cycle as $\{(v_1, v_2), (v_2, v_3), \cdots, (v_{n-1}, v_n), (v_n, v_1)\}$, where (v_s, v_t) and (v_n, v_1) are the two added edges. Let MM be the minimum-weight maximal matching obtained in our algorithm.

Case 1. If n is even, then each maximal matching has $\frac{n}{2}$ edges since the given graph is complete. It can be seen that $M_1 = \{(v_1, v_2), (v_3, v_4), \cdots, (v_{n-1}, v_n)\}$ is a maximal matching and $M_2 = \{(v_2, v_3), (v_4, v_5), \cdots, (v_n, v_1)\}$ is also a maximal matching. Suppose MM contains k_1 edges with weight 1. Then both M_1 and M_2 contain at most k_1 edges with weight 1; otherwise, MM cannot be minimum. Thus, $C = M_1 \cup M_2$ contains at most $2k_1$ edges with weight 1. If $k_1 < \frac{k}{2}$, then

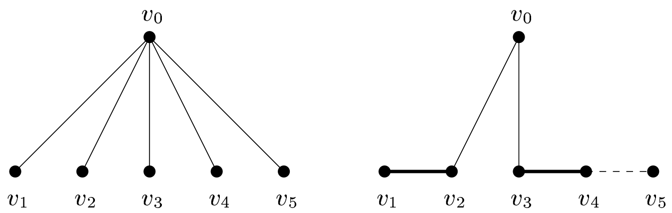


Fig. 5. Finding the minimum-weight maximal matching on a 1-star.

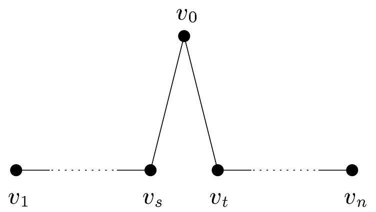


Fig. 6. A Hamiltonian path in $G' = (V, E_0 \cup E_2^*)$.

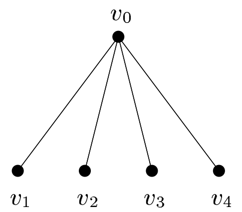


Fig. 7. An example showing that ratio 1.5 is tight.

$2k_1 < k$ and C contains fewer than k edges with weight 1. This contradicts with the assumption that the subset E_2^* of C already has k edges with weight 1.

Case 2. If n is odd, then each maximal matching has $\frac{n-1}{2}$ edges. It can be seen that $M_1 = \{(v_1, v_2), (v_3, v_4), \cdots, (v_{n-2}, v_{n-1})\}$ is a maximal matching and $M_2 = \{(v_2, v_3), (v_4, v_5), \cdots, (v_{n-1}, v_n)\}$ is a maximal matching, too. Suppose MM contains k_1 edges with weight 1. Then with similar reasoning, both M_1 and M_2 will have less than or equal to k_1 edges with weight 1. Again, $k_1 < \frac{k}{2}$ implies that the path $\{(v_1, v_2), (v_2, v_3), (v_3, v_4), \cdots, (v_{n-1}, v_n)\}$ contains fewer than k edges with weight 1. This also causes a contradiction because its subset E_2^* already contains k edges with weight 1.

In either case, our algorithm finds at least $\frac{k}{2}$ edges with weight 1 in the minimum-weight maximal-matching MM. □

Theorem 4. *The performance ratio of Algorithm 1 is* $\frac{3}{2}$.

Proof. The augment E_2 obtained by our algorithm contains $\lfloor \frac{n}{2} \rfloor + \lceil \frac{n}{2} \rceil - 2 = n - 2$ edges, just the same as the optimal augment. Suppose the optimal solution E_2^* contains k edges with weight 1, and h edges with weight 2 (i.e., $k + h = n - 2$). According to Lemma 1, our approximation algorithm which performs minimum-weight maximal-matching to get a partial result will choose at least $\frac{k}{2}$ edges with weight 1. Even in the worst case that all the other edges added later are with weight 2, the performance ratio of our approximate solution will be

$$\frac{\frac{k}{2} + 2(\frac{k}{2} + h)}{k + 2h} = \frac{\frac{3k}{2} + 2h}{k + 2h} \leq \frac{3}{2}.$$

Remarks. Precisely, by Lemma 1, the above formula should be written as

$$\frac{\lceil \frac{k}{2} \rceil + 2(k + h - \lceil \frac{k}{2} \rceil)}{k + 2h}.$$

When k is odd, it becomes

$$\frac{\frac{k+1}{2} + 2(k + h - \frac{k+1}{2})}{k + 2h} = \frac{\frac{3k-1}{2} + 2h}{k + 2h} < \frac{\frac{3k}{2} + 2h}{k + 2h} \leq \frac{3}{2}.$$

□

The time complexity of the Algorithm 1 is analyzed as follows. Let $n+1$ be the number of vertices in G. The minimum-weight maximal matching in Step 2 takes $O(n^3)$ time [13]. Step 3 will take $O(n)$ time to connect the edges one by one. Therefore, we obtain a 1.5-approximate algorithm to solve the (1,2)-Hamiltonian path completion problem on 1-star that runs in $O(n^3)$ time.

To show that the analysis for the performance ratio of this algorithm is tight, let us see the example in Figure 7, where the edge weights are

$$w(e) = \begin{cases} 1, & \text{if } e \in \{(v_1, v_2), (v_1, v_3)\}, \\ 2, & \text{otherwise.} \end{cases}$$

Our approximation algorithm will find a minimum-weight maximal matching first, say $\{(v_1, v_2), (v_3, v_4)\}$ and adopts this as the approximate solution. Its cost will be 3. However, the optimal augment is $\{(v_1, v_2), (v_1, v_3)\}$, whose cost is 2. The performance ratio then is $\frac{3}{2} = 1.5$.

6 A 1.5-Approximate Algorithm for (1,2)-Hamiltonian Path Completion Problem on k-Star

A tree with k internal nodes is called a k-star. In this section, we are going to show that the approximation algorithm developed in the previous section can be extended to obtain similar results on k-stars. Let us introduce some notation first.

Definition 1. *Suppose $G = (V, E)$ is a weighted graph with weight function $w(e)$ defined on all edges e in E and let $P = (u_1, \cdots, u_k)$ be a path in G. Then the new graph $G' = (V', E')$ obtained from G by shrinking P into a single node is define to be*

- $V' = V \setminus \{u_2, u_3, \cdots, u_k\}$.
- $E' = E \setminus \bigcup_{v \in V, u \in \{u_2, u_3, \cdots, u_k\}} \{(u, v)\}$.
- $w'(e) = \begin{cases} \min\{d(u_1, v), d(u_k, v)\}, & \textit{if } e = (u_1, v) \textit{ for some } v \in V', \\ w(e), & \textit{otherwise.} \end{cases}$

Let $G_{\{P\}}$ denote the graph obtained from G by shrinking the path P. For two *vertex-disjoint* paths P_1 and P_2, the result of shrinking will be the same no matter P_1 is shrunk first or P_2 is shrunk first, because the `min` operation satisfies the associative law. That is $(G_{\{P_1\}})_{\{P_2\}} = (G_{\{P_2\}})_{\{P_1\}}$. Therefore, it is sound to simply write it as $G_{\{P_1, P_2\}}$.

Informally speaking, our approximation algorithm will first find a minimum-weight maximal-matching on the shrunk graph $G_{\{P_1, P_2, \cdots, P_i\}}$, and then map these matching edges back to G. By our construction of $G_{\{P_1, P_2, \cdots, P_i\}}$, each edge in $G_{\{P_1, P_2, \cdots, P_i\}}$ corresponds to an edge in G, and each vertex in $G_{\{P_1, P_2, \cdots, P_i\}}$ corresponds to either a single vertex or a path in G. Therefore, each edge in $G_{\{P_1, P_2, \cdots, P_i\}}$ will be mapped to a path (containing one or more edges) in G. These paths are then concatenated serially to form a Hamiltonian path.

To formally analyze the performance ratio of our approximation algorithm, let us try to introduce some lemmas. In G, we call the edges in the given subset E_0 the e_0-*edges*, and edges in $E \setminus E_0$ with weight 1 the e_1-*edges*, and edges in $E \setminus E_0$ with weight 2 the e_2-*edges*.

Observation 1 *Suppose H is a Hamiltonian path in G, E_0 is a k-star and $H \cap E_0$ are vertex-disjoint paths $P_1, P_2, \cdots, P_i$. If we focus on edges in H, it can be observed that during the process of shrinking $P_1, P_2, \cdots, P_i$, only e_0-edges will be deleted, and some e_2-edges may be turned to e_1-edges. Therefore, $H_{\{P_1,P_2,\cdots,P_i\}}$ contains at least n_1 edges with weight 1 if H contains n_1 such edges.*

Lemma 2. *If a Hamiltonian path H in graph G contains n_1 e_1-edges, then a minimum-weight maximal-matching on G contains at least $\frac{n_1}{2}$ e_1-edges.*

Proof. The proof is similar to that of Lemma 1. □

Theorem 5. *If a Hamiltonian path H contains n_1 e_1-edges and $H \cap E_0$ consists of some vertex-disjoint paths $P_1, P_2, \cdots, P_i$, then a minimum-weight maximal-matching on $G_{\{P_1,P_2,\cdots,P_i\}}$ contains at least $\frac{n_1}{2}$ e_1-edges.*

Proof. By Observation 1 and Lemma 2. □

Informally speaking, we shall map these matching edges in $G_{\{P_1,P_2,\cdots,P_i\}}$ to G. For each edge $e = (x, u)$ in $G_{\{P_1,P_2,\cdots,P_i\}}$, let u and v be the terminals of path P_j, for $1 \leq j \leq i$. Then we map the e to e' in G as follows.

$$e' = \begin{cases} (x,u), & \text{if } d(x,u) \text{ on G is equal to } d(x,u) \text{ on } G_{\{P_1,P_2,\cdots,P_i\}}, \\ (x,v), & \text{otherwise.} \end{cases}$$

Therefore, if H contains n_0 e_0-edges, n_1 e_1-edges, n_2 e_2-edges, then this result will also contain the same n_0 e_0-edges, and at least $\frac{n_1}{2}$ edges with weight 1. As we have seen in the previous section, the cost will be less than or equal to $\frac{n_1}{2} + 2 \times (n_1 + n_2 - \frac{n_1}{2}) = \frac{3}{2}n_1 + 2n_2$. Therefore, the performance ratio is $\frac{\frac{3}{2}n_1 + 2n_2}{n_1 + 2n_2} \leq \frac{3}{2}$.

Now we have a natural question: How does our approximation algorithm know what paths in E_0 must be chosen to shrink? If we could choose the ones exactly as an optimal path H^* contains, then the cost of the solution obtained by our approximation algorithm is guaranteed to be within $\frac{3}{2}$ times the one of the optimal solution. However, we do not know what edges may be contained in H^* and what may not. If we have to try each possibility, since there are $O(n)$ e_0-edges, trying all the $O(2^n)$ combinations will lead to an exponential algorithm. Fortunately, we have the following lemma.

Lemma 3. *If H is a Hamiltonian path in G and E_0 is a k-star, then H contains at most $2k$ e_0-edges.*

Proof. All e_0-edges must be incident to the k internal nodes. However, in a path, at most two edges are incident to each internal node, so at most $2k$ e_0-edges are contained in this path. □

Hence, we only have to test all combinations that contain 0 e_0-edges, 1 e_0-edges, 2 e_0-edges, $\cdots$, $2k$ e_0-edges. For each combination, we apply the shrink operation and find an approximate solution on it. The minimum one of these solutions will be chosen as our final approximate solution to be reported. Therefore, let us state our algorithm formally below.

Algorithm 2
1. $W \leftarrow \infty, AUG \leftarrow \emptyset$.
2. **for** all subsets of E_0 with no more than $2k$ edges **do**
2.1. **if** the subset has 3 or more edges incident to the same vertex **then**
/* Do nothing. */
2.2. **else**
Suppose the subset consists of vertex-disjoint paths $P_1, P_2, \cdots, P_i$.
2.2.1. Shrink paths $P_1, P_2, \cdots, P_i$ to obtain $G_{\{P_1,P_2,\cdots,P_i\}}$;
2.2.2. Find a minimum-weight maximal-matching MM on $G_{\{P_1,P_2,\cdots,P_i\}}$;
2.2.3. Map these matching edges to paths in G;
/* Let MM be mapped to MM'. */
2.2.4. Add edges (denoted by S) to concatenate these paths serially to form a Hamiltonian path;
2.2.5. **if** the cost of $S \cup MM'$ is smaller than W **then**
$W \leftarrow w(S \cup MM')$, $AUG \leftarrow S \cup MM'$;
3. Report AUG as the solution and stop;

There are $O(n^{2k})$ iterations in Step 2 of Algorithm 2, and each iteration takes $O(n^3)$ time as we have seen in the previous section. Hence, we obtain a 1.5-approximate algorithm to solve the (1,2)-Hamiltonian path completion problem on k-stars that runs in $O(n^{2k+3})$ time.

7 Conclusion

Some version of Hamiltonian completion problems finds the augment to make the given graph having a Hamiltonian circuit instead of a Hamiltonian path. It can be checked that our results can be applied to obtain the same results on Hamiltonian cycle completion problems.

In [21], an approximation algorithm for (1,2)-HPCT with performance ratio 2 was proposed for general trees. For this problem, whether there exists a PTAS or an approximation algorithm with performance ratio less than 2 as we derived for k-stars is still unknown.

References

1. S. R. Arikati and C. Pandu Rangan. Linear algorithm for optimal path cover problem on interval graphs. *Inf. Process. Lett.*, 35(3):149–153, 1990.
2. S. Arora. Polynomial time approximation schemes for euclidean traveling salesman and other geometric problems. In *Proc. 37th Annual IEEE Symposium on Foundations of Computer Science*, pages 2–11. IEEE Computer Society, 1996.

3. G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi. *Complexity and Approximation.* Springer-Verlag, Berlin Heidelberg, 1999.
4. B. S. Baker. Approximation algorithms for NP-complete problems on planar graphs. *Journal of the ACM*, 41(1):153–180, 1994.
5. A. Blum, M. Li, J. Tromp, and M. Yannakakis. Linear approximation of shortest superstrings. *Journal of the ACM*, 41(4):630–647, 1994.
6. F. T. Boesch, S. Chen, and J. A. M. McHugh. On covering the points of a graph with point disjoint paths. In *Lecture Notes in Mathematics*, volume 46, pages 201–212. Springer, Berlin, 1974.
7. M. A. Bonuccelli and D. P. Bovet. Minimum node disjoint path covering for circular-arc graphs. *Inf. Process. Lett.*, 8:159–161, 1979.
8. G. Galbiati, F. Maffioli, and A. Morzenti. A short note on the approximability of the maximum leaves spanning tree problem. *Inf. Process. Lett.*, 52:45–49, 1994.
9. M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* Freeman, San Francisco, CA, 1979.
10. S. E. Goodman, S. T. Hedetniemi, and P. J. Slater. Advances on the Hamiltonian completion problem. *Journal of the ACM*, 22(3):352–360, 1975.
11. O. H. Ibarra and C. E. Kim. Fast approximation algorithms for the knapsack and sum of subset problems. *Journal of the ACM*, 22(4):463–468, 1975.
12. S. Kundu. A linear algorithm for the Hamiltonian completion number of a tree. *Inf. Process. Lett.*, 5:55–57, 1976.
13. E. L. Lawler. *Combinatorial Optimization: Networks and Matroids*, chapter 6. Holt, Rinehart and Winston, New York, 1976.
14. J. Misra and R. E. Tarjan. Optimal chain partitions of trees. *Inf. Process. Lett.*, 4:24–26, 1975.
15. C. H. Papadimitriou and M. Yannakakis. Optimization, approximation, and complexity classes. *J. Comput. Syst. Sci.*, 43:425–440, 1991.
16. C. H. Papadimitriou and M. Yannakakis. The traveling salesman problem with distances one and two. *Mathematics of Operations Research*, 18:1–11, 1993.
17. S. Sahni and T. Gonzalez. *P*-complete approximation problems. *Journal of the ACM*, 23(3):555–565, 1976.
18. Z. Skupien. Path partitions of vertices and Hamiltonicity of graphs. In *Proc. 2nd Czechoslovakian Symp. on Graph Theory*, pages 481–491, Prague, 1974.
19. R. Srikant, R. Sundaram, K. S. Singh, and C. P. Rangan. Optimal path cover problem on block graphs and bipartite permutation graphs. *Theoretical Computer Science*, 115(2):351–357, 1993.
20. P.-K. Wong. Optimal path cover problem on block graphs. *Theoretical Computer Science*, 225(1–2):163–169, 1999.
21. Q. S. Wu. *On the Complexity and Approximability of Some Hamiltonian Path Problems.* PhD Thesis, National Tsing Hua University, Taiwan, 2000.
22. J. H. Yan and G. J. Chang. The path-partition problem in block graphs. *Inf. Process. Lett.*, 52:317–322, 1994.