Graph Properties that Facilitate Travelling*

Dimitris A. Fotakis[1,2] Paul G. Spirakis[1,2]

[1] Computer Engineering and Informatics Department
University of Patras, 265 00 Rion, Patras, Greece

[2] Computer Technology Institute — CTI
Kolokotroni 3, 262 21 Patras, Greece

Fax: +30-61-993973

Email: fotakis@cti.gr, spirakis@cti.gr

Abstract. In this work, we study two special cases of the metric Travelling Salesman Problem, Graph TSP and TSP(1,2). At first, we show that dense instances of TSP(1,2) and Graph TSP are essentially as hard to approximate as general instances of TSP(1,2).

Next, we present an $\mathcal{NC}$ algorithm for TSP(1,2) that produces a tour of length at most n plus the cardinality of the Maximum Independent Set of the input graph. Thus, it always achieves a ratio of $\frac{3}{2}$, while the best known parallel algorithm achieves the same ratio in $\mathcal{RNC}$. Moreover, a variant of this algorithm achieves in $\mathcal{NC}$ a constant approximation ratio for Maximum Independent Set, if the optimal value of TSP(1,2) is at least $(1+\beta)n$, for some constant $\beta > 0$.

Then, we prove that, given a graph and a partition of the vertices into κ cliques, optimal solutions for both Graph TSP and TSP(1,2) can be found in time $n^{\mathcal{O}(\kappa^2)}$. Additionally, we show that optimal solutions can be computed in time $n^{\mathcal{O}(\kappa)}$ for a certain class of instances, and we conjecture that the time complexity can be improved to $n^{\mathcal{O}(\kappa)}$ for any partition of a graph into κ cliques.

Also, we obtain a $\frac{22}{15}$-approximation algorithm for TSP in graphs of diameter 3, and a $\frac{17}{12}(\frac{7}{5})$ approximation algorithm for TSP in graphs that have diameter 4(3) and contain a perfect, triangle-free 2-matching.

1 Introduction

In the *Travelling Salesman Problem* (TSP), we are given n nodes, and for each pair of distinct nodes $\{i, j\}$ a distance $d_{i,j}$. The objective is to find a tour (i.e. a cycle that visits each node exactly once) of minimum length. TSP is a fundamental problem in combinatorial optimization, and the most important algorithmic ideas have been applied to it during the past decades (cf. [LLRS85]).

Even very special cases of TSP are known to be $\mathcal{NP}$-complete [GJ79]. As for the approximability, it is well known that the general version of TSP is not approximable within any constant factor, unless $\mathcal{P} = \mathcal{NP}$. However, the metric TSP, where the nodes lie in a metric space (i.e. the distances satisfy the triangle inequality) is approximable within $\frac{3}{2}$ [Chr76]. In this work, we study two special cases of metric TSP, *Graph TSP* and *TSP(1,2)*. In both cases, the nodes are the vertices of an unweighted graph. In Graph TSP, the distance metric is the shortest path metric, while in TSP(1,2) the distance between a pair of vertices is either one, if they are connected by an edge, or two, otherwise. Clearly, TSP(1,2) is the simplest special case of metric TSP.

* This work was partially supported by ESPRIT LTR Project no. 20244—ALCOM–IT.

The best known approximation algorithm for general instances of Graph TSP is the $\frac{3}{2}$-algorithm of Christofides [Chr76], and it has been a major open problem for more than two decades if this bound can be improved. Recently, it was conjectured [Goe95] that the Held-Karp lower bound can achieve a ratio of $\frac{4}{3}$. The best known approximation algorithm for TSP(1,2) achieves a ratio of $\frac{7}{6}$ [PY93]. Moreover, TSP(1,2) is MAX–SNP-hard [PY93], thus polynomial time approximation schemes (PTAS) do not exist for general instances, unless $\mathcal{P} = \mathcal{NP}$. Obviously, this result generalizes to metric TSP.

During the very last years, a few remarkable approximation algorithms for special cases of TSP were invented. These algorithms are the PTAS for planar instances of Graph TSP [GKP95], the PTAS for Euclidean TSP (where the nodes lie in $\mathbb{R}^d$, and the distances are defined by the Euclidean norm) [Aro96], and the PTAS for weighted planar graphs [AGKKW98].

Initially, our motivation came from the recent success in approximating dense instances of many combinatorial optimization problems (e.g. Max Cut, Max κ-Sat, Min Linear Arrangement, Bandwidth, Vertex Cover, Steiner Tree, see [Kar97] for a survey). In particular, Karpinski [Kar97] stated the question on the existence of a PTAS for dense instances of TSP(1,2). In this work, after answering this question in the negative for both TSP(1,2) and Graph TSP, we restrict our attention to the approximability of other special classes of instances. We present (approximation and exact) algorithms for several interesting special cases.

1.1 Definitions

The Travelling Salesman Problem with distances one and two (TSP(1,2)) is a special case of TSP restricted to complete graphs, where all edge lengths are either 1 or 2; clearly, this satisfies the triangle inequality. Alternatively, TSP(1,2) is a generalization of the Hamiltonian Cycle problem, where each edge of the input graph has length 1, and each non-edge has length 2. In this case, we seek the tour (a simple spanning cycle) with the fewest possible non-edges.

In Graph TSP, the input is an unweighted graph, and the distance between a pair of vertices is the length of the shortest path from the first vertex to the second one.

A graph $G(V, E)$ is called *dense*, if it is δ-dense. A graph G is *δ-dense*, if the minimum degree of G is at least $\delta|V|$, for some constant $\delta > 0$. By definition, there exists a correspondence between graphs and instances of TSP(1,2) and Graph TSP. An instance of TSP(1,2)/Graph TSP is called δ-dense (or simply dense), if the corresponding graph is δ-dense.

1.2 Summary of Results and Comparison to Previous Work

In this work, we start with an approximation preserving transformation from general instances of TSP(1,2) to dense instances of TSP(1,2) and Graph TSP. Thus, we prove that dense instances of TSP(1,2) and Graph TSP are essentially as hard to approximate as general instances of TSP(1,2). Therefore, we answer the question of [Kar97], by practically excluding the possibility of a PTAS for dense instances of TSP(1,2) and Graph TSP. Very recently, a different proof appears independently in [FVK98]

Then, we provide an $\mathcal{NC}$ algorithm that approximates TSP(1,2) within $\frac{3}{2}$. The best known parallel algorithm for TSP(1,2) is the algorithm of Christofides (an $\mathcal{RNC}$ implementation for metric TSP is described in [DSST97]). However, since the latter algorithm requires the computation of a perfect matching of minimum weight (maximum cardinality matching for TSP(1,2)), it is in $\mathcal{RNC}$, but it is not known to be in $\mathcal{NC}$. Our algorithm is based on the local search paradigm, and its performance on general instances matches the performance of a (sequential) 4-local search algorithm presented in [KMSV94]. The sequential running time of our algorithm is quasi-quadratic, that is significantly better than the running time of the 4-local search of [KMSV94].

Additionally, our $\mathcal{NC}$ algorithm performs remarkably well in instances with Independent Sets of bounded cardinality. Namely, if the value of the Maximum Independent Set (MIS) of the input graph $G(V, E)$ is MIS_G, the algorithm produces a tour of length no more than $|V| + \mathrm{MIS}_G$. Thus, it outperforms all known sequential approximation algorithms for the restriction of TSP(1,2) in graphs G, such that $\mathrm{MIS}_G < \frac{|V|}{6}$. As a side effect, we show that a variant of this algorithm produces in $\mathcal{NC}$ an Independent Set of cardinality at least $\mathrm{TSP}_G - |V|$, where TSP_G is the optimal value of TSP(1,2) in the input graph G. This is a $\frac{2\beta}{1+\beta}$-approximation for MIS, if $\mathrm{TSP}_G \geq (1+\beta)n$, for some constant $\beta > 0$. It is well known that, unless $\mathcal{P} = \mathcal{NP}$, only special cases of MIS can be approximated within a constant factor, even in sequential polynomial time.

Next, we show that, given a graph and a partition of the vertices into κ cliques, both Graph TSP and TSP(1,2) can be solved optimally in $n^{\mathcal{O}(\kappa^2)}$ time. Moreover, we prove that optimal tours can be computed in time $n^{\mathcal{O}(\kappa)}$ for a certain class of instances, and we conjecture that the time complexity can be improved to $n^{\mathcal{O}(\kappa)}$ for any partition of a graph into κ cliques. Clearly, if the number of cliques is constant, then we obtain optimal solutions in polynomial time. This algorithm is applicable to the complements of graphs colorable with constant number of colors in polynomial time (e.g. complements of planar graphs). Our motivation was from the study of TSP in some *very dense in average* instances, such as the complements of planar graphs. We are not aware of a similar algorithm that exploits a partition into cliques for solving TSP optimally. Other special cases of TSP, which can be solved optimally in polynomial time, are presented in [LLRS85], Chapter 4.

Finally, we extend the ideas and the analysis of the $\frac{7}{6}$-algorithm for TSP(1,2) [PY93], so as to understand its performance for special cases of Graph TSP. As a result, we obtain a $\frac{22}{15}$-approximation algorithm for TSP in graphs of diameter 3, and a $\frac{17}{12}(\frac{7}{5})$-approximation algorithm for TSP in graphs that have diameter 4(3) and contain a perfect, triangle-free 2-matching. An algorithm presented in [Har84] can be used for deciding this class of graphs in polynomial time.

2 The Approximability of Dense Instances

The reduction of [PY93] implies that TSP(1,2) and Graph TSP are MAX–SNP-hard for dense in average graphs, namely graphs with $\Omega(n^2)$ edges. In this section, we provide an approximation preserving transformation from general instances of TSP(1,2) to dense instances of TSP(1,2) and Graph TSP.

Theorem 1. *There exist a constant $\delta^\star > 0$, such that, for all $\delta \leq \delta^\star$, δ-dense instances of TSP(1,2) do not admit a PTAS, unless $\mathcal{P} = \mathcal{NP}$.*

Proof. Let $G(V, E)$ be any graph of n vertices, and TSP_G be the optimal value of the corresponding TSP(1,2) instance. Clearly, $n \leq \mathrm{TSP}_G \leq 2n$. For any constant $\delta > 0$, let the X_δ be the independent set of δn vertices. For any $\delta > 0$, we construct a graph G' by adding X_δ to G and connecting each vertex of X_δ to each vertex of V. Formally, the new graph is $G'(V \cup X_\delta, E')$, where

$$E' = E \cup \{\{u, v\} : \forall (u, v) \in V \times X_\delta\}$$

Obviously, G' is a δ'-dense graph, $\delta' = \frac{\delta}{1+\delta} \geq \delta - \delta^2$. Let $\mathrm{TSP}_{G'}$ be the optimal value of the TSP(1,2) instance, that corresponds to G'.

By the construction of G', if we are given any spanning cycle $H_{G'}$ of G', we can find a spanning cycle H_G of G of length $l(H_G) \leq l(H_{G'})$. (Shortcut the vertices of X_δ in $H_{G'}$. This cannot increase the length of the cycle). Obviously, both cycles traverse the vertices of G in the same order.

Moreover, given any spanning cycle H_G of G, we can construct a spanning cycle $H_{G'}$ of G', that traverses the vertices of G in the same order with H_G, and has length

$$l(H_{G'}) = n + \max(\delta n, n - l(H_G))$$

If $u, v \in V$ are joined by an edge of length 2 in H_G, then a vertex of X_δ is imposed between them. If H_G consists of less than δn edges of length 2, then all edges of $H_{G'}$ will be of length 1. Consequently,

$l(H_{G'}) = (1+\delta)n$. Otherwise $(l(H_G) \geq (1+\delta)n)$, the vertices of X_δ cannot increase the length of the resultant cycle. Therefore, $l(H_{G'}) = l(H_G)$. The previous discussion justifies that the optimal values TSP_G and $\mathrm{TSP}_{G'}$ fulfill the following inequalities:

$$\mathrm{TSP}_{G'} - \delta n \leq \mathrm{TSP}_G \leq \mathrm{TSP}_{G'} \leq (1+\delta)\mathrm{TSP}_G$$

Let us assume that for some constant $\epsilon > 0$, TSP(1,2) is approximable within $(1+\epsilon)$ for δ'-dense graphs. Given any graph G, we can construct a δ'-dense graph G' by adding δn, $\delta = \frac{\delta'}{1-\delta'}$, independent vertices, and find an $(1+\epsilon)$ approximate solution $H_{G'}$ to this instance. Then, we can transform $H_{G'}$ to a solution H_G of the original instance by shortcuting the additional vertices. Clearly, the following inequalities hold:

$$l(H_G) \leq l(H_{G'}) \leq (1+\epsilon)\mathrm{TSP}_{G'} \leq (1+\epsilon)(1+\delta)\mathrm{TSP}_G$$

Since TSP(1,2) is MAX–SNP-hard, there exist a constant $\epsilon^\star > 0$, such that TSP(1,2) is not approximable within $(1+\epsilon^\star)$, unless $\mathcal{P} = \mathcal{NP}$ [ALMSS92]. Let $\alpha > 0$ be any small constant, and $\delta^\star = \frac{\epsilon^\star - \alpha}{1+\epsilon^\star-\alpha} > 0$. If δ-dense instances of TSP(1,2) admitted a PTAS for some $\delta \leq \delta^\star$, then general instances would be approximable within $(1+\epsilon^\star)$. □

Moreover, the previous transformation implies that approximating TSP(1,2) in dense instances is essentially as hard as approximating TSP(1,2) in general instances. This statement becomes clear by considering instances with large optimal values, namely graphs G such that $\mathrm{TSP}_G \geq (1+\delta)n$. Then, the previous transformation is an L-reduction [PY91] from general instances to δ'-dense instances. However, Theorem 1 leaves open the possibility that there exist some constants $\delta > \delta^\star$, such that δ-dense graphs with optimal tours of length less than $(1+\delta)n$ admit a PTAS for TSP(1,2).

It is trivial to check that the diameter of the graph G' is equal to 2. Thus, the corresponding instance of Graph TSP is a dense instance of TSP(1,2). This implies that the restriction of Graph TSP to dense instances is essentially at least as hard to approximate as TSP(1,2) in general instances.

Corollary 2. *There exist a constant $\delta^\star > 0$, such that, for all $\delta \leq \delta^\star$, δ-dense instances of Graph TSP do not admit a PTAS, unless $\mathcal{P} = \mathcal{NP}$.*

Remark. It easy to check that, independently of the structure of the original instance G, the graph G' is an expander graph with expansion factor at least $\frac{2\delta}{1+\delta} = 2\delta'$. Therefore, the inapproximability results also hold for dense expander graphs.

3 An $\mathcal{NC}$ Approximation Algorithm for TSP(1,2)

Let $G(V,E)$ be any graph of n vertices, TSP_G be the optimal value of the corresponding TSP(1,2) instance, and MIS_G be the cardinality of the Maximum Independent Set of G. Next, we show that a simple $\mathcal{NC}$ algorithm produces a tour of length at most $n + \mathrm{MIS}_G$, and an Independent Set of cardinality at least $\mathrm{TSP}_G - n$. This $\mathcal{NC}$ algorithm outperforms all known sequential approximation algorithms for TSP(1,2), if the input is restricted to graphs with Independent Sets of cardinality less than $\frac{n}{6}$. Moreover, it matches the performance of the best known $\mathcal{RNC}$ algorithm for general instances.

Theorem 3. *The following inequalities hold for any graph $G(V,E)$ of n vertices:*

$$2\mathrm{MIS}_G \leq \mathrm{TSP}_G \leq n + \mathrm{MIS}_G \tag{1}$$

Moreover, a tour and an independent set that fulfill (1) can be computed in $\mathcal{NC}$.

Proof. The main part of the proof consists of an algorithm that computes a tour and an independent set of G, that fulfill (1). Consider the following algorithm:

```
Algorithm TSP-MIS
Input:  A graph G(V, E).
Output: A set P of edges that cover V with vertex disjoint simple paths.
        A set S* of independent vertices.
P := ∅;
i := 0;
repeat
     i := i + 1;     /* i^th phase */
     V^i := {v ∈ V : deg_P(v) = 0};
     For any non-trivial path p ∈ P, add exactly one of the end vertices of p to V^i;
     E^i := {{v, w} ∈ E : v ∈ V^i ∧ w ∈ V^i};
     Find a Maximal Matching M^i in G^i(V^i, E^i);
(*)  P := P ∪ M^i;
     S^i := {v ∈ V^i : deg_{M^i}(v) = 0};
until M^i is empty or i > K;
S* := S^i of maximum cardinality;
return(P, S*);
```

Correctness: The algorithm proceeds in phases, such that the following holds for the i^{th} phase: *For all vertices v, $v \in V^i$ implies that $\deg_P(v) \leq 1$. Moreover, for each path $p \in P$, exactly one of the end vertices of p is in V^i.*

Therefore, at the end of the algorithm, the set P consists of simple paths (of 1-edges) and isolated vertices. If the isolated vertices are considered as paths of length 0 (trivial paths), the edge set P covers V with vertex disjoint simple paths. Additionally, the vertex sets S^i are independent sets, because M^i is a maximal matching in $G^i(V^i, E^i)$.

Performance: The performance of the algorithm is determined by the number of edges (of length 1) that are contained in the set P at the end of the algorithm. Initially, the set P is empty. At each phase i, the edges of M^i are added to P in step (*). Let $|M^i|$ be the number of edges of a maximal matching M^i of the graph G^i. The algorithm runs for $K+1$ phases, i.e. $i = 1, \ldots, K+1$, where K will be fixed later. Clearly, $|P| = \sum_{i=1}^{K} |M^i|$.

Since the vertices of V^i that are not covered by a maximal matching M^i form an independent set, and any independent set of G^i contains at most MIS_G vertices, we obtain that:

$$\frac{|V^i| - \mathrm{MIS}_G}{2} \leq |M^i| \leq \frac{|V^i|}{2}$$

Furthermore, for any pair of vertices of V^i that are matched by M^i, exactly one vertex is added to V^{i+1}. Therefore, the following holds for the size of V^{i+1}:

$$\frac{|V^i|}{2} \leq |V^{i+1}| = |V^i| - |M^i| \leq \frac{|V^i| + \mathrm{MIS}_G}{2}$$

The following inequality holds for all $i \geq 1$, and can be proved by induction on i.

$$|V^{i+1}| \leq \frac{|V^1| + (2^i - 1)\mathrm{MIS}_G}{2^i} \tag{2}$$

If $K = \lceil \log n \rceil$, then inequality (2) implies that

$$|V^{K+1}| = |V^{\lceil \log n \rceil + 1}| \leq \frac{|V^1| + (n-1)\mathrm{MIS}_G}{n} < \mathrm{MIS}_G + 1$$

By summing the equalities $|V^{i+1}| = |V^i| - |M^i|$, for $1 \leq i \leq K+1$, we obtain that

$$|V^{K+1}| = |V^1| - \sum_{i=1}^{K} |M^i| = |V| - |P|$$

Therefore, we obtain that $|P| \geq n - \mathrm{MIS}_G$, because if $K = \lceil \log n \rceil$, then $|V^{K+1}| \leq \mathrm{MIS}_G$. Since the set P consists of simple paths of total length at least $n - \mathrm{MIS}_G$, we can construct a tour from P by adding no more than MIS_G edges of length 2. The length of the resultant tour will be no more than $n + \mathrm{MIS}_G$. Therefore, $\mathrm{TSP}_G \leq n + \mathrm{MIS}_G$.

Complexity: The algorithm runs for at most $\lceil \log n \rceil + 1$ phases. The basic operations of each phase can be implemented in $\mathcal{NC}$ (cf. Chapter 2 of [GR88]). The P-degrees of the vertices are updated in step (*) in constant parallel time. Since the connected components of P are simple paths, the computation of G^i can be implemented in $\mathcal{NC}$. The total complexity of the algorithm is dominated by the computation of the maximal matching. There exist a CRCW PRAM algorithm that produces a maximal matching in $\mathcal{O}(\log^3 n)$ time using $\mathcal{O}(n^2)$ processors [IS86]. Thus, the TSP-MIS algorithm can be executed in a CRCW PRAM machine in $\mathcal{O}(\log^4 n)$ time using $\mathcal{O}(n^2)$ processors. Moreover, it is straight forward to check that the sequential complexity of TSP-MIS is $\mathcal{O}(n^2 \log n)$.

Next, we prove that $2\mathrm{MIS}_G \leq \mathrm{TSP}_G$. Let $\mathrm{TSP}_G = n + \gamma$, for any $0 \leq \gamma \leq n$, and let $\pi : [n] \mapsto V$ be a permutation that corresponds to an optimal spanning cycle of G. Obviously, π defines a set $\{p_1, p_2, \ldots, p_\gamma\}$ of γ simple paths, which consist of 1-edges and are linked to a tour by 2-edges. Clearly, if a path p_i consists of $|p_i|$ vertices, it cannot contribute to any independent set more than $\frac{|p_i|+1}{2}$ vertices, if $|p_i|$ is odd, and $\frac{|p_i|}{2}$ vertices, if $|p_i|$ is even. Therefore, the cardinality of any independent set in G cannot be greater than the sum of the cardinalities of the independent sets of p_i's. Consequently,

$$\begin{aligned} \mathrm{MIS}_G &\leq \sum_{i=1}^{\gamma} \left\lceil \frac{|p_i|}{2} \right\rceil \\ &\leq \sum_{i=1}^{\gamma} \frac{|p_i|+1}{2} \\ &\leq \frac{n+\gamma}{2} = \frac{\mathrm{TSP}_G}{2} \end{aligned}$$

It remains to prove that the algorithm TSP-MIS computes in $\mathcal{NC}$ an independent set of cardinality at least $\mathrm{TSP}_G - n$. Assume that $\mathrm{TSP}_G - n = \gamma > 0$. We run TSP-MIS for $K = \lceil \log n \rceil$ phases, and we output the set $S^\star$, namely the set S^i, $i = 1, \ldots, K+1$, of maximum cardinality. Clearly, S^i's are sets of independent vertices. If $|S^\star| = \gamma' < \gamma$, we can substitute the MIS_G with γ' in the analysis on the size of P. Therefore, if $|S^i| < \gamma$, for all $1 \leq i \leq K+1$, the resultant set P would contain more than $n - \gamma$ edges. This is a contradiction, because it implies that we can compute a super-optimal tour. Consequently, the algorithm TSP-MIS always produces an independent set of cardinality at least $\mathrm{TSP}_G - n$.

There exist instances such that $\mathrm{TSP}_G = n + \mathrm{MIS}_G - 1$. For example, for any $x > 1$, consider a graph G consisting of a clique of $n - x$ vertices and an independent set of x vertices, all of them connected to the same vertex of the clique. Clearly, $\mathrm{MIS}_G = x + 1$, and $\mathrm{TSP}_G = n + x$.

Also, the inequality $2\mathrm{MIS}_G \leq \mathrm{TSP}_G$ becomes tight for many instances. It is worth mentioning the C_n graph (a simple cycle of n vertices), and the complete bipartite graph, where the sizes of the classes are δn and $(1-\delta)n$, for any $\delta > 0$. □

The algorithmic consequences of Theorem 3 are summarized by the following corollaries:

Corollary 4. *TSP(1,2) can be approximated within $\frac{3}{2}$ by an $\mathcal{NC}$ algorithm that runs in a CRCW PRAM in time $\mathcal{O}(\log^4 n)$ using $\mathcal{O}(n^2)$ processors.*

Corollary 5. *TSP(1,2) restricted to graphs with Maximum Independent Set no more than αn is $(1+\alpha)$-approximable by an $\mathcal{NC}$ algorithm that runs in a CRCW PRAM in time $\mathcal{O}(\log^4 n)$ using $\mathcal{O}(n^2)$ processors.*

Corollary 6. *Maximum Independent Set restricted to graphs with optimal value of TSP(1,2) greater than $(1+\beta)n$ is $\frac{2\beta}{1+\beta}$-approximable by an $\mathcal{NC}$ algorithm that runs in a CRCW PRAM in time $\mathcal{O}(\log^4 n)$ using $\mathcal{O}(n^2)$ processors.*

The algorithm TSP-MIS outperforms the algorithm of Papadimitriou and Yannakakis [PY93], if we restrict our attention to graphs with Maximum Independent Set less than $\frac{n}{6}$. Furthermore, these algorithms can be combined as follows: (1) Run the algorithm of Papadimitriou and Yannakakis; (2) Initialize P to the set of paths produced by the step (1), and output the result of TSP-MIS.

Remark. TSP-MIS is based on the local search paradigm, that has been modified so as to be implemented in the class $\mathcal{NC}$. Then, we show that the cardinality of the Maximum Independent Set always bounds from above the distance between the length of any local optimum tour and $\mathrm{TSP}_G \geq n$.

We use a sequential 4-local search algorithm for TSP(1,2) that has been analyzed in [KMSV94]. Their algorithm starts with an arbitrary tour, and in each iteration, it checks if the current tour contains two disjoint edges (v, w) and (v', w'), such that replacing them with the edges (v, v') and (w, w') yields a tour of smaller cost. Consider an arbitrary orientation of a local optimum tour t. Let $S_L = \{v \in V : v$ is the left vertex of a 2-edge of $t\}$, and S_R be defined similarly for the right vertices. Obviously, the local optimality of t under the above neighbourhood implies that both S_L and S_R are independent sets. Therefore, any local optimum tour cannot have more than MIS_G 2-edges. $\square$

4 Optimal Tours through Constant Number of Cliques

Then, we show that, given a graph $G(V, E)$ and a partition of V into a constant number of cliques, we can decide if G is Hamiltonian in polynomial time. This implies that the corresponding instances of TSP(1,2) and Graph TSP can be solved optimally in polynomial time.

Let CCP (*Constant number of Cliques in Polynomial time*) be the class of graphs, such that a partition of the vertices into constant number of cliques can be computed in polynomial time. The class CCP consists of the complements of graphs, which can be colored with constant number of colors in polynomial time. Thus, CCP is a non-trivial class. An interesting subclass of CCP consists of the complements of planar graphs. In particular, we are able to prove the following theorem for graphs in CCP.

Theorem 7. *Given a graph $G(V, E)$, $|V| = n$, and a partition of V into κ cliques, there exists a deterministic algorithm that runs in time $\mathcal{O}\left(n^{\kappa(2\kappa-1)}\right)$ and decides if G is Hamiltonian. If G is Hamiltonian, the algorithm outputs a Hamiltonian Cycle.*

Proof. Given a set $M \subseteq E$ of inter-clique edges, the *clique graph* $T(C, M)$ contains exactly κ vertices, that correspond to the cliques of $\mathcal{C}$, and represents how the edges of M connect the different cliques. For any spanning subgraph $H(V, E_H)$ of G, there exists a clique graph $T_H(C, M_H)$ that corresponds to H. Obviously, if H is connected, then T_H is connected, and if H is a Hamiltonian Cycle, then T_H is euclidean. However, the converse statements are not always true.

On the other hand, any clique graph $T(C, M)$ corresponds to a family $\mathcal{H}$ of spanning subgraphs of G, that share M as the set of inter-clique edges. The graphs of $\mathcal{H}$ may differ from each other with respect to the edges connecting vertices that belong to the same clique. A clique graph T is an *HC-clique graph*, if the family $\mathcal{H}$ contains at least one spanning cycle (Figure 1.b).

Given a clique graph $T(C, M)$, we color an edge RED, if it shares a vertex of G with another edge of M. Otherwise, we color it BLUE. The corresponding edges of G are colored with the same colors, while the remaining edges $(E - M)$ are colored BLACK. Additionally, we color RED each vertex $w \in V$, which is the common end vertex of two or more (RED) edges. We color BLUE each vertex $w \in V$, which is adjacent to exactly one edge of M (RED or BLUE). The remaining vertices of G are colored BLACK (Figure 1.a). Notice that the coloring of the vertices and the edges of G is completely determined by the set M of inter-clique edges.

Let H be any spanning cycle of G and T be the corresponding HC-clique graph. Obviously, RED vertices cannot be exploited for visiting any BLACK vertices belonging to the same clique. Let H visit a clique C_i through a vertex v, and leaves C_i through w. Then $v, w \in C_i$ consist a pair of BLUE vertices. Clearly, H can use some BLACK edges for visiting some BLACK vertices of C_i. A BLUE pass through a clique C_i is a simple path P of length at least one, that entirely consists of non-RED vertices of C_i. We say that *a clique C_i is covered by M*, if all the vertices of C_i have degree at most 2 in M, and the existence of a non-RED vertex implies the existence of at least one BLUE vertex pair. The following proposition, which is proved in the Appendix, characterizes the class of HC-clique graphs.

Proposition 8. *A connected, eulerian, clique graph $T(C, M)$ is an HC-graph iff it fulfills the following property (HC-Property):*

(HC1) For any $i = 1, \ldots, \kappa$, C_i is covered by M; and

(HC2) There exists an eulerian trail R for T such that: For any RED *vertex $v \in V$, R traverses the* RED *edges that are incident to v in consecutive order (i.e. R passes through any* RED *vertex exactly once using the corresponding* RED *edge pair).*

Proof. By definition, if T is an HC-graph, then there exists a cycle H that spans V. Additionally, the set of inter-clique edges of H is exactly M. Since any vertex of V has degree 2 in H, all the cliques C_i are covered by M. Also, for any RED vertex v, H traverses the RED edges that are incident to v consecutively. Clearly, if we remove the BLACK vertices and edges from H, then we obtain a traversal of the edges of M that fulfills (HC2). Moreover, since H is a spanning cycle, the resultant traversal is an eulerian trail for T.

Conversely, we will show that given a connected, eulerian clique graph $T(C, M)$, that fulfills the HC-Property, we can easily construct a Hamiltonian Cycle H, such that the set of inter-clique edges of H is exactly M. Let R be an eulerian trail for T that fulfills (HC2). The cycle H will follow the trail R. Since T is connected and all the cliques are covered by M, the trail passes through all the cliques $C_i \in \mathcal{C},\ i = 1, \ldots, \kappa$. The Property (HC2) implies that every time R visits a RED vertex v, H passes through v using the corresponding RED edge pair. Since R is an eulerian trail, all the RED vertices are included in H with degree 2. The first time that R passes through a BLUE vertex pair $v, w \in C_i$ (case (a)), a path consisting of all the BLACK vertices of C_i is imposed between v and w in H. The Property (HC1) implies that all the BLACK vertices are included in H with degree 2. If R passes through a BLUE vertex pair $v, w \in C_i$, and C_i does not contain any unvisited BLACK vertices, then the BLACK edge $\{v, w\}$ is added to H. Since R is an eulerian trail that fulfills (HC2), all the BLUE vertices are included in H with degree 2. Moreover, H is connected and the inter-clique edges of H are exactly the edges of M. □

The proof of Proposition 8 implies a polynomial time, deterministic procedure for deciding if a clique graph $T(C, M)$ is an HC-clique graph. Moreover, in case that T is an HC-graph, this procedure outputs a Hamiltonian Cycle. The time complexity of this deterministic decision procedure is polynomial in n and κ. For convenience, we consider that an HC-clique graph can be decided in time $\mathcal{O}(n^{\kappa})$.

It remains to show how to exploit the partition of G into κ cliques so as to decide Hamiltonicity. Intuitively, if G is Hamiltonian, there should exist a spanning cycle H that takes advantage of the cliques of G. Namely, H is not expected to pass through the cliques many times. Clearly, if we prove that G is Hamiltonian iff it contains a spanning cycle that passes through the cliques a small number of times, then this cycle could be found using exhaustive search. The previous discussion implies that this is equivalent to the existence of an HC-clique graph with a small number of edges. The following technical lemmas formalizes the aforementioned intuition.

Lemma 9. *Let $B_\kappa \geq 2$ be some integer only depending on κ, such that the following is true for any graph $G(V,E)$, any partition of V into κ cliques: If G is Hamiltonian and $|V| > B_\kappa$, G contains at least one Hamiltonian Cycle not entirely consisting of* RED *vertices (inter-clique edges).*
Then, for any graph $G(V,E)$ and any partition of V into κ cliques, G is Hamiltonian iff it contains a Hamiltonian Cycle with at most B_κ inter-clique edges.

Proof. Let H be the Hamiltonian Cycle of G containing the minimum number of inter-clique edges and M be the corresponding set of inter-clique edges. Assume that $|M| > B_\kappa$. Because of (i), H cannot entirely consist of RED vertices. Therefore, H should contain at least one BLUE vertex pair.

We substitute any BLUE pass of H through a clique C_i with a single (RED) super-vertex $\hat{v}$, that is also considered to belong to the clique C_i. Hence, $\hat{v}$ is connected to all the remaining vertices of C_i. These substitutions result in a cycle H' that entirely consists of RED vertices, and contains exactly the same set M of inter-clique edges with H.

Obviously, the substitutions of all the BLUE passes of H with single RED super-vertices belonging to the corresponding cliques result in a graph $G'(V', E')$ different from G. However, since H' is a Hamiltonian Cycle, G' is also Hamiltonian, and V' is partioned into κ cliques. Moreover, it is not hard to verify that, if H' is any Hamiltonian Cycle of G', then the reverse substitutions of all the RED super-vertices $\hat{v}$ with the corresponding BLUE passes result in a Hamiltonian Cycle of G, that contains exactly the same set of inter-clique edges with H' (Figure 1).

Since H' is a Hamiltonian Cycle that entirely consists of RED vertices and contains more than B_κ inter-clique edges, the statement (i) implies that there exists another Hamiltonian Cycle of G', that contains strictly less inter-clique edges than H'. Therefore, there exists a Hamiltonian Cycle of G, that contains strictly less inter-clique edges than H. Obviously, this contradicts the selection of H. □

Consequently, for proving that G is Hamiltonian iff there exists a Hamiltonian Cycle using at most B_κ inter-clique edges, it only suffices to prove the same bound for RED Hamiltonian Cycles (i.e. Hamiltonian Cycles that entirely consist of inter-clique edges). Intuitively, the class of graphs for which all the Hamiltonian Cycles entirely consist of inter-clique edges should be very restricted.

Lemma 10. *Given a graph $G(V,E)$ and a partition of V into κ cliques, G is Hamiltonian iff there exists an HC-clique graph $T(C,M)$, such that $|M| \leq \kappa(\kappa-1)$.*

Proof. By definition, the existence of an HC-clique graph for G implies that G is Hamiltonian (independently of the number of edges of M). Thus, (2) implies (1) trivially.

Conversely, let H be the Hamiltonian Cycle of G that contains the minimum number of inter-clique edges, and $T(C,M)$ be the corresponding HC-clique graph. If $|M| \leq \kappa(\kappa-1)$, then we are done. Otherwise, Lemma 9 implies that it suffices to consider only Hamiltonian Cycles H, that entirely consist of inter-clique edges.

Consequently, assume that the coloring of V under M entirely consists of RED vertices, and consider an arbitrary orientation of the Hamiltonian Cycle H (e.g. a traversal of the edges of H in the clockwise direction). If there exist a pair of cliques C_i and C_j and four vertices $v_1, v_2 \in C_i$ and $u_1, u_2 \in C_j$, such that both v_x are followed by u_x $(x = 1, 2)$ in a traversal of H, then the (BLACK)

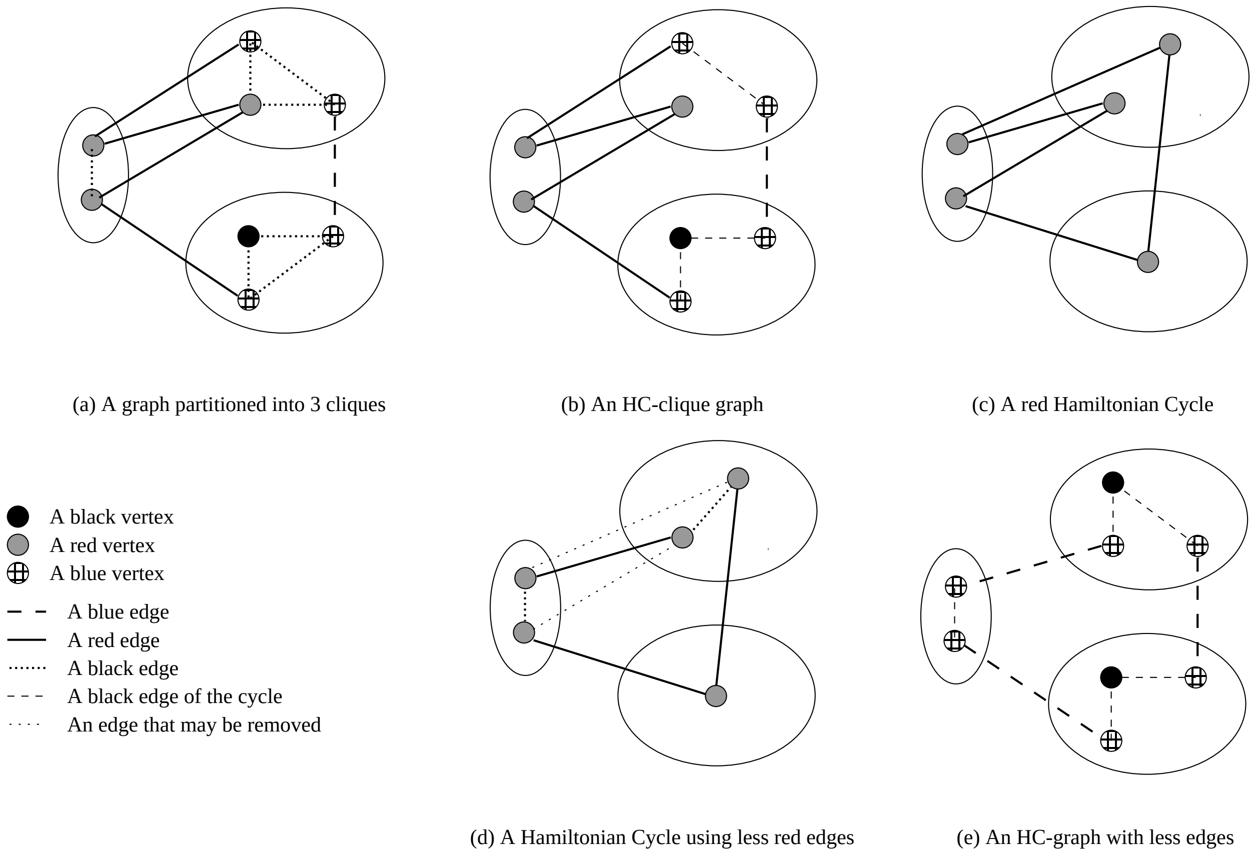

Fig. 1. A decrease in the number of inter-clique edges contained by a Hamiltonian Cycle.

edges $\{v_1, v_2\}$ and $\{u_1, u_2\}$ can be exploited in order to obtain a Hamiltonian Cycle containing less inter-clique edges than H. Clearly, this would be a contradiction to the selection of H. The previous situation can be avoided, only if for all $i = 1, \ldots, \kappa$, and $j = 1, \ldots, \kappa$, $j \neq i$, at most one vertex $v_j \in C_i$ is followed by a vertex u of the clique C_j in a traversal of H. Obviously, this restriction results in Hamiltonian Cycles H that contain at most $\kappa(\kappa-1)$ inter-clique edges. Hence, any Hamiltonian Cycle containing more than $\kappa(\kappa-1)$ inter-clique edges cannot be a Hamiltonian Cycle with the minimum number of inter-clique edges. This contradicts the selection of H and the corresponding HC-clique graph T. □

Lemma 10 implies that we can decide if G is Hamiltonian in time $\mathcal{O}\left(n^{\kappa(2\kappa-1)}\right)$, because the number of the different edge sets containing at most $\kappa(\kappa-1)$ inter-clique edges is at most $n^{2\kappa(\kappa-1)}$, and we can decide if a clique graph is an HC-graph in time $\mathcal{O}(n^{\kappa})$. □

Then, we prove that if G contains a BLUE Hamiltonian Cycle (i.e. a Hamiltonian Cycle that does not contain any RED vertices and edges), then there exists a BLUE Hamiltonian Cycle using at most $2(\kappa-1)$ inter-clique edges.

Lemma 11. *Given a graph $G(V, E)$ and a partition of V into κ cliques, if G contains a* BLUE *Hamiltonian Cycle, then there exists a* (BLUE) *HC-clique graph $T(C, M)$, such that $|M| \leq 2(\kappa-1)$.*

Proof. Let H be the BLUE Hamiltonian Cycle of G that contains the minimum number of inter-clique edges, and $T(C, M)$ be the corresponding HC-clique graph. If $|M| \leq 2(\kappa-1)$, then we are done. Notice that, since RED vertices cannot be created by removing edges, any eulerian, connected, spanning subgraph of T is an HC-graph that corresponds to a BLUE Hamiltonian Cycle of G.

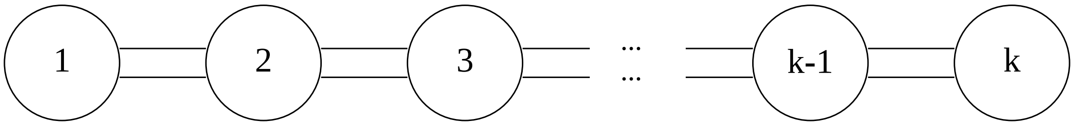

Fig. 2. An HC-graph that contains exactly $2(\kappa-1)$ inter-clique edges.

Let $S_T(C, M_S)$ be any spanning tree of T. Since M_S consists of $\kappa-1$ edges, the graph $T^{(S)}(C, M-M_S)$, obtained by removing the edges of the spanning tree from T, contains at least κ edges. Therefore, $T^{(S)}$ contains a simple cycle L. The removal of the edges of L does not affect connectivity (the edges of L do not touch the spanning tree), and subtracts 2 from the degrees of the involved vertices. Clearly, the graph $T'(C, M-L)$ is an HC-graph with less edges than T. This contradicts to the selection of T and H. □

There exist HC-graphs that contain exactly $2(\kappa-1)$ edges and correspond to a Hamiltonian Cycle using the minimum number of inter-clique edges (Figure 2). Therefore, the bound of $2(\kappa-1)$ is tight. However, even if we allow RED vertices, we are not able to construct examples of HC-clique graphs that contain more than $2(\kappa-1)$ edges, and correspond to Hamiltonian Cycles using the minimum number of inter-clique edges. Therefore, we conjecture that the bound of $2(\kappa-1)$ holds for any graph and any partition into κ cliques. An inductive (on the cardinality of V) application of Lemma 9 suggests that this conjecture is equivalent to the following:

Conjecture 12. *For any Hamiltonian graph $G(V,E)$ of $2\kappa-1$ vertices and any partition of V into κ cliques, there exists at least one Hamiltonian Cycle not entirely consisting of inter-clique edges.*

The previous conjecture is very similar to a theorem proved by C.A.B. Smith in 1946. It states that the number of Hamiltonian Cycles that contain any given edge of a cubic graph (i.e. a simple regular graph of degree 3) is even. A simple algebraic proof of this theorem can be found in [Ber85]. If we only consider Hamiltonian graphs, Smith's Theorem can be applied to graphs $G(V,E)$ consisting of a Hamiltonian Cycle and a perfect matching. On the other hand, Conjecture 12 has a weaker conclusion than Smith's Theorem, but it is applicable to the graphs consisting of a RED Hamiltonian Cycle and κ cliques of arbitrary cardinalities.

Remark. Notice that our algorithm works for all graphs, integers $\kappa > 1$, and partitions into κ cliques. Also, any graph can be partitioned into a (non-constant) number of cliques in polynomial time (e.g. maximum matching). Hence, any (deterministic) algorithm that (i) exploits a partition into κ cliques for deciding Hamiltonicity, and (ii) works for all graphs, integers $\kappa > 1$, and partitions into κ cliques (as our algorithm does), should run in time exponential in κ (e.g. $n^{\mathcal{O}(\kappa)}$, $2^{\mathcal{O}(\kappa)}$), unless $\mathcal{NP}$ has (deterministic) subexponential simulations.

The proof of Lemma 11 is related to the Minimum Eulerian Subgraph problem. Given a connected, eulerian (multi)graph $G(V,E)$, the Minimum Eulerian Subgraph problem is to find a connected, eulerian, spanning subgraph of G with the minimum number of edges. Clearly, G contains a Hamiltonian Cycle iff there exists a connected Eulerian Subgraph with exactly $|V|$ edges. Thus, Minimum Eulerian Subgraph is $\mathcal{NP}$-complete, because any connected graph becomes eulerian (multigraph), if we double its edges. Clearly, this transformation does not affect the size the longest simple cycle. Consider the following heuristic: delete the edges of a minimum spanning tree from G, and while the residual graph contains more than $n-1$ edges, delete the edges of a simple cycle. This simple heuristic can be implemented in polynomial time. Additionally, the resultant graph is connected, eulerian and contains at most $2(n-1)$ edges. Thus, it is a 2-approximation for the Minimum Eulerian Subgraph problem. □

4.1 A Reduction from TSP to Hamiltonian Cycle

Next, we prove that, given a graph $G(V, E)$ and a partition of V into constant number of cliques, the corresponding instances of TSP(1,2) and Graph TSP can be reduced (in polynomial time) to the Hamiltonian Cycle problem.

Theorem 13. *Given a connected graph $G(V, E)$ of n vertices and a partition of V into $\kappa \geq 1$ cliques, an optimal solution to the corresponding Graph TSP/TSP(1,2) instance can be found in time $\mathcal{O}\left(n^{\kappa(2\kappa+3)}\right)/\mathcal{O}\left(n^{\kappa(2\kappa+1)}\right)$.*

Proof. Let $\mathcal{A}$ be the algorithm of Theorem 7. At first, we prove the claim for Graph TSP. Let $T(C, M)$ be the clique graph of G, and $T_S(C, M_S)$ be any spanning tree of T. Since T is connected, the total length of the edges of T_S is $\kappa - 1$. For each edge $e \in M_S$, we add an edge e' parallel to e. Namely, if the cliques C_i and C_j are joined in T_S by an edge $e = \{v, w\}$, $v \in C_i, w \in C_j$, we add an edge $e' = \{w', v'\}$, $v' \in C_i, w' \in C_j$, where v', w' are different from v, w, if $|C_i|, |C_j|$ suffice. The length of e' is at most 3 (at most 1 for $\{v', v\}$, 1 for $\{v, w\}$, and at most 1 for $\{w, w'\}$). Let M'_S be the resultant edge set. The Minimum Spanning Tree heuristic results in a tour of length no more than the total length of the edges of M'_S. Therefore, $\mathrm{TSP}_G \leq n + 2(\kappa - 1)$, and any optimal tour contains at most $2(\kappa - 1)$ non-edges of G.

We can find an optimal tour, by calling $\mathcal{A}$ at most $K = \mathcal{O}\left(n^{4(\kappa-1)}\right)$ times with input the graphs $G_i(V, E \cup N_i)$, $i = 1, \ldots, K$. The sets N_i are all possible subsets of non-edges of G with at most $2(\kappa - 1)$ elements, including the empty one. Let G_i be a Hamiltonian graph, such that the edges contained in the corresponding set N_i are of minimum total length. Then, the Hamiltonian Cycle produced by $\mathcal{A}(G_i)$ is an optimal tour for G.

It is easy to check that the length of an optimal tour for the corresponding TSP(1,2) instance is at most $n + \kappa - 1$. Therefore, at most $\mathcal{O}\left(n^{2(\kappa-1)}\right)$ calls to the algorithm $\mathcal{A}$ are required for computing an optimal tour for TSP(1,2). □

Remark. Obviously, Conjecture 12 implies that, given a graph $G(V, E)$ and a partition of V into κ cliques, optimal solutions for TSP(1,2) and Graph TSP can be computed in time $n^{\mathcal{O}(\kappa)}$.

5 Approximating TSP in Graphs with Small Diameter

In this section, we extend the ideas of the $\frac{7}{6}$-approximation algorithm for TSP(1,2) [PY93] so as to improve the approximation ratio for TSP in graphs with diameter 3 and 4. First, we study graphs that contain a perfect, triangle-free 2-matching, and we obtain a $\frac{7}{5}$-approximation algorithm, if the diameter is 3, and a $\frac{17}{12}$-approximation, if the diameter is 4. Also, we obtain a $\frac{22}{15}$-approximation algorithm for general graphs with diameter 3. Our algorithms are based on the use of 2-matchings as a relaxation for TSP.

A *2-matching* M_2 of a graph $G(V, E)$ is a subset of edges ($M_2 \subseteq E$), such that any vertex of V has degree at most 2 in M_2 (cf. [Har84] for a detailed study). A 2-matching is called *simple*, if no edge of E is included in M_2 more than once (i.e. M_2 does not contain cycles of length 2), and it is called *perfect*, if all the vertices of V have degree exactly 2. The computation of a maximum simple 2-matching of a graph is a combinatorial optimization problem, that has been extensively studied. If the edges of E are weighted, then we seek a simple 2-matching of maximum weight, otherwise we seek a simple 2-matching of maximum cardinality. Hartvigsen [Har84] described polynomial time, primal-dual algorithms for the computation of a maximum weighted, simple 2-matching, and a maximum cardinality, simple, triangle-free 2-matching. Moreover, he conjectured that there exists a polynomial time, primal-dual algorithm for the computation of a maximum weighted, simple, triangle-free 2-matching (cf. [CW97]).

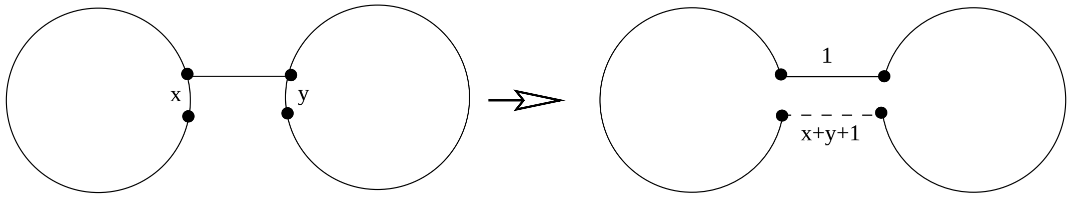

Fig. 3. An example of cycle patching.

Obviously, matchings of maximum weight can be translated to matchings of minimum length (and vice versa), by setting $w_{ij} = W - d_{ij}$, for some large enough W. It is well known that a perfect, simple, κ-cycle-free, 2-matching of minimum length is a natural relaxation for TSP, since any tour is such a 2-matching by itself. Moreover, the relaxation becomes tighter as κ tends to $|V| - 1$.

The algorithm of Papadimitriou and Yannakakis for TSP(1,2) uses the well-known technique of subtour patching. Two cycles (subtours) can be patched together by picking one edge in each, deleting these two edges, and connecting the four end vertices in any one of the other two ways (Figure 3). The fact that cycles with 2-edges can be patched together at no additional cost is crucial to the analysis of [PY93]. Clearly, this is not true for instances of Graph TSP with diameter greater than 2. However, in case of Graph TSP, the extra cost incurred can be bounded if the diameter of the graph is very small (i.e. 3 or 4);

Next, we obtain improved approximation algorithms for TSP in graphs, that have diameter 3 or 4, and contain a triangle-free 2-matching. This class of graphs is decidable in polynomial time and strictly includes the class of Hamiltonian graphs with diameter at most 4.

Theorem 14. *Let $G(V, E)$ be a graph of diameter 4(3), such that E contains a perfect, simple, triangle-free 2-matching. Then Graph TSP in G is approximable in polynomial time within $\frac{17}{12}(\frac{7}{5})$.*

Proof. Wlog. we consider connected input graphs $G(V, E)$, $|V| = n$. We can use the algorithm of Hartvigsen [Har84] for computing a perfect, simple, triangle-free 2-matching of length n, because G is assumed to contain such a 2-matching. Let L be the set of the cycles of an optimum 2-matching. We assume that $|L| > 1$, otherwise we are done. We form a bipartite graph B with vertex classes L and V. The graph B contains an edge from a cycle $l \in L$ to a vertex $v \in V$, if v does not belong to l and there exists an edge $\{w, v\} \in E$ from a vertex w of l to v.

We start with the case that G is Hamiltonian, and $\text{TSP}_G = n$. Since G is Hamiltonian, we can find a matching in B covering all cycles in L. Then, we construct a directed graph $D(L, A)$. The vertex set of D corresponds to the cycles of L, and an arc $(l, l') \in A$, if l is matched in B with a vertex v of l'. Obviously, since any cycle of L is matched with exactly one vertex v, the graph D is a function, i.e. any vertex of D has outdegree 1. The following lemma is proved in [PY93]:

Lemma 15 [PY93]. *Any function D has a spanning subgraph consisting of vertex-disjoint in-trees of depth 1 and paths of length 2. Moreover, such a decomposition can be found in polynomial time.*

By the construction of D, any arc $(l, l') \in A$ corresponds to an edge $\{v, v'\} \in E$, such that v is in cycle l, and v' is the vertex of l' to which l is matched. Therefore, if two arcs have the same head l' in D, then the corresponding edges are incident to distinct vertices of G. Clearly, this may not be the case if l' is the head of one arc and the tail of another one.

We decompose D as indicated by Lemma 15. All the cycles that belong to the same component are merged together. If a component is an in-tree, then it consists of a cycle l (the root), and other cycles $l_1, \ldots, l_k$, each of them adjacent to a different vertex $v_1, \ldots, v_k$ of l. We go around the cycle l in a clockwise direction, starting from any vertex of l, that is not connected to a cycle l_i, if such a

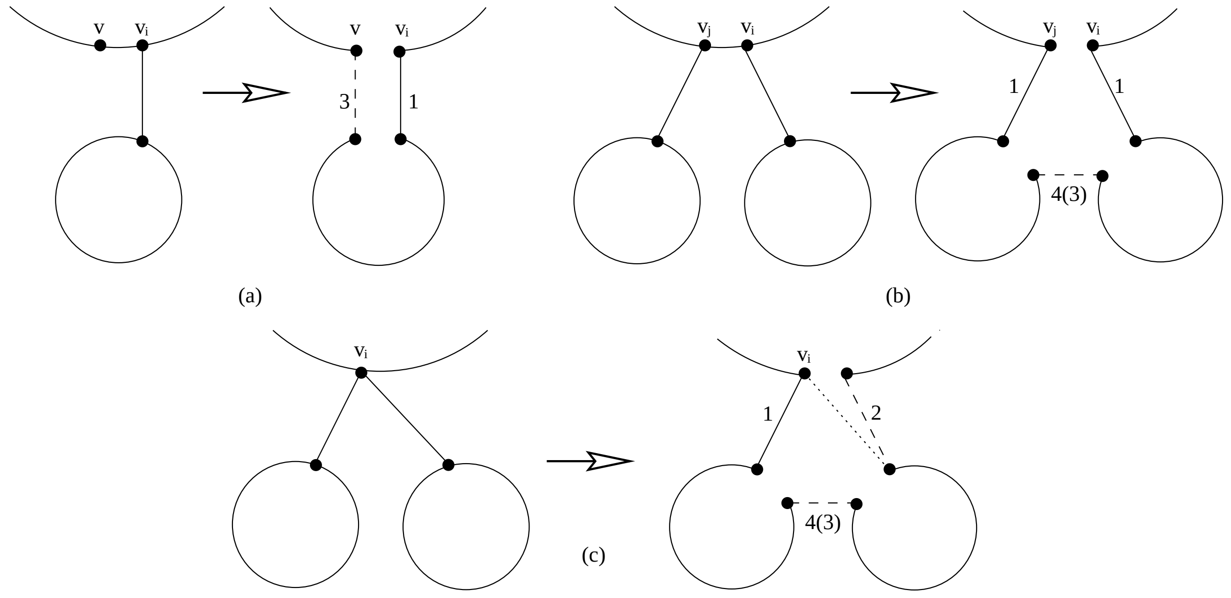

Fig. 4. The three cases of cycle merging.

vertex exists, or from an arbitrary vertex of l, otherwise. Whenever we encounter a vertex v_i (the mate of a cycle l_i), we look at the next vertex of l. If the next vertex is not a mate of a cycle l_j, $1 \leq j \leq k, j \neq i$, then we merge the cycle l_i with l as shown by Figure 4.a. Otherwise (the next vertex is the mate of a cycle l_j), we merge l_i and l_j with l as shown by Figure 4.b.

If a component is a path of length 2, it corresponds to three cycles. If the two edges are incident to the same vertex of the middle cycle (because it is the head of one arc and the tail of another arc), then the three cycles are merged as shown by Figure 4.c. Otherwise, this case can be handled as a special case of either Figure 4.a or Figure 4.b.

In the case of Figure 4.b, the non-edge may be of length 5. However, since the diameter of G is at most 4, the non-edge has length at most equal to the diameter of the graph. In the case of Figure 4.c, one of the non-edges is of length at most 4 (3, if $\mathrm{diam}(G) = 3$). Also, in case of Figure 4.a, the non-edges are of length at most 3. Then, if the graph has diameter 3, all the cycles can be patched together at no additional cost, because they contain at least one edge of length 3. Otherwise, we have to add 1 to the cost of the merging of Figure 4.a. Then, we can patch all the cycles together at no additional cost.

Lemma 16. *The resultant tour is of length at most* $\frac{17}{12}n$ *(at most* $\frac{4}{3}n$ *if* $\mathrm{diam}(G) = 3$*).*

Proof. The length of the resultant tour is n (the length of the 2-matching) plus the cost of all mergings. The extra cost of the merging of Figure 4.a is 2. This cost is charged to the vertices of the cycle l_i (at least 4), to v_i, and to the neighbour of v_i in l, which is not charged by any other merging. Therefore, each vertex is charged with at most $\frac{1}{3}$. The merging of Figure 4.b costs 3 (2 if the diameter of G is 3), and the vertices of the cycles l_i and l_j are charged. Thus, each vertex is charged with at most $\frac{3}{8}$ ($\frac{1}{4}$, if $\mathrm{diam}(G) = 3$). Finally, the merging of Figure 4.c costs 4 (3, if $\mathrm{diam}(G) = 3$) and at least 12 vertices are involved. Each vertex is charged with at most $\frac{1}{3}(\frac{1}{4})$. Clearly, no vertex is charged by two mergings.

If the diameter of G is 3, the resultant cycles can be patched together at no additional cost. Since, each vertex is charged with at most $\frac{1}{3}$, the bound on the length of the tour follows.

If the diameter of G is 4, two cycles can be patched together at no additional cost, only if they both contain edges of length 4. Thus, we have to add 1 to the cycles produced by the merging 4.a.

Let the cycles l and $l_1, \ldots, l_k$ form an in-tree, such that all mergings fall in case 4.a. If $k = 1$, then at least 8 vertices are involved, and the total cost is 3. Otherwise, at least $4k + 2k = 6k$ vertices are involved, and the total cost is $2k + 1$. Obviously, each vertex is charged with at most $\frac{5}{12}$, and the claim about the length of the tour follows. □

Next, we consider the case that G is not Hamiltonian, and $\mathrm{TSP}_G > n$. As before, we start with an optimum 2-matching, we form the bipartite graph B, and we find a maximum matching in B. Since G is not Hamiltonian, this matching may not cover all the cycles. Again, we construct the directed graph $D(L, A)$, with $(l, l') \in A$, whenever l is matched to a vertex of l'. In this case, the graph D may be a partial function, i.e. the unmatched cycles have outdegree 0. We can decompose D as indicated by Lemma 15. The resultant spanning subgraph consists of in-trees of depth 1, paths of length 2, and some trivial components (isolated vertices), which are unmatched cycles. We merge the cycles of the non-trivial components as before. The produced cycles can be patched together to one large cycle l_1 at no additional cost. Let n_1 be the number of vertices that are included in l_1. The analysis of Lemma 16 implies that each vertex of l_1 is charged with at most $\frac{5}{12}$ ($\frac{1}{3}$, if $\mathrm{diam}(G) = 3$).

Let I be the set of cycles associated with the trivial components of the decomposition of D. Let l, l' be any cycles in I. Obviously, there do not exist 1-edges that join a vertex of l to a vertex of l', because the members of I are unmatched cycles. Since the graph G is connected, any cycle $l \in I$ is joined by an 1-edge to the cycle l_1 produced by the non-trivial components of D. If we merge two cycles connected by an edge of length 1, the additional cost is at most 2 (Figure 3). Thus, we can merge all the isolated cycles with l_1 at a cost at most $2|I|$.

Let $v_1, \ldots, v_n, v_1$ be an optimum tour and U be the set of vertices v_i such that the edge $\{v_i, v_{i+1}\}$ has length at least 2 (non-edge), v_i belongs to a cycle l, and v_{i+1} belongs to a different cycle l'. Let z be the number of cycles that contain a vertex from U. Obviously, the optimum tour has length at least $n + z$. Moreover, given the optimum tour, we can easily obtain a matching in B with at most z cycles unmatched. Therefore, $|I| \leq z$. Additionally, if $n_2 = n - n_1$ is the number of vertices belonging to the isolated cycles of I, then $|I| \leq \frac{n_2}{4}$.

Since the length of the 2-matching is n, the total length of the resultant tour H_G is:

$$\begin{aligned} l(H_G) &\leq n + \frac{5}{12}n_1 + 2|I| \\ &\leq n + \frac{5}{12}n_1 + \frac{5}{12}n_2 + \frac{1}{3}z \\ &\leq \frac{17}{12}\mathrm{TSP}_G \end{aligned}$$

It is straight forward that, if the diameter of G is 3, the same arguments provide a ratio of $\frac{7}{5}$. □

If the input graph does not contain a perfect, simple, triangle-free, 2-matching, we only know how to compute (in polynomial time) an optimum perfect, simple 2-matching, that may contain triangles. Consequently, a careful analysis is needed so as to obtain an improved approximation algorithm for the graphs of diameter 3, that do not contain a perfect, triangle-free 2-matching.

Theorem 17. *Graph TSP restricted to graphs of diameter 3 is approximable in polynomial time within a factor of* $\frac{22}{15}$.

Proof. We assume that $G(V, E)$, $|V| = n$, is a connected graph that does not contain a perfect, simple, triangle-free 2-matching. Otherwise, we can use the $\frac{7}{5}$-approximation algorithm of Theorem 14. Let $d_G(v, u)$ be the length of the shortest path between a vertex pair v and u in G, and $\hat{G}$ be a complete weighted graph on vertex set V, such that, for all vertex pairs $v, u \in V$, $w(v, u) = W - d_G(v, u)$. We start with computing an optimum weighted, simple 2-matching in $\hat{G}$ (polynomial time algorithms are described in [Har84, PM94]).

Proposition 18. *If $W \geq \operatorname{diam}(G)+1$, then any maximum weighted, simple 2-matching of $\hat{G}$ is also perfect.*

Proof. Let $M_2^\star$ be a maximum weighted, simple 2-matching of $\hat{G}$, and $V_1 \subseteq V$ be a set of vertices that does not belong to a cycle of $M_2^\star$. Clearly, if $V_1 = \emptyset$, then we are done. It is easy to verify that if $|V_1| \geq 3$, then we can connect the corresponding connected components to a cycle by adding edges/weight to $M_2^\star$. Moreover, this can be performed without creating any cycles of length 2.

By connectivity of G, any vertex $v \in V_1$, that is isolated in $M_2^\star$, is connected by an edge $e_v \in E$ to a cycle l of $M_2^\star$. Therefore, if we use e_v for merging v with l, the weight of $M_2^\star$ increases by at least $W-2$. If $v, w \in V_1$ are matched to each other by an edge of E, then at least one of them (wlog. v) is connected by an edge $e_v \in E$ to a cycle l of $M_2^\star$. Therefore, if we use e_v for merging v and w with l, the weight of $M_2^\star$ increases by at least $W - \operatorname{diam}(G)$. □

Let an optimum perfect, simple 2-matching of G that contains n_2 edges of length 2, and n_3 edges of length 3. Obviously, such a matching covers all the vertices of V with vertex disjoint, simple cycles. A cycle is called *non-pure*, if it contains an edge of length more than 1. Otherwise, it is called *pure*. Additionally, $\mathrm{TSP}_G \geq n + n_2 + 2n_3$. Clearly, if two non-pure cycles contain 3-edges, then they can be patched together at no additional cost. The same holds, if an 1-edge connects a vertex of a pure cycle with an end vertex of a 3-edge of a non-pure cycle. Therefore, we can assume that we start from a 2-matching with the following properties: (i) all the 3-edges are included in a single non-pure cycle l_3, and (ii) no 1-edge joins an end vertex of a 3-edge of l_3 to a vertex of a pure cycle.

Again, we form the bipartite graph B, but the first vertex class of B only contains vertices corresponding to the pure cycles. As before, we find a maximum cardinality matching in B, but since G is not Hamiltonian, such a matching may not cover all the pure cycles. Similarly to the non-Hamiltonian case of Theorem 14, we construct the directed graph D, which may be a partial function. Again, we decompose D as indicated by Lemma 15 into in-trees of depth 1, simple paths of length 2, and isolated vertices. We also merge the cycles in the non-trivial connected components as in Figure 4, but we amortize the merging costs in a different way.

Notice that some pure cycles of L may be matched to a vertex of a non-pure cycle. However, a non-pure cycle cannot be the tail of an arc, and the proof of Lemma 15 implies that a non-pure cycle cannot be involved in a path of length 2. Therefore, a non-pure cycle can be either the root of an in-tree, or an isolated vertex.

In case of a path of length 2, the merging cost is amortized to all the vertices involved, and each vertex is charged with at most $\frac{4}{10}$. In case of Figure 4.b, the additional cost is 2, and it is amortized to the vertices of the cycles l_i and l_j (at least 6). In case of Figure 4.a, the additional cost is 2. Each vertex of the cycle l_i is charged with $\frac{4}{9}$, and the vertices v and v_i are charged with at most $\frac{1}{3}$.

The cycles obtained from the non-trivial components of D, and the non-pure cycle l_3 can be merged to a single cycle l_1 at no additional cost, because each of them contains at least one 3-edge. Let I_2 be the set of the remaining isolated non-pure cycles, and I_1 be the set of the remaining isolated pure cycles. Each cycle $l \in I_2$ can be merged to l_1 at a cost at most 1, because it contains at least one edge of length 2. This cost is amortized to the vertices of l (at least 3). Each cycle of I_1 can be merged to the current partial tour at an additional cost at most 2, because it is connected by an 1-edge to a vertex of the partial tour.

Notice that, if the non-pure cycle l_3 is the root of an in-tree of D, then the property (ii) implies that each vertex charged is preceded by an edge of length at most 2. If l_3 is isolated in D, then it is merged at no additional cost. Therefore, at least n_3 vertices do not charged at all. Furthermore, the remaining vertices of the non-pure cycles (at least n_2) are charged with at most $\frac{1}{3}$. Let n_1 be the number of the vertices of the isolated pure cycles of I_1. Clearly, at most $n' = n - n_1 - n_2 - n_3$ vertices are charged with at most $\frac{4}{9}$, by the mergings of the cycles associated with the non-trivial components of D.

The cardinality of I_1 cannot be more than $\frac{n_1}{3}$. Additionally, if U and z are defined as in the proof of Theorem 14, then $|I_1| \leq z$, and $\mathrm{TSP}_G \geq n + \max\{n_2 + 2n_3, z\}$. Clearly, the length of the resultant tour H_G is:

$$
\begin{aligned}
l(H_G) &\leq n + n_2 + 2n_3 + \frac{4}{9}n' + \frac{n_2}{3} + 2|I_1| \\
&\leq n + n_2 + 2n_3 + \frac{4}{9}n' + \frac{n_2}{3} + \frac{7}{15}n_1 + \frac{9}{15}z \\
&\leq n + \frac{7}{15}(n_1 + n_2 + n_3 + n') + n_2\left(1 + \frac{1}{3} - \frac{7}{15}\right) + 2n_3\left(1 - \frac{7}{30}\right) + \frac{9}{15}z \\
&\leq \frac{22}{15}n + \frac{13}{15}n_2 + \frac{23}{30}(2n_3) + \frac{9}{15}z \\
&\leq \frac{22}{15}n + \frac{13}{15}(n_2 + 2n_3) + \frac{9}{15}z \\
&\leq \frac{22}{15}(n + \max\{n_2 + 2n_3, z\})
\end{aligned}
$$

□

Remark. It is not hard to verify that Proposition 18 also holds for maximum weighted, simple, triangle-free 2-matchings. In [Har84], a polynomial time algorithm for the weighted case of maximum simple, triangle-free 2-matching was conjectured (see also [CW97]). If this conjecture is true, we can extend the proof of Theorem 14 so as to obtain a $\frac{17}{12}(\frac{7}{5})$-approximation algorithm for TSP in graphs with diameter 4(3).

6 Conclusions

An interesting open problem is to consider the approximability of dense instances of Graph TSP comparing with the approximability of general instances. The proof of Theorem 1 only implies that dense instances of Graph TSP are essentially as hard to approximate as general instances of TSP(1,2). Hence, it may be possible for dense instances of Graph TSP to be approximated within factors less than $\frac{3}{2}$.

The major obstacle in obtaining an approximation preserving reduction from general instances of Graph TSP to dense ones is to relate the distances on the resultant graph with the distances on the original graph. In particular, some vertex pairs may be very close in the resultant dense graph, while they are far apart in the original one. If we try to start from graphs of constant diameter (because a dense graph always has a constant diameter), it seems that we have to solve a problem similar to PARTITION INTO CLIQUES, which is $\mathcal{NP}$-complete.

On the other hand, we prove that a variant of the algorithm of Papadimitriou and Yannakakis for TSP(1,2) achieves an improved approximation ratio for TSP in graphs of very small diameter. However, this algorithm requires some additional ideas so as to be applied to dense instances of Graph TSP.

Another direction for further research is the conjecture that, given a graph $G(V, E)$ and a partition of V into κ cliques, G is Hamiltonian iff there exists a Hamiltonian Cycle containing at most $2(\kappa - 1)$ inter-clique edges. In this paper, we prove the conjectured bound for graphs and partitions that contain at least one Hamiltonian Cycle H, such that any vertex of V has degree at most 1 in the set of inter-clique edges of H. Moreover, we show that the conjectured bound is equivalent to a generalized version of Smith's Theorem on the number of Hamiltonian Cycles contained by cubic graphs.

References

[Aro96] S. Arora. Polynomial Time Approximation Schemes for Euclidean TSP and Other Geometric Problems. *Proc. of the 37th IEEE Symposium on Foundations of Computer Science*, pp. 2–12, 1996.

[AGKKW98] S. Arora, M. Grigni, D. Karger, P. Klein, and A. Woloszyn. Polynomial Time Approximation Scheme for Weighted Planar Graph TSP. To appear in *Proc. of the 9th ACM-SIAM Symposium on Discrete Algorithms*, 1998.

[ALMSS92] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and hardness of approximation problems. *Proc. of the 33th Annual IEEE Symposium on Foundations of Computer Science*, pp. 14–23, 1992.

[Ber85] C. Berge. *Graphs* (second edition). North Holland, 1985.

[Chr76] N. Christofides. Worst-Case Analysis of a New Heuristic for the Travelling Salesman Problem. J.F. Traub (editor) *Symposium on new directions and recent results in algorithms and complexity*, page 441, 1976.

[CW97] W.H. Cunningham and Y. Wang. Restricted 2-Factor Polytopes. A preliminary version is available from `http://math.uwaterloo.ca/%7Ewhcunnin/preprint.html`, 1997.

[DSST97] J. Díaz, M.J. Serna, P. Spirakis, and J. Torán. *Paradigms for Fast Parallel Approximability.* Cambridge University Press, 1997.

[FVK98] W. Fernandez de la Vega and M. Karpinski. On Approximation Hardness of Dense TSP and other Path Problems. Available from `http://cs.uni-bonn.de/info5/publications/CS-1998-en.html`, 1998.

[GJ79] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of $\mathcal{NP}$-Completeness.* Freeman, San Francisco, 1979.

[GR88] A. Gibbons and W. Rytter. *Efficient Parallel Algorithms.* Cambridge University Press, 1988.

[Goe95] M. Goemans. Worst-case Comparison of Valid Inequalities for TSP. *Mathematical Programming* **69**, pp. 335-349, 1995.

[GKP95] M. Grigni, E. Koutsoupias, and C.H. Papadimitriou. An Approximation Scheme for Planar TSP. *Proc. of the 36th IEEE Symposium on Foundations of Computer Science*, pp. 640-645, 1995.

[Har84] D.B. Hartvigsen. *Extensions of Matching Theory.* PhD Thesis, Carnegie-Mellon University, 1984.

[IS86] A. Israeli and Y. Shiloach. An improved algorithm for maximal matching. *Information Processing Letters* **33**, pp. 57–60, 1986.

[Kar97] M. Karpinski. Polynomial Time Approximation Schemes for Some Dense Instances of $\mathcal{NP}$-hard Optimization Problems. Invited Talk in *School and Workshop on Randomized Algorithms in Sequential, Parallel, and Distributed Computing*, October 1997. See also *Proc. of the 1st Symposium on Randomization and Approximation Techniques in Computer Science*, pp. 1–14, 1997.

[KMSV94] S. Khanna, R. Motwani, M. Sudan, and U. Vazirani. On Syntactic versus Computational Views of Approximability. *Proc. of the 35th IEEE Symposium on Foundations of Computer Science*, pp. 819–830, 1994.

[LLRS85] E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, and D.B. Shmoys. *The Travelling Salesman Problem: A Guided Tour of Combinatorial Optimization.* John Wiley, 1985.

[PY91] C.H. Papadimitriou and M. Yannakakis. Optimization, Approximation and Complexity Classes. *Journal of Computer and System Sciences* **43**, pp. 425–440, 1991.

[PY93] C.H. Papadimitriou and M. Yannakakis. The Traveling Salesman Problem with Distances One and Two. *Mathematics of Operations Research* **18**(1), pp. 1–11, 1993.

[PM94] J.F. Pekny and D.L. Miller. A Staged Primal-Dual Algorithm for Finding a Minimum Cost Perfect Two-Matching in an Undirected Graph. *ORSA Journal on Computing* **6**(1), pp. 68–81, 1994.