Theoretical Computer Science 1020 (2024) 114836

Contents lists available at ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

# Approximations for the Steiner Multicycle problem ☆

Cristina G. Fernandes [a], Carla N. Lintzmayer [b,*], Phablo F.S. Moura [c]

[a] Department of Computer Science, University of São Paulo, São Paulo, Brazil
[b] Center for Mathematics, Computing and Cognition, Federal University of ABC, Santo André, São Paulo, Brazil
[c] Research Center for Operations Research & Statistics, KU Leuven, Belgium

ARTICLE INFO

*Keywords:*
Combinatorial optimization
Approximation algorithms
Steiner problems
Traveling salesman problem
Collaborative logistics

ABSTRACT

The Steiner Multicycle problem consists in, given a complete graph, a weight function on its edges, and a collection of pairwise disjoint non-unitary sets called terminal sets, finding a minimum weight collection of vertex-disjoint cycles in the graph such that, for every terminal set, all of its vertices are in a same cycle of the collection. This problem generalizes the Traveling Salesman problem and, therefore, is hard to approximate in general. On the practical side, it models a collaborative less-than-truckload problem with pickup and delivery locations. Using an algorithm for the Survivable Network Design problem and $T$-joins, we obtain a 3-approximation for the metric case, improving on the previous best 4-approximation. Furthermore, we present an (11/9)-approximation for the particular case of the Steiner Multicycle in which each edge weight is 1 or 2. This algorithm can be adapted to obtain a (7/6)-approximation when every terminal set contains at least four vertices. Finally, we devise an $O(\lg n)$-approximation algorithm for the asymmetric version of the problem.

## 1. Introduction

In the Steiner Multicycle problem, one is given a complete graph $G$, a weight function $w\colon E(G) \to \mathbb{Q}_+$, and a collection $\mathcal{T} \subseteq \mathcal{P}(V(G))$ of pairwise disjoint non-unitary sets of vertices, called *terminal sets*. We say that a cycle $C$ *respects* $\mathcal{T}$ if, for all $T_i \in \mathcal{T}$, either every vertex of $T_i$ is in $C$ or no vertex of $T_i$ is in $C$, and a set $\mathcal{C}$ of vertex-disjoint cycles *respects* $\mathcal{T}$ if all cycles in $\mathcal{C}$ respect $\mathcal{T}$ and every vertex in a terminal set is in some cycle of $\mathcal{C}$. The cost of such set $\mathcal{C}$ is the sum of the edge weights over all cycles in $\mathcal{C}$, a value naturally denoted by $w(\mathcal{C})$. The goal of the Steiner Multicycle problem is to find a set of vertex-disjoint cycles of minimum cost that respects $\mathcal{T}$. We denote by $\mathrm{opt}(G, w, \mathcal{T})$ the cost of such a minimum cost set. Note that the number of cycles in a solution might be smaller than $|\mathcal{T}|$, that is, it might be cheaper to join some terminal sets in the same cycle.

We consider that, in a graph $G$, a cycle is a non-empty connected subgraph of $G$ all of whose vertices have degree two. Consequently, such cycles have at least three vertices. Here, as a set $T_i \in \mathcal{T}$ can have only two vertices, we would like to consider a single edge as a cycle, of length two, whose cost is twice the weight of the edge, so that the problem also includes solutions that choose to connect some set from $\mathcal{T}$ with two vertices through such a length-2 cycle. So, for each set $T_i \in \mathcal{T}$ with $|T_i| = 2$, we duplicate in $G$ the edge linking the vertices in $T_i$, and allow the solution to contain length-2 cycles.

☆ This article belongs to Section A: Algorithms, automata, complexity and games, Edited by Paul Spirakis.

* Corresponding author.
*E-mail addresses:* cris@ime.usp.br (C.G. Fernandes), carla.negri@ufabc.edu.br (C.N. Lintzmayer), phablo.moura@kuleuven.be (P.F.S. Moura).

https://doi.org/10.1016/j.tcs.2024.114836
Received 8 November 2023; Received in revised form 27 May 2024; Accepted 2 September 2024

Available online 6 September 2024

The Steiner Multicycle problem is a generalization of the Traveling Salesman problem (TSP), thus it is NP-hard and its general form admits the same inapproximability results as the TSP. It was proposed by Pereira et al. [22] as a generalization of the so-called Steiner Cycle problem (see Salazar-González [24]), with the assumption that the graph is complete and the weight function satisfies the triangle inequality. We refer to such an instance of the Steiner Multicycle problem as *metric*, and to the problem restricted to such instances as the Metric Steiner Multicycle problem.

Pereira et al. [22] presented a 4-approximation algorithm for the Metric Steiner Multicycle problem, designed Refinement Search and GRASP-based heuristics, and proposed an integer linear programming formulation for the problem. Lintzmayer et al. [17] then considered the version restricted to the Euclidean plane and presented a randomized approximation scheme for it, which combines some techniques for the Euclidean TSP [2] and for the Euclidean Steiner Forest [5].

Xu and Rodrigues [30] designed a $\frac{3}{2}$-approximation for a related generalization of TSP, namely Multidepot TSP. Given a complete graph $G$, a function $w\colon E(G) \to \mathbb{Q}_+$ satisfying the triangle inequality, and a set $D \subset V(G)$ with $k$ vertices (a.k.a. depots), this problem consists in finding a minimum-weight collection of $k$ vertex-disjoint cycles such that each of them contains exactly one vertex in $D$. In contrast to Multidepot TSP, a solution to a Steiner Multicycle instance may be a single cycle containing all pairs of terminals.

On the practical side, the Steiner Multicycle problem models a collaborative less-than-truckload problem with pickup and delivery locations. In this scenario, several companies operating in the same geographic regions must periodically transport products between different locations. To reduce the costs of transporting their goods, these companies may collaborate to create routes for shared cargo vehicles that visit the places defined by them for the collection and delivery of their products (see Ergun et al. [9,10]).

This paper addresses three variations of the Steiner Multicycle. The first is the metric case, for which we present a 3-approximation, improving on the previously best known. The proposed algorithm uses an approximate solution $S$ for a derived instance of the Survivable Network Design problem and a minimum weight $T$-join in $S$, where $T$ is the set of odd-degree vertices in $S$. The second one is the so-called $\{1,2\}$-Steiner Multicycle problem, in which the weight of each edge is either 1 or 2. Note that this is a particular case of the metric one, and it is a generalization of the $\{1,2\}$-TSP, therefore it is also APX-hard [21]. In some applications, there might be little information on the actual cost of the connections between points, but there might be at least some distinction between cheap connections and expensive ones. These situations could be modeled as instances of the $\{1,2\}$-Steiner Multicycle. For this variation, we design an $\frac{11}{9}$-approximation following the strategy for the $\{1,2\}$-TSP proposed by Papadimitrou and Yannakakis [21]. The third variation is the asymmetric case, in which one is now given a complete digraph in which the weight of an arc $(u,v)$ is not necessarily the same as the weight of the arc $(v,u)$, but the weights still satisfy the triangle inequality. For this case, we design an $O(\lg n)$-approximation algorithm, where $n$ is the number of vertices in the graph, following some ideas for the Asymmetric TSP proposed by Frieze, Galbiati, and Maffioli [12].

Note that the three variations we consider are metric. In this case, we assume that the terminal sets partition the vertex set. Indeed, because the graph (or digraph) is complete and the weight function is metric, any solution containing non-terminal vertices does not have its cost increased by *shortcutting* these vertices, that is, removing them and adding the edge linking their neighbors in the cycle. Under this assumption, the set of cycles of any solution is a *2-factor that respects* the terminal sets. A *2-factor* is a set of vertex-disjoint cycles that spans all vertices of the graph.

A preliminary version of this paper was published in the LATIN 2022 proceedings [11]. In addition to the results presented there, this manuscript contains a new algorithm for the asymmetric version of the problem, improved proofs, more examples, and a detailed discussion on minimum weight triangle-free 2-factors.

The 3-approximation for the Metric Steiner Multicycle is presented in Section 2, together with a discussion involving the previous 4-approximation and the use of perfect matchings on the set of odd degree vertices of intermediate structures. The $\{1,2\}$-Steiner Multicycle problem is addressed in Section 3. The asymmetric case is investigated in Section 4, and we make some final considerations in Section 5.

## 2. Metric Steiner Multicycle problem

An instance for the Steiner Multicycle is also an instance for the well-known Steiner Forest problem [29, Chapter 22], but the goal in the latter is to find a minimum weight forest in the graph that connects vertices in the same terminal set, that is, every terminal set is in some connected component of the forest. The optimum value of the Steiner Forest is a lower bound on the optimum for the Steiner Multicycle: one can produce a feasible solution for the Steiner Forest from an optimal solution for the Steiner Multicycle by throwing away one edge in each cycle without increasing its cost.

The existing 4-approximation [22] for the metric Steiner Multicycle problem is inspired by the famous 2-approximation for the metric TSP [23], and it consists in doubling the edges in a Steiner forest for the terminal sets, finding an Eulerian trail in each of its components, and shortcutting[1] these trails to obtain cycles. As there are 2-approximations for the Steiner Forest problem, this leads to a 4-approximation.

It is tempting to try to use a perfect matching on the odd-degree vertices of the approximate Steiner forest solution, as Christofides' algorithm [6] does to achieve a better ratio for the Metric TSP. However, the best upper bound we can prove so far on such a matching is the weight of the approximate Steiner forest solution, which implies that such a matching weights at most twice the optimum. With this bound, we also derive an approximation algorithm with ratio at most 4.

[1] To shortcut a walk or a trail means to remove repeated vertices from them to generate a path or a cycle; this is possible in complete graphs and it can only decrease the costs for metric weight functions.

Another problem that can be used with this approach is known as the SURVIVABLE NETWORK DESIGN problem [29, Chapter 23]. An instance of this problem consists of the following: a graph $G$, a weight function $w\colon E(G) \to \mathbb{Q}_+$, and a non-negative integer $r_{ij}$ for each pair of vertices $i, j$ with $i \neq j$, representing a connectivity requirement. The goal is to find a minimum weight subgraph $G'$ of $G$ such that, for every pair of vertices $i, j \in V(G)$ with $i \neq j$, there are at least $r_{ij}$ edge-disjoint paths between $i$ and $j$ in $G'$.

From an instance of the STEINER MULTICYCLE problem, we can naturally define an instance of the SURVIVABLE NETWORK DESIGN problem: set $r_{ij} = 2$ for every two vertices $i, j$ in the same terminal set, and set $r_{ij} = 0$ otherwise. As all vertices are terminals, all connectivity requirements are defined in this way. The optimum value of the SURVIVABLE NETWORK DESIGN problem is also a lower bound on the optimum for the STEINER MULTICYCLE problem: indeed an optimal solution for the STEINER MULTICYCLE problem is a feasible solution for the SURVIVABLE NETWORK DESIGN problem with the same cost.

There also exists a 2-approximation for the SURVIVABLE NETWORK DESIGN problem [16]. By applying the same approach of the 2-approximation for the metric TSP, of doubling edges and shortcutting, we achieve a ratio of 4 for the metric STEINER MULTICYCLE again. However, next, we will show that one can obtain a 3-approximation for the metric STEINER MULTICYCLE problem, from a 2-approximate solution for the SURVIVABLE NETWORK DESIGN problem, using not a perfect matching on the odd degree vertices of such solution, but the related concept of $T$-joins.

### 2.1. A 3-approximation algorithm for the metric case

Let $T$ be a set of vertices of even size in a graph $G$. A set $J$ of edges in $G$ is a *$T$-join* if the collection of vertices of $G$ that are incident to an odd number of edges in $J$ is exactly $T$. Any perfect matching on the vertices of $T$ is a $T$-join, so $T$-joins are, in some sense, a generalization of perfect matching on a set $T$. It is known that a $T$-join exists in $G$ if and only if the number of vertices from $T$ in each component of $G$ is even. Moreover, there are polynomial-time algorithms that, given a connected graph $G$, a weight function $w\colon E(G) \to \mathbb{Q}_+$, and an even set $T$ of vertices of $G$, find a minimum weight $T$-join in $G$. For these and more results on $T$-joins, we refer the reader to the book by Schrijver [25, Chapter 29].

The idea of our 3-approximation is similar to Christofides [6]. It is presented in Algorithm 1. Let $(G, w, \mathcal{T})$ be a metric instance of the STEINER MULTICYCLE problem. The first step is to build the corresponding SURVIVABLE NETWORK DESIGN problem instance and to obtain a 2-approximate solution $G'$ for this instance. The procedure 2APPROXSND represents the algorithm by Jain [16] for the SURVIVABLE NETWORK DESIGN. The second step considers the set $T$ of the vertices in $G'$ of odd degree and finds a minimum weight $T$-join $J$ in $G'$. The procedure MINIMUMTJOIN represents the algorithm by Edmonds and Johnson [8] for this task. Note that adding $J$ to $G'$ creates an Eulerian multigraph $H$, because all vertices now have even degree. This is represented as $G' + J$ in Algorithm 1. We then use the procedure SHORTCUT to apply shortcuts over the Eulerian tour in $H$ and find a 2-factor $C$ in $G$, which is the output of the algorithm.

**Algorithm 1** STEINERMULTICYCLEAPPROX_METRIC($G$, $w$, $\mathcal{T}$).

**Input:** a complete graph $G$, a weight function $w\colon E(G) \to \mathbb{Q}_+$ satisfying the triangle inequality, and a partition $\mathcal{T} = \{T_1, \ldots, T_k\}$ of $V(G)$
**Output:** a 2-factor $C$ in $G$ that respects $\mathcal{T}$
1: $r_{ij} \leftarrow 2$ for every $i, j \in T_a$ for some $1 \le a \le k$
2: $r_{ij} \leftarrow 0$ for every $i \in T_a$ and $j \in T_b$ for $1 \le a < b \le k$
3: $G' \leftarrow$ 2APPROXSND($G$, $w$, $r$)
4: Let $T$ be the set of odd-degree vertices in $G'$
5: Let $w'$ be the restriction of $w$ to the edges in $G'$
6: $J \leftarrow$ MINIMUMTJOIN($G'$, $w'$, $T$)
7: $H \leftarrow G' + J$
8: $C \leftarrow$ SHORTCUT($H$)
9: **return** $C$

Because the number of vertices of odd-degree in any connected graph is even, the number of vertices with odd degree in each component of $G'$ is even. Therefore there is a $T$-join in $G'$. Moreover, the collection $C$ produced by Algorithm 1 is indeed a feasible solution for the STEINER MULTICYCLE.

Next, we prove that the proposed algorithm is a 3-approximation.

**Theorem 1.** *Algorithm 1 is a 3-approximation for the METRIC STEINER MULTICYCLE problem.*

**Proof.** First, it suffices to prove that $w(J) \le \frac{1}{2} w(G')$. Indeed, because $G'$ is a 2-approximate solution for the SURVIVABLE NETWORK DESIGN problem, and the optimum for this problem is a lower bound on $\mathrm{opt}(G, w, \mathcal{T})$, we have that $w(G') \le 2\,\mathrm{opt}(G, w, \mathcal{T})$. Hence we deduce that $w(J) \le \mathrm{opt}(G, w, \mathcal{T})$, and therefore that $w(C) \le w(G') + w(J) \le 3\,\mathrm{opt}(G, w, \mathcal{T})$. We now show that inequality $w(J) \le \frac{1}{2} w(G')$ holds.

A *bridge* is an edge $uv$ in a graph whose removal leaves $u$ and $v$ in different components of the resulting graph. First, observe that we can delete from $G'$ any bridges and the remaining graph, which we still call $G'$, remains a solution for the SURVIVABLE NETWORK DESIGN problem instance. Indeed a bridge is not enough to assure the connectivity requirement between two vertices in the same terminal set, so it will not separate any such pair of vertices, and hence it can be removed. In other words, we may assume that each component of $G'$ is 2-edge-connected.

Edmonds and Johnson [8] gave an exact description of a polyhedra related to $T$-joins. This description will help us to prove the claim. For a set $S$ of edges in a graph $(V, E)$, let $\nu(S)$ denote the corresponding $|E|$-dimensional incidence vector (with 1 in the $i$-th coordinate if edge $i$ lies in $S$ and 0 otherwise). For a set $X$ of vertices, let $\delta(X)$ denote the set of edges with one endpoint in $X$ and the other in $V \setminus X$. An *upper $T$-join* is any superset of a $T$-join. Let $P(G,T)$ be the convex hull of all vectors $\nu(J)$ corresponding to the incidence vector of upper $T$-joins $J$ of a graph $G = (V, E)$. The set $P(G,T)$ is called the *up-polyhedra of $T$-joins*, and it is described by

$$\sum_{e \in \delta(W)} x(e) \geq 1 \text{ for every } W \subseteq V \text{ such that } |W \cap T| \text{ is odd}, \tag{1}$$

$$0 \leq x(e) \leq 1 \text{ for every edge } e \in E. \tag{2}$$

(For more on this, see [25, Chapter 29].)

So, as observed in [3], any feasible solution $x$ to the system of inequalities above can be written as a convex combination of upper $T$-joins, that is, $x = \sum \alpha_i \nu(J_i)$, where $0 \leq \alpha_i \leq 1$ and $\sum_i \alpha_i = 1$, leading to the following.

**Lemma 1** *(Corollary 1 in [3]). If all the weights $w(e)$ are non-negative, then, given any feasible assignment $x(e)$ satisfying the inequalities above, there exists a $T$-join with weight at most $\sum_{e \in E} w(e)x(e)$.*

Recall that, for each component $C$ of $G'$, $|V(C) \cap T|$ is even. Hence, for every $W \subseteq V(G')$ such that $|W \cap T|$ is odd, there must exist a component $C$ of $G'$ with $V(C) \cap W \neq \emptyset$, and $V(C) \setminus W \neq \emptyset$. As a consequence, it holds that $|\delta(W)| \geq 2$ because every component of $G'$ is 2-edge-connected. Consider now the $|E(G')|$-dimensional vector $\bar{x}$ which assigns value $1/2$ to each edge of $G'$. From the discussion above, it is clear that $\bar{x}$ satisfies inequalities (1) and (2) for $G'$ and $T$. Then Lemma 1 guarantees that there is a $T$-join $J$ in $G'$ such that $w(J) \leq \frac{1}{2} w(G')$. This completes the proof of the theorem. □

### 2.2. Matchings, $T$-joins, and Steiner forests

Because $G$ is complete and $w$ is metric, the proof of Theorem 1 in fact implies that a minimum weight perfect matching in the graph $G[T]$ weights at most $w(G')/2$, and therefore at most $\text{opt}(G, w, \mathcal{T})$. However, we have no direct proof for this fact; only this argument that goes through a minimum weight $T$-join. But this fact means that one can exchange line 6 to compute, instead, a minimum weight perfect matching $J$ in $G[T]$.

We investigated the possibility that one could achieve a ratio of 3 using a Steiner forest instead of a survivable network design solution. However, using a $T$-join does not work so well with the Steiner forest, once its components are not 2-edge-connected. Indeed, if $T$ is the set of odd-degree vertices in a Steiner forest $F$, a bound as in the proof of Theorem 1 on a minimum weight $T$-join in $F$ would not hold in general: there are examples for which such a $T$-join in $F$ has weight $w(F)$.

Let $\text{opt}_{\text{SND}}$ denote the optimum value for the Survivable Network Design instance used in Algorithm 1, and $\text{opt}_{\text{SF}}$ denote the optimum value for the Steiner Forest instance used in the 4-approximation from the literature [22]. Let $\text{opt}_{\text{SMC}}$ be the Steiner Multicycle optimum value. Note that $\text{opt}_{\text{SF}} \leq \text{opt}_{\text{SND}} \leq \text{opt}_{\text{SMC}} \leq 2\,\text{opt}_{\text{SF}}$, where the last inequality holds because a duplicated Steiner forest solution leads to a cheaper feasible solution for the Survivable Network Design and the Steiner Multicycle instances. Let $G'$ and $J$ be the subgraph and the $T$-join used in Algorithm 1, respectively, and let $M$ be a minimum weight perfect matching in $G[T]$. Then $w(M) \leq w(J) \leq \frac{1}{2} w(G') \leq \text{opt}_{\text{SND}} \leq w(G')$. (For the first inequality, recall that $J$ is a $T$-join in $G'$ while $M$ is a minimum weight perfect matching in $G[T]$.) If $T'$ is the set of odd-degree vertices in an optimal Steiner forest and $M'$ is a minimum weight perfect matching in $G[T']$, then $w(M') \leq 2\text{opt}_{\text{SF}}$, and there are instances for which this upper bound is tight. So, as far as we know, there might be an instance where $w(M') > \text{opt}_{\text{SMC}}$. Even if this is not the case, in fact, what we can compute in polynomial time is a minimum weight perfect matching $M''$ for the set of odd-degree vertices in a 2-approximate Steiner forest solution, so it would still be possible that $w(M'') > \text{opt}_{\text{SMC}}$ for some instances. We tried to find an instance where this is the case, but we have not succeeded so far.

## 3. {1,2}-Steiner Multicycle problem

In this section, we shall address the particular case of the metric Steiner Multicycle problem that allows only edge weights 1 or 2.

It is a well-known result that there exists a polynomial-time algorithm for finding a 2-factor of minimum weight in weighted graphs [18,28]. Specifically, for a complete graph on $n$ vertices, one can find such a 2-factor by computing a maximum weight perfect matching in a graph with $O(n^2)$ vertices and edges. This can be done in time $O(n^4)$ using Orlin's maximum flow algorithm [20].

The algorithm for this case of the Steiner Multicycle problem starts from a minimum weight 2-factor of the given weighted graph, and then repeatedly joins two cycles until a feasible solution is obtained. Joining two cycles may increase the cost of the initial 2-factor, and so the key to guarantee a good approximation ratio is a clever choice of the cycles to join at each step. To proceed with the details, we need the following definitions.

Let $(G, w, \mathcal{T})$ be an instance of the Steiner Multicycle problem with $w\colon E(G) \to \{1,2\}$. Recall that $G$ is complete and $\bigcup_{T_a \in \mathcal{T}} T_a = V(G)$, and that, for each set $T_a \in \mathcal{T}$ with $|T_a| = 2$, we duplicated in $G$ the edge linking the vertices in $T_a$, to allow the solution to contain length-2 cycles. We say an edge $e \in E(G)$ is an *$i$-edge* if $w(e) = i$, for $i \in \{1,2\}$. A cycle containing only 1-edges is called *pure*; otherwise, it is called *nonpure*.

Let us start with a general description of the process of joining the cycles of a minimum weight 2-factor to produce a feasible solution for the STEINER MULTICYCLE problem. First observe that joining two nonpure cycles of a 2-factor does not increase the cost of the 2-factor: one can always remove a 2-edge from each of the nonpure cycles, and connect them with 2-edges to get a single nonpure cycle. Hence the increase of the cost is due to joins involving pure cycles. The idea of the algorithm is to merge as many of these cycles into nonpure cycles, paying as little as possible. Then one may join the resulting nonpure cycles that still do not respect $\mathcal{T}$ without increasing the cost, as mentioned above, and there will be fewer cycles that do not respect $\mathcal{T}$ to be joined in a final phase. Supposing the existence of a set of 1-edges pairwise connecting exactly all pure cycles that do not respect $\mathcal{T}$, we would be able to join pairs of these cycles and obtain a 2-factor of cost at most the cost of the initial 2-factor plus $n/6$ because each pair of these cycles spans at least 6 vertices. From this 2-factor, by joining nonpure cycles for free, we would get a 2-factor that respects $\mathcal{T}$ of cost at most $7n/6$. Unfortunately, such a set of 1-edges might not exist. However, as we shall see, there is a way to group sets of cycles using 1-edges that allows us to derive a 2-factor that respects $\mathcal{T}$ of cost at most $11n/9$.

All steps of our procedure are summarized in Algorithm 2. In what follows, we explain in details these steps and the auxiliary procedures used in the algorithm.

**Algorithm 2** STEINERMULTICYCLEAPPROX_$\{1,2\}$-WEIGHTS($G$, $w$, $\mathcal{T}$).

**Input:** a complete graph $G$, a weight function $w\colon E(G) \to \{1,2\}$, and a partition $\mathcal{T} = \{T_1, \ldots, T_k\}$ of $V(G)$
**Output:** a 2-factor $C$ in $G$ that respects $\mathcal{T}$
1: $\mathcal{F} \leftarrow$ SPECIAL2FACTOR($G$, $w$)
2: $B \leftarrow$ BUILDBIPARTITEGRAPH($G$, $w$, $\mathcal{T}$, $\mathcal{F}$)
3: $M \leftarrow$ MAXIMUMMATCHING($B$)
4: Let $D$ be a digraph with $V(D) = \mathcal{F}$ and $(C, C') \in E(D)$ if $C$ is matched by $M$ to a vertex of $C'$
5: $D' \leftarrow$ SPECIALSPANNINGDIGRAPH($D$)
6: $C' \leftarrow$ JOINCOMPONENTCYCLES($\mathcal{F}$, $D'$) (details in Section 3.1)
7: $C \leftarrow$ JOINDISRESPECTINGCYCLES($C'$, $D'$, $\mathcal{T}$) (details in Section 3.1)
8: **return** $C$

Procedure SPECIAL2FACTOR finds a minimum weight 2-factor $\mathcal{F}$ of $(G, w)$ with the following properties:

(i) $\mathcal{F}$ contains at most one nonpure cycle; and
(ii) if $\mathcal{F}$ contains a nonpure cycle, no 1-edge in $G$ connects an endpoint of a 2-edge in the nonpure cycle to a pure cycle in $\mathcal{F}$.

Given any minimum weight 2-factor $\mathcal{F}'$, one can construct in polynomial time a 2-factor $\mathcal{F}$ from $\mathcal{F}'$ having properties (i) and (ii) as follows. To ensure property (i), recall that $G$ is complete, we repeatedly join two nonpure cycles by removing one 2-edge from each and adding two appropriate edges that turn them into one cycle. This clearly does not increase the weight of the 2-factor and reduces the number of nonpure cycles. To ensure property (ii), while there is a 1-edge $yz$ in $G$ connecting an endpoint of a 2-edge $xy$ of the nonpure cycle to an endpoint of a 1-edge $wz$ of a pure cycle, we remove $xy$ and $wz$ and add $yz$ and $xw$, reducing the number of cycles without increasing the weight of the 2-factor. The resulting 2-factor is returned by SPECIAL2FACTOR. Finding the desired edges at each step clearly takes polynomial time on $n$, where $n$ is the number of vertices in $G$. Because the initial 2-factor has at most $n/2$ cycles, the amount of steps in this procedure is polynomial on $n$.

To modify $\mathcal{F}$ into a 2-factor that respects $\mathcal{T}$ without increasing too much its weight, Algorithm 2 builds some auxiliary structures that capture how the cycles in $\mathcal{F}$ that do not respect $\mathcal{T}$ attach to each other and to other cycles in $\mathcal{F}$.

The second step of Algorithm 2 is to build a bipartite graph $B$ (line 2) as follows. Let $V(B) = V(G) \cup \{C \in \mathcal{F}\colon C$ be a pure cycle that does not respect $\mathcal{T}\}$ and there is an edge $vC$ in $E(B)$ if $v \notin V(C)$ and $uv$ is a 1-edge for some vertex $u \in V(C)$. Note that length-2 cycles in $\mathcal{F}$ are not vertices of $B$, because they respect $\mathcal{T}$. Procedure MAXIMUMMATCHING in line 3 computes in polynomial time a maximum matching $M$ in $B$ (e.g., using Edmonds' algorithm [7]).

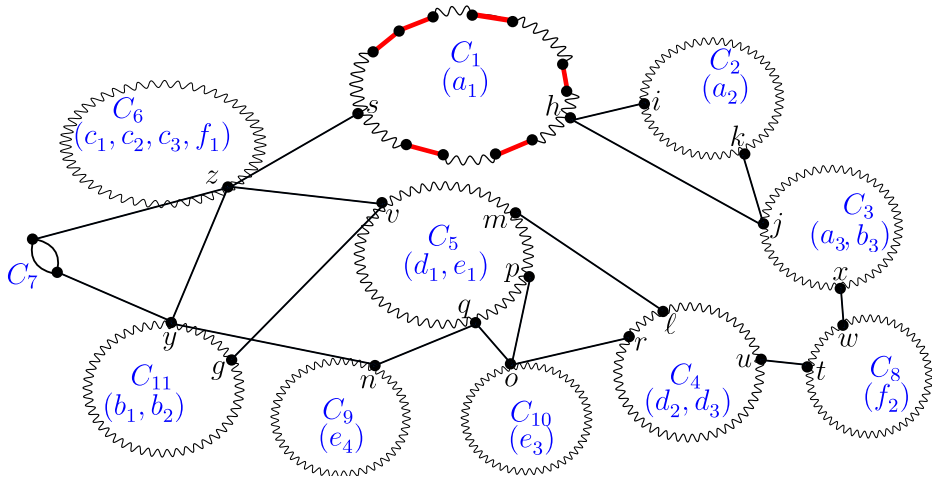
Algorithm 2 then proceeds by building a digraph $D$ where $V(D) = \mathcal{F}$ and there is an arc $(C, C') \in E(D)$ if $C$ is matched by $M$ to a vertex of $C'$. Note that the vertices of $D$ have outdegree 0 or 1: cycles in $B$ matched by $M$ have outdegree 1 while cycles in $B$ unmatched by $M$ and cycles of $\mathcal{F}$ not in $B$ have outdegree 0 in $D$.

An *in-tree of depth $d$* is a digraph obtained from an undirected rooted tree of depth $d$ by orienting every edge towards its root, where the depth of a tree is the number of edges in the longest path between any vertex and the root. Clearly, the root vertex of an in-tree has outdegree 0 and every other vertex has outdegree 1. The vertices of indegree 0 belonging to in-trees are called *leaves*.

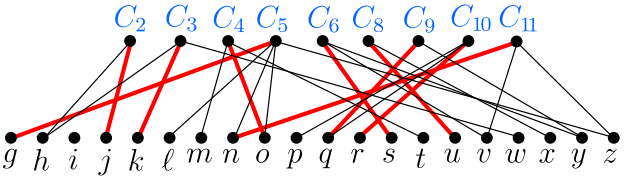
In line 5, Algorithm 2 applies procedure SPECIALSPANNINGDIGRAPH($D$), which returns a minimal spanning digraph $D'$ of $D$ whose nontrivial components span all vertices corresponding to cycles in $B$ matched by $M$. Furthermore, the components of $D'$ are either vertex-disjoint in-trees of depth 1, or length-2 paths, or trivial components. The SPECIALSPANNINGDIGRAPH procedure is similar to one described by Papadimitriou and Yannakakis [21, Lemma 2] and it takes linear time. We adapt their result in Lemma 2 for completeness. See Fig. 1 for an example of these constructions.

**Lemma 2.** *The digraph $D$ contains a minimal spanning subdigraph $D'$ consisting of vertex-disjoint in-trees of depth 1, length-2 paths, and trivial components. Furthermore, the nontrivial components of $D'$ span all vertices corresponding to cycles in $V(B)$ matched by $M$.*

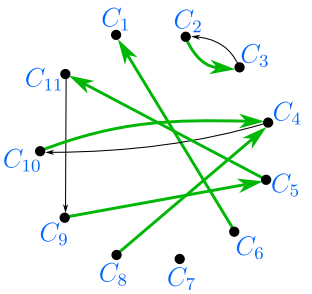
**Proof.** Start with $V(D') = V(D)$. We shall add arcs to $D'$ in a simple greedy manner. The procedure is applied separately to each nontrivial weakly connected component of $D$. Since all vertices of $D$ have outdegree 0 or 1, each such component contains at most




(a) Original graph $G$ and the 2-factor (depicting only 1-edges in black and straight lines, and some 2-edges in red and bold lines; squiggly lines correspond to one or more 1-edges). Inside each $C_i$, in parenthesis, we list some of the terminal vertices it contains.




(b) Bipartite graph $B$ and a matching $M$ highlighted in red and bold. Note that there is no edge incident to $C_{11}$ because it already respects $\mathcal{T}$.




(c) Digraph $D$ and corresponding subgraph $D'$ highlighted in green and bold.

**Fig. 1.** Example of auxiliar graphs and structures built by Algorithm 2.

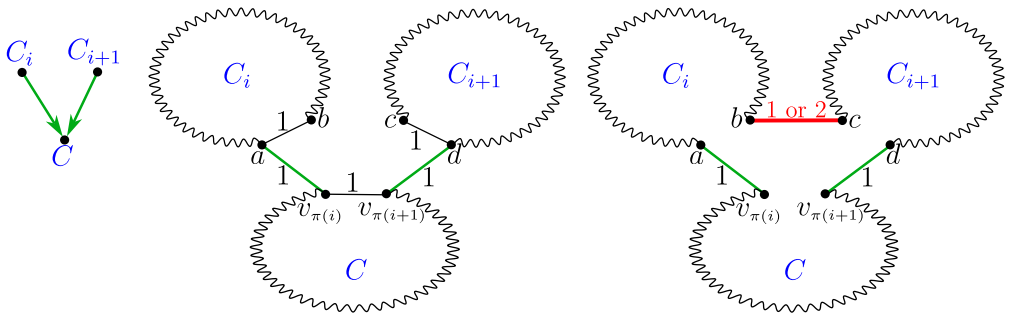
one (directed) cycle, which we will call *circuit*, to avoid confusion with the cycles from the 2-factor. Let $H$ denote a nontrivial weakly connected component of $D$. Then $H$ has at most one circuit.

If $H$ has no circuit, then let $v$ be a vertex in $H$ such that all its in-neighbors in $H$ have indegree 0. Add to $D'$ all arcs of $D$ with head $v$, and notice that they form an in-tree of depth 1, say $T$, rooted at $v$ (even if $T$ is just a path of length 1). Iterate the procedure with $H - V(T)$ playing the role of $H$ until $H$ vanishes or is a singleton, in which case it is not a cycle in $V(B)$ matched by $M$, because its outdegree in $D$ is 0.

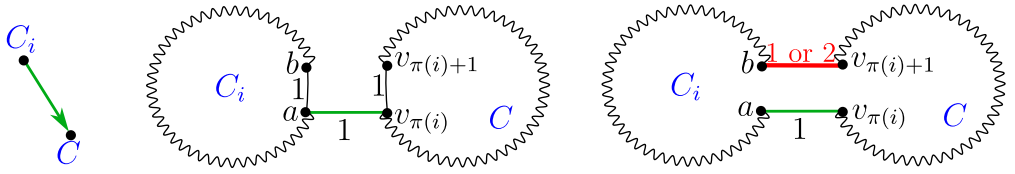
If $H$ has a circuit and there is a vertex $v$ in $H$ not in $C$ such that all in-neighbors of $v$ in $H$ have indegree 0, we proceed similarly. Add to $D'$ all arcs of $D$ with head $v$, and notice that they form an in-tree of depth 1, say $T$, rooted at $v$. Iterate the procedure with $H - V(T)$ playing the role of $H$ until $H$ is a circuit, possibly with some extra vertices of indegree 0 and out-neighbor in $C$.

If $H$ consists only of a circuit $C = (v_0, \dots, v_t = v_0)$, then add to $D'$ the arcs $(v_i, v_{i+1})$ for all odd $i < t$; in case $t$ is odd, also add the edge $(v_{t-1}, v_t)$ to $D'$. Note that $H$ was decomposed into in-trees of depth 1 consisting of paths of length 1 and at most one path of length 2 (whose root is not a digon, as it corresponds to a cycle that does not respect $\mathcal{T}$).

If $H$ has a circuit $C = (v_0, \dots, v_t = v_0)$ and some extra vertices of indegree 0 with out-neighbor in $C$, let $v_{i_1}, \dots, v_{i_k}$ be the vertices of $C$ with in-neighbors not in $C$, where $1 \le i_1 \le \cdots \le i_k \le t$ and let $i_0 = i_k$. We aim at including in $D'$ the in-trees rooted at each $v_{i_j}$ so that the remaining of $C$ consists of odd-length paths, which can be decomposed into in-trees consisting of one arc. This can be achieved by deciding, for each $j \in \{1, \dots, k\}$, whether to include the arc $(v_{i_j - 1}, v_{i_j})$ in the in-tree rooted at $v_{i_j}$ or not. Precisely, add to $D'$ the arcs $(u, v_{i_j})$ for every in-neighbor $u$ of $v_{i_j}$ outside $C$; if the number of vertices in $C$ strictly between $v_{i_{j-1}}$ and $v_{i_j}$ is odd, then also add to $D'$ the arc $(v_{i_j - 1}, v_{i_j})$. Apply the procedure of the first case (when $H$ has no circuit) to decompose the rest of $D$, which consists of odd-length paths. Notice that the parity of each path assures that no singleton will remain, and this is important because all vertices of $C$ are cycles in $V(B)$ matched by $M$. □




(a) Vertices $v_{\pi(i)}$ and $v_{\pi(i+1)}$ are adjacent in $C$.




(b) Vertex $v_{\pi(i)}$ is not adjacent to $v_{\pi(i+1)}$ in $C$.

**Fig. 2.** Joining cycles that belong to depth-1 in-trees of $D'$ into a unique cycle. For the purpose of the analysis, the red and bold edges, which are labeled "1 or 2", will be considered as 2-edges even if they are 1-edges.

At last, Algorithm 2 joins some cycles of $\mathcal{F}$ in order to obtain a 2-factor that respects $\mathcal{T}$. This will happen in two phases. In the first phase, we join cycles that belong to the same nontrivial component of $D'$. In the second (and last) phase, we repeatedly join cycles if they have vertices from the same set in $\mathcal{T}$, to obtain a feasible solution to the problem. This final step prioritizes joining nonpure cycles.

Details of these two phases, done by procedures JoinComponentCycles and JoinDisrespectingCycles, as well as the analysis of the cost of joining cycles, are given in Section 3.1. For now, observe that all cycles at the end of this process respect $\mathcal{T}$. Also, note that length-2 cycles exist in the final solution only if they initially existed in $\mathcal{F}$ and connected terminals of some set $T_a \in \mathcal{T}$ with $|T_a| = 2$. The analysis of the approximation ratio of the algorithm is discussed in Section 3.2.

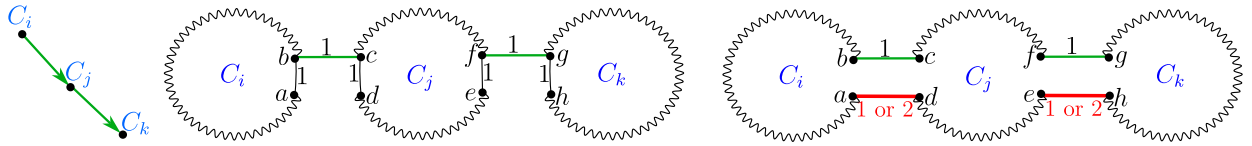
### 3.1. Joining cycles

In the first phase, we join cycles in $\mathcal{F}$ if they belong to the same nontrivial component of $D'$, which can be either an in-tree of depth 1 or a length-2 path. Note that, by Lemma 2, the nonpure cycle and length-2 cycles can only appear as the root of an in-tree in $D'$.

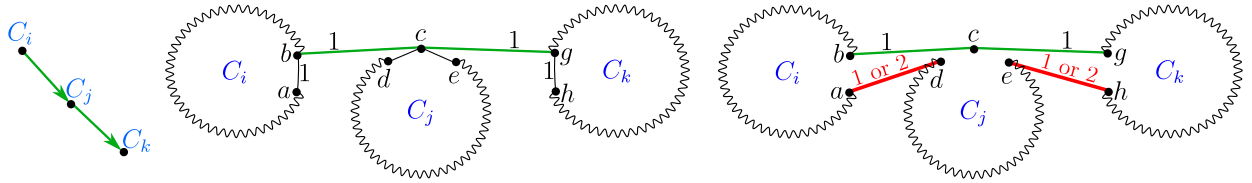
An in-tree of depth 1 of $D'$ consists of a root $C$ and some other cycles $\{C_j\}_{j=1}^{t}$, with $t \geq 1$. Each arc $(C_j, C)$ can be associated with a 1-edge from $G$ such that no two such edges are incident on the same vertex in $C$, because they come from the matching $M$. Let $C = (v_0, \ldots, v_q = v_0)$, and let $\pi : \{1, \ldots, t\} \to \{1, \ldots, q\}$ be an injection such that $v_{\pi(i)}$ is the endpoint in $C$ of the 1-edge associated with $C_i$ in the matching $M$, and $\pi(i) < \pi(j)$ whenever $i < j$. Assume, without loss of generality, that $\pi(i) \neq q$ for all $i \in \{1, \ldots, t\}$ if $t < q$. Note that $v_{\pi(i)}$ and $v_{\pi(i+1)}$ are not necessarily adjacent in $C$ for $i \in \{1, \ldots, t-1\}$. We shall join $C$ to $C_1, \ldots, C_t$ into one single cycle starting with $i = 1$. If $i < t$ and $v_{\pi(i)}v_{\pi(i+1)}$ is an edge of $C$, then we join $C_i$ and $C_{i+1}$ to $C$ as follows (see Fig. 2a). Let $ab$ be a 1-edge in $C_i$ such that $av_{\pi(i)}$ is associated with $(C_i, C)$ and let $cd$ be a 1-edge in $C_{i+1}$ such that $dv_{\pi(i+1)}$ is associated with $(C_{i+1}, C)$. Note that $v_{\pi(i)}v_{\pi(i+1)}$ is also a 1-edge (by the construction of $\mathcal{F}$); remove $ab$, $cd$, and $v_{\pi(i)}v_{\pi(i+1)}$, and add $bc$ (which can be a 1- or a 2-edge), $av_{\pi(i)}$, and $dv_{\pi(i+1)}$; increase $i$ by 2 and start a new iteration. If $i = t$ or $v_{\pi(i)}$ and $v_{\pi(i+1)}$ are not adjacent in $C$, then we join $C_i$ to $C$ in the following manner (see Fig. 2b). Let $ab$ be a 1-edge in $C_i$ such that $av_{\pi(i)}$ is a 1-edge. Note that $v_{\pi(i)}v_{\pi(i)+1}$ is a 1-edge (by construction of $\mathcal{F}$) and that $v_{\pi(i)+1} \neq v_{\pi(j)}$ for all $j \in \{1, \ldots, t\}$. Remove $ab$ and $v_{\pi(i)}v_{\pi(i)+1}$ from $C_i$ and $C$, respectively, and add $av_{\pi(i)}$ and $bv_{\pi(i)+1}$; increment $i$ by 1 and start a new iteration.

As for a component of $D'$ which is a length-2 path, let $C_i$, $C_j$, and $C_k$ be the three cycles that compose it, where $C_i$ is the beginning of the path and $C_k$ is its end. The arcs $(C_i, C_j)$ and $(C_j, C_k)$ are also associated with 1-edges of $G$. Let $ab$ be a 1-edge in $C_i$, $cd$ and $ef$ be 1-edges in $C_j$, and $gh$ be a 1-edge in $C_k$ such that $bc$ is associated with $(C_i, C_j)$ and $fg$ is associated with $(C_j, C_k)$; remove $ab$, $cd$, $ef$, and $gh$, and add $bc$, $fg$, $ad$, and $eh$. Note that possibly $c = f$. The case in which $c \neq f$ is depicted in Fig. 3a and the case in which $c = f$ is depicted in Fig. 3b.

Let $C'$ be the resulting 2-factor after the first phase. This is the output of procedure JoinComponentCycles. For the purpose of the analysis, we shall consider that all cycles obtained from the joining process above contain at least one 2-edge (because the cost of such an edge was already paid for in this part of the analysis). It may still be the case that two separated cycles in $C'$ contain terminals from the same set $T_a \in \mathcal{T}$. Hence, in the last phase, while there are two such cycles, join them in the following order of priority: both




(a) Case $c \neq f$.




(b) Case $c = f$.

**Fig. 3.** Joining cycles that belong to length-2 paths of $D'$ into a unique cycle. For the purpose of the analysis, the red and bold edges, which are labeled "1 or 2", will be considered as 2-edges even if they are 1-edges.

cycles contain a 2-edge, precisely one of the cycles contains a 2-edge, and none contains a 2-edge. The resulting 2-factor of this phase, denoted by $C$, is computed by JoinDisrespectingCycles and is the one returned by Algorithm 2.

Now we proceed to analyze the cost increase caused by joining cycles in these two phases. Note that $w(C)$ is equal to $w(\mathcal{F})$ plus some value due to the increases caused by joining cycles. Let $e_2(X)$ be the total amount of 2-edges in a collection $X$ of cycles.

For the first phase, we charge the increment of the cost for joining cycles to some of the vertices in the cycles being joined. This is done in such a way that each vertex is charged at most once according to the following.

**Claim 2.** *During the first phase, at least $e_2(\mathcal{F})$ vertices are not charged, and all other vertices of $\mathcal{F}$ are charged at most $2/9$.*

**Proof.** Recall that all vertices in length-2 paths and leaves of in-trees in $D'$ are pure cycles of length at least 3. Consider a length-2 path with vertices $C_i$, $C_j$, and $C_k$. The cycles $C_i$, $C_j$, and $C_k$ were joined as in Figs. 3a and 3b, so the extra cost is at most 2, which is charged to the at least 9 vertices (3 per cycle) that belong to these cycles, giving a cost of at most $2/9$ per vertex.

Now consider an in-tree of depth 1 with root $C$ and leaves $C_1, \ldots, C_t$ with $t \geq 1$. When we join cycles $C_i$ and $C_{i+1}$ with $C$, as in Fig. 2a, note that the increase on the cost is at most 1. We charge this cost to the vertices in $C_i$ and $C_{i+1}$, which are at least 6 (3 per cycle), thus costing at most $1/6$ per vertex. In this case, no vertex in cycle $C$ was charged.

When we only join a cycle $C_i$ with $C$, as in Fig. 2b, the increase is also at most 1. We charge this cost to the vertices in $C_i$ and also to the two vertices involved in $C$. Since there are at least 3 vertices in $C_i$, each of these vertices is charged at most $1/5$. Notice that this case happens only if $i = t$ or $v_{\pi(i)}$ and $v_{\pi(i+1)}$ are not adjacent in $C$. This, and the fact that $\pi(i) \neq q$ for all $i \in \{1, \ldots, t\}$ if $t < q$, imply that the charged vertices in $C$ are distinct, and thus each vertex in $C$ is charged at most $1/5$.

It remains to prove that there are at least $e_2(\mathcal{F})$ vertices that were not charged. If the nonpure cycle is not the root of any in-tree, then none of its vertices were charged, and therefore at least $e_2(\mathcal{F})$ vertices were not charged. On the other hand, if the root $C$ of an in-tree is the nonpure cycle, then possibly some vertices of $C$ were charged. However, this happens only if $i = t$ or for $i$ such that $v_{\pi(i)}$ and $v_{\pi(i+1)}$ are not adjacent in $C$. Recall that $\pi(i) \neq q$ for all $i \in \{1, \ldots, t\}$ if $t < q$. Thus, if $t = q$, then $\pi(i) = i$ for all $i \in \{1, \ldots, q\}$, and this means $e_2(\mathcal{F}) = 0$ (by construction of $\mathcal{F}$), so there is nothing to prove. Otherwise, the two vertices charged by the joining of a cycle $C_i$ with $C$ are $v_{\pi(i)}$ and its consecutive vertex in cycle $C$. The two edges incident to $v_{\pi(i)}$ in $C$ are 1-edges (by construction of $\mathcal{F}$). So the only vertices charged in $C$ are the second endpoint of a 1-edge (considering the order in which the vertices appear in $C = (v_0, \ldots, v_q = v_0)$) Thus the second endpoint of each 2-edge is never charged, and there are at least as many vertices not charged as 2-edges in $C$, that is, at least $e_2(\mathcal{F})$ vertices are not charged. □

As for the last phase, the increase in the cost will be considered for each pair of cycles being joined. If both cycles contain 2-edges, joining them will not increase the cost of the solution. If only one of the cycles contains a 2-edge, then the increase in the cost is at most 1. Joining cycles that do not contain 2-edges may increase the cost by 2.

**Claim 3.** *The increase in the last phase is at most $c_p$, where $c_p$ is the number of pure cycles in $\mathcal{F}$ that do not respect $\mathcal{T}$ and are isolated in $D'$.*

**Proof.** In the last phase, due to our consideration at the end of the first phase, all cycles generated in the first phase contain a 2-edge. Thus the only possible increases in cost come from joining one of the $c_p$ pure cycles. The increase is at most 2 if two such cycles are joined and at most 1 if one such cycle is joined to some cycle other than these $c_p$ ones. Therefore, the increase in this phase is at most $c_p$. □

Note that $c_p$ is exactly the number of cycles in $V(B)$ not matched by $M$.

### 3.2. Approximation ratio

We next prove that Algorithm 2 guarantees an 11/9 approximation ratio in general, and then show a slight modification of this algorithm that yields a $\frac{7}{9}$-approximation when every terminal set has size at least 4.

**Theorem 4.** *Algorithm 2 is an $\frac{11}{9}$-approximation for the $\{1,2\}$-Steiner Multicycle problem.*

**Proof.** Let $(G, w, \mathcal{T})$ be an instance of the $\{1,2\}$-Steiner Multicycle problem. Let $n := |V(G)|$ and recall that $e_2(X)$ the total amount of 2-edges in a collection $X$ of cycles.

We start with two lower bounds on $\text{opt}(G, w, \mathcal{T})$. Let $\mathcal{F}$ be the 2-factor obtained in line 1 of Algorithm 2 when applied to $(G, w, \mathcal{T})$. The first one is $w(\mathcal{F})$ because any solution for the Steiner Multicycle problem is a 2-factor in $G$. Thus

$$\text{opt}(G, w, \mathcal{T}) \geq w(\mathcal{F}) = n + e_2(\mathcal{F}). \tag{3}$$

The other one is related to pure cycles in $\mathcal{F}$. Consider an optimal solution $\mathcal{C}^*$ for instance $(G, w, \mathcal{T})$. Thus $\text{opt}(G, w, \mathcal{T}) = n + e_2(\mathcal{C}^*)$. Let $C_1^*, \ldots, C_r^*$ be the cycles of $\mathcal{C}^*$, where $C_i^* = (v_{i0}, \ldots, v_{i|C_i^*|})$ for each $i \in \{1, \ldots, r\}$, with $v_{i0} = v_{i|C_i^*|}$. Let $U = \{v_{ij} : i \in \{1, \ldots, r\}, j \in \{0, \ldots, |C_i^*| - 1\}$, and $v_{ij}v_{ij+1}$ is a 2-edge$\}$ and note that $|U| = e_2(\mathcal{C}^*)$. Let $\ell$ be the number of pure cycles in the 2-factor $\mathcal{F}$ that contain vertices in $U$. Clearly $e_2(\mathcal{C}^*) \geq \ell$, which gives us

$$\text{opt}(G, w, \mathcal{T}) \geq n + \ell. \tag{4}$$

Now let $\mathcal{C}$ be the 2-factor produced by Algorithm 2 for input $(G, w, \mathcal{T})$. Let us show an upper bound on the cost of $\mathcal{C}$. Note that $\mathcal{C}$ has cost $w(\mathcal{F})$ plus the increase in the cost made in the first phase, and then in the final phase of joining cycles. Let us start bounding the total cost increase in the first phase. Let $c_p$ be as in Claim 3. Recall that these $c_p$ cycles are not matched by $M$ and note that the number of vertices in them is at least $3c_p$, because each such cycle does not respect $\mathcal{T}$ and hence has at least three vertices. By Claim 2, at least $e_2(\mathcal{F})$ vertices of $\mathcal{F}$ are not charged during the first phase. Thus, at most $n - 3c_p - e_2(\mathcal{F})$ vertices were charged in the first phase. Also, by Claim 2, each such vertex was charged at most 2/9.

By Claim 3, the increase in the last phase is at most $c_p$. Thus we have

$$\begin{aligned}
w(\mathcal{C}) &\leq w(\mathcal{F}) + \frac{2}{9}(n - 3c_p - e_2(\mathcal{F})) + c_p \\
&= n + e_2(\mathcal{F}) + \frac{2}{9}(n - 3c_p - e_2(\mathcal{F})) + c_p \\
&= \frac{11}{9}n + \frac{7}{9}e_2(\mathcal{F}) + \frac{1}{3}c_p \\
&\leq \frac{11}{9}n + \frac{7}{9}e_2(\mathcal{F}) + \frac{1}{3}\ell \qquad (5) \\
&\leq \frac{7}{9}(n + e_2(\mathcal{F})) + \frac{4}{9}(n + \ell) \\
&\leq \frac{7}{9}\,\text{opt}(G, w, \mathcal{T}) + \frac{4}{9}\,\text{opt}(G, w, \mathcal{T}) = \frac{11}{9}\,\text{opt}(G, w, \mathcal{T}), \qquad (6)
\end{aligned}$$

where (5) holds by Claim 5, and (6) holds by (3) and (4). It remains to prove the following.

**Claim 5.** $c_p \leq \ell$.

**Proof.** Recall that $c_p$ is the number of pure cycles in $\mathcal{F}$ that are isolated in $D'$ and do not respect $\mathcal{T}$, and observe that $\ell \leq |U|$.

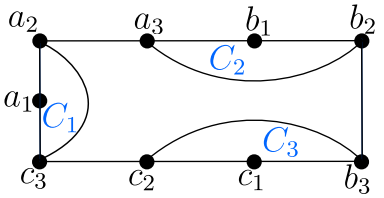
We shall describe a matching in the bipartite graph $B$ with at most $\ell$ unmatched cycles. From this, because $M$ is a maximum matching in $B$ and there are at least $c_p$ cycles not matched by $M$, we conclude that $c_p \leq \ell$.

For each $i \in \{1, \ldots, r\}$, go through the vertices of $C_i^*$ from $j = 0$ to $|C_i^*| - 1$ and if, for the first time, we find a vertex $v_{ij} \notin U$ that belongs to a pure cycle $C$ (which does not respect $\mathcal{T}$) such that $v_{ij+1}$ is not in $C$, we match $C$ to $v_{ij+1}$ in $B$. Note that, as $v_{ij} \notin U$, the edge between $C$ and $v_{ij+1}$ is indeed in $B$. Every pure cycle that does not respect $\mathcal{T}$ is matched by this procedure, except for at most $\ell$. ◊

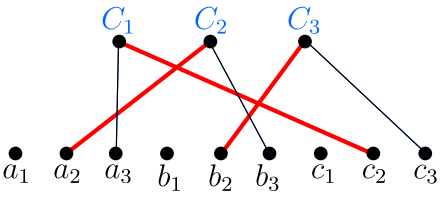
This completes the proof of the theorem. □

We next show that the previous analysis is tight. Consider the instance depicted in Fig. 4a, with 9 vertices and $\mathcal{T} = \{\{a_1, a_2, a_3\}, \{b_1, b_2, b_3\}, \{c_1, c_2, c_3\}\}$.[2] There is a Hamiltonian cycle in the graph with only 1-edges, so the optimum costs 9. However, there is also a 2-factor of cost 9 consisting of the three length-3 cycles $C_1$, $C_2$ and $C_3$, as in Fig. 4a. The matching in the graph $B$

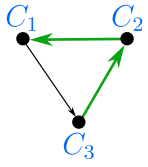
[2] There is a mistake in this tight example in the preliminary version published in LATIN 2022 proceedings.



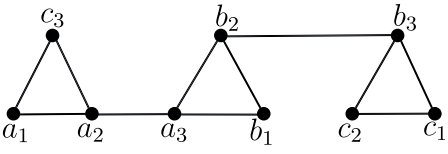

(a) Initial graph $G$ and 2-factor $\mathcal{F}$. All depicted edges are 1-edges while the missing ones are 2-edges.



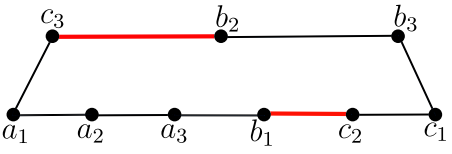

(b) Bipartite graph built from $\mathcal{F}$ and matching $M$ highlighted.




(c) Digraph $D$ and subdigraph $D'$ highlighted.




(d) Joining cycles of the length-2 path in $D'$.




(e) Final solution $\mathcal{C}$, with two 2-edges, and cost 11.

**Fig. 4.** Tight example for Algorithm 2.

might correspond to the 1-edge between $C_2$ and $C_1$, the 1-edge between $C_3$ and $C_2$, and the 1-edge between $C_1$ and $C_3$, as in Fig. 4b. This leads to a length-2 path in $D'$, as in Fig. 4c. The process of joining these cycles, as the algorithm does, might lead to an increase of 2 in the cost, resulting in the solution of cost 11 depicted in Fig. 4e, which achieves a ratio of exactly $11/9$. This example can be generalized to have $n = 9k$ vertices, for any positive integer $k$. Indeed, we can replicate $k$ times the graph depicted in Fig. 4a putting 2-edges between them. The optimum would have cost $9k$, consisting of the cycles on 9 vertices in each block, and the algorithm's output can cost $11k$, consisting of the same cycles of weight 11 in each block.

Similarly to what Papadimitrou and Yannakakis [21] achieved for the $\{1,2\}$-TSP, we also derive the following.

**Corollary 1.** *Algorithm 2 is a $\frac{7}{6}$-approximation for the $\{1,2\}$-Steiner Multicycle problem when $|T_a| \geq 4$ for all $T_a \in \mathcal{T}$.*

**Proof.** For weights 1 and 2, there is a polynomial-time algorithm that computes a minimum-weight 2-factor that contains no triangle. (See Lemma 3 in Appendix A.) Using this algorithm within Special2Factor in Algorithm 2, we can guarantee that there are at least 4 vertices per cycle in the produced 2-factor $\mathcal{F}$. The charging argument presented in Claim 2 can use the fact that the cycles have length at least 4, which increases the number of vertices that can be charged for the cost increase.

For instance, when we join a cycle $C_i$ with $C$, as in Fig. 2b, the increase is at most 1, and we charge this cost to the vertices in $C_i$ and also to the two vertices involved in $C$. Now there are at least 4 vertices in $C_i$, so each of these vertices is charged at most $1/6$. The case in which the cost per charged vertex is the greatest is when the three cycles of a length-2 path of $D'$ are joined, as in Figs. 3a and 3b. In this case, the extra cost is at most 2, which is now charged to the at least 12 vertices that belong to these cycles, giving a cost of at most $1/6$ per vertex. So the value charged per vertex is at most $1/6$ in all cases, and the result follows. □

## 4. Asymmetric Steiner Multicycle problem

In this section, we consider a version of the Steiner Multicycle in which the input graph is a complete digraph $D$ on $n$ vertices with arc set $A(D)$, and the weight function $w\colon A(D) \to \mathbb{Q}_+$ does not necessarily satisfy $w(u,v) = w(v,u)$ for all $u, v \in V(D)$ with $u \neq v$. We shall assume that the arc weights still satisfy the triangular inequality: $w(a,c) \leq w(a,b) + w(b,c)$ for all distinct $a, b, c \in V(D)$. As before, we also have a collection $\mathcal{T}$ of terminal sets which partitions $V(D)$, and the goal now is to find a minimum weight *directed* 2-factor of $D$ that respects $\mathcal{T}$.

We next devise an $O(\lg n)$-approximation algorithm for this problem that is inspired by the algorithm with the same approximation ratio for the Asymmetric TSP, proposed by Frieze, Galbiati, and Maffioli [12]. At each iteration, their algorithm proceeds as follows. It starts with an induced subdigraph $D'$ of $D$ and what we call a *strongly Eulerian* spanning subdigraph $C$ of $D$ (initially $D' = D$ and $C$ has no arcs). Then, it finds a minimum weight 2-factor $\mathcal{F}$ in $D'$, and makes $C = C \cup \mathcal{F}$. If $\mathcal{F}$ has only one cycle, then $C$ is connected, and their algorithm outputs a Hamiltonian cycle obtained from shortcutting $C$ into a cycle. If $\mathcal{F}$ has more than one cycle, then their algorithm chooses a vertex in each cycle of $\mathcal{F}$, called its *representative*, it lets $D'$ be the subdigraph of $D$ induced on these representatives, and it starts the next iteration with the new $D'$ and $C$. The authors observed that each 2-factor $C$ has weight bounded by the length of the optimal TSP tour, and the number of iterations is bounded by $\lg n$, because the number of components of $C$ is divided by two in each iteration. This implies the $O(\lg n)$ approximation ratio.

Our algorithm aims at obtaining a 2-factor that respects $\mathcal{T}$. Hence it stops once each terminal set is contained in a component of $C$. It also differs from the algorithm due to Frieze, Galbiati, and Maffioli [12] in the way it chooses the representatives. At each iteration

of our algorithm, one has to guarantee that the 2-factor $\mathcal{F}$ has weight bounded by the optimal value, and that the number of iterations is still $O(\lg n)$. We shall see that this can be done using a minimal edge cover of an auxiliary graph to find *good* representatives. Recall that an *edge cover* in a graph is a set $M$ of edges such that every vertex is incident to an edge in $M$. A minimal edge cover on a graph with $n$ vertices can be computed in time $O(n^{2.5})$ using the algorithm for the maximum matching problem in general graphs due to Micali and Vazirani [19].

A digraph $D'$ is said to be *strongly Eulerian* if, for every $v \in V(D')$, the indegree and outdegree of $v$ in $D'$ are each equal to some $k(v) \in \mathbb{Z}_+$, and $D' - v$ contains precisely $k(v) - 1$ more components than $D$. We say that a component $K$ of $D' - v$ is *adjacent* to $v$ if there is a vertex in $K$ which is a neighbor of $v$ in $D'$. Analogously to the observation in [12], one may notice that, for every $v \in V(D')$ and each connected component $K$ of $D' - v$ which is adjacent to $v$, there exist distinct vertices $u, w \in V(K)$ such that $(u, v)$ and $(v, w)$ belong to $A(D')$. Procedure DIRECTEDSHORTCUT shows how to obtain a directed 2-factor $\mathcal{F}$ of $D$ from a strongly Eulerian spanning subdigraph $D'$ of $D$ so that $\mathcal{F}$ has the same connected components as $D'$. If there is an underlying weight function $w : A(D) \to \mathbb{Q}_+$ satisfying the triangular inequality, then $w(\mathcal{F}) \leq w(D')$. For each iteration of the while loop in line 1, $D'$ is a strongly Eulerian digraph with the same connected components. We remark that this algorithm corresponds to the shortcutting procedure described in [12] applied to every component of $D'$. For the sake of completeness, procedure DIRECTEDSHORTCUT is presented in Algorithm 3. Note that this takes polynomial time.

**Algorithm 3** DIRECTEDSHORTCUT($D'$).

**Input:** a strongly Eulerian digraph $D'$
**Output:** a directed 2-factor with the same connected components as $D'$
1: **while** $D'$ is not a directed 2-factor **do**
2: Let $v \in V(D')$ be such that with $k(v) > 1$
3: Let $C_1, C_2$ be distinct components of $D' - v$ that are adjacent to $v$
4: Let $u_i, w_i \in V(C_i)$ such that $(u_i, v), (v, w_i) \in A(D)$ for $i \in \{1, 2\}$
5: $A(D') \leftarrow [A(D') \setminus \{(u_1, v), (v, w_2)\}] \cup \{(u_1, w_2)\}$
6: **return** $D'$

Let $\eta_{\mathcal{T}}(C)$ denote the number of cycles in a 2-factor $C$ that do not respect $\mathcal{T}$. The procedure REPRESENTATIVES takes as input $C$ and $\mathcal{T}$, and it creates an auxiliary undirected graph $G$ with vertex set being the $\eta_{\mathcal{T}}(C)$ cycles in $C$ that do not respect $\mathcal{T}$ and edge set $\{\{C, C'\} : C \neq C', V(C) \cap T \neq \emptyset$, and $V(C') \cap T_a \neq \emptyset$ for some $T_a \in \mathcal{T}\}$. Then, it computes a minimal edge cover $M$ of $G$ and, for each edge $\{C, C'\} \in M$, it chooses a pair of vertices $\{r_C, r_{C'}\}$ such that $r_C \in V(C) \cap T_a$ and $r_{C'} \in V(C') \cap T_a$ where $T_a$ is a terminal set in $\mathcal{T}$ that intersects both $C$ and $C'$. The procedure then returns the set of vertices $R = \bigcup_{\{C,C'\} \in M} \{r_C, r_{C'}\}$.

**Algorithm 4** REPRESENTATIVES($\mathcal{C}, \mathcal{T}$).

**Input:** a directed 2-factor $\mathcal{C}$ of an induced subdigraph of $D$ and $\mathcal{T}$
**Output:** a set of vertices $R \subseteq V(D)$
1: Let $E = \{\{C, C'\} : C \neq C', V(C) \cap T_a \neq \emptyset$, and $V(C') \cap T_a \neq \emptyset$ for some $T_a \in \mathcal{T}\}$
2: Let $G$ be the graph with vertex set $\{C \in \mathcal{C} : C$ does not respect $\mathcal{T}\}$ and edge set $E$
3: $M \leftarrow$ MINIMALEDGECOVER($G$)
4: $R \leftarrow \emptyset$
5: **for** each edge $\{C, C'\} \in M$ **do**
6: Let $T_i \in \mathcal{T}$ such that $T_i \cap C \neq \emptyset$ and $T_i \cap C' \neq \emptyset$
7: Let $r_C \in V(C) \cap T_i$ and $r_{C'} \in V(C') \cap T_i$
8: $R \leftarrow R \cup \{r_C, r_{C'}\}$
9: **return** $R$

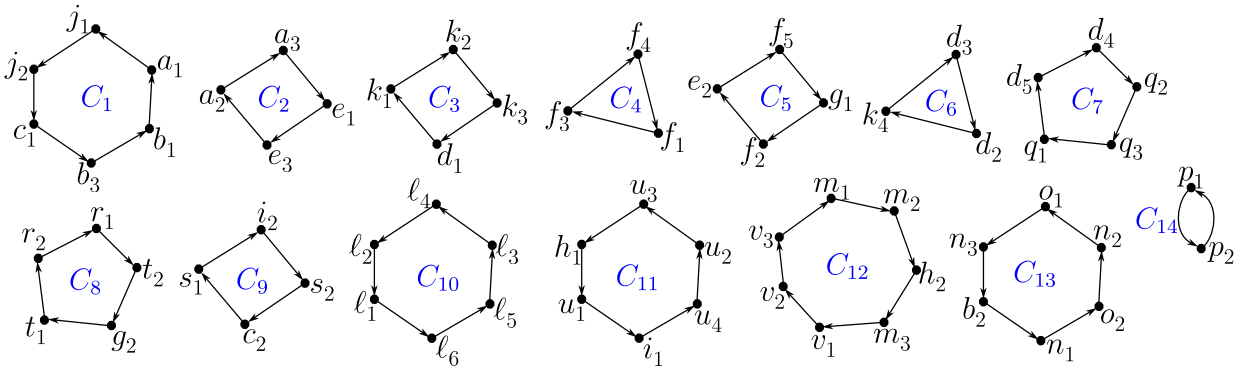
We next argue that Algorithm 4 produces a set $R$ satisfying

(i) $R \cap V(C) \neq \emptyset$ for every $C \in V(G)$;
(ii) $|R \cap T_a| \neq 1$ for every terminal set $T_a \in \mathcal{T}$; and
(iii) $|R \cap V(C)| = 1$ for at least $\eta_{\mathcal{T}}(C)/2$ cycles $C$ in $\mathcal{C}$.

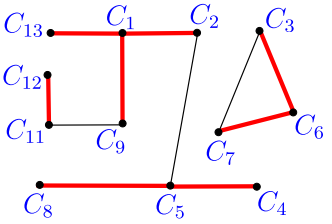
The first property holds because $M$ is an edge cover of $G$, thus, for every $C \in V(G)$, at least one vertex from $V(C)$ was included in $R$. The second property follows from the fact that, for each edge in $M$, two distinct vertices of the same terminal set were simultaneously included in $R$. For a terminal set $T_a \in \mathcal{T}$ contained in a cycle $C \in V(G)$, we have $|R \cap T_a| = 0$. The last property holds because, in every minimal edge cover, at least half of the vertices are covered exactly once. Indeed, every edge of a minimal edge cover is incident to a vertex that is only covered by this edge, and there are at least $|V(G)|/2 = \eta_{\mathcal{T}}(C)/2$ edges in any edge cover of $G$. Every cycle $C \in \mathcal{C}$ such that $|R \cap V(C)| = 1$ is said to be *lonely*. Note that property (iii) guarantees that there are at least $\eta_{\mathcal{T}}(C)/2$ lonely cycles.

Algorithm 5 formalizes the steps of our algorithm for the ASYMMETRIC STEINER MULTICYCLE problem. It uses an auxiliary procedure that computes a minimum weight directed 2-factor in a weighted digraph. See Fig. 5 for an example.

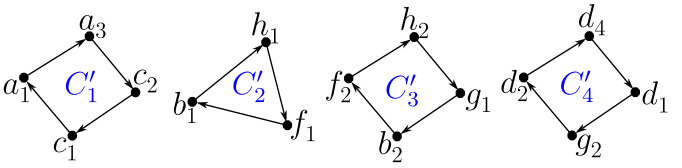
The way of choosing representatives in this algorithm is more complex than the way used in [12]. This is because deriving an upper bound on the weight of $C'$ in terms of an optimal 2-factor is more challenging than in terms of a minimum weight TSP tour. Specifically, a TSP tour can be shortcut into a 2-factor for any set $R$ of representatives. However, this might not be the case for an optimal 2-factor. Indeed, in [12], only one representative vertex is (arbitrarily) chosen from each cycle. However, in the example



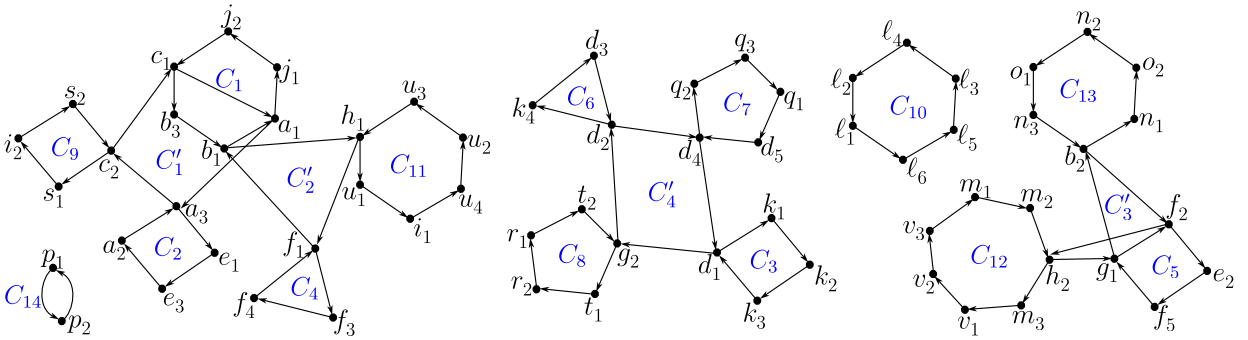

(a) Minimum directed 2-factor $\mathcal{C}$ obtained at line 1.



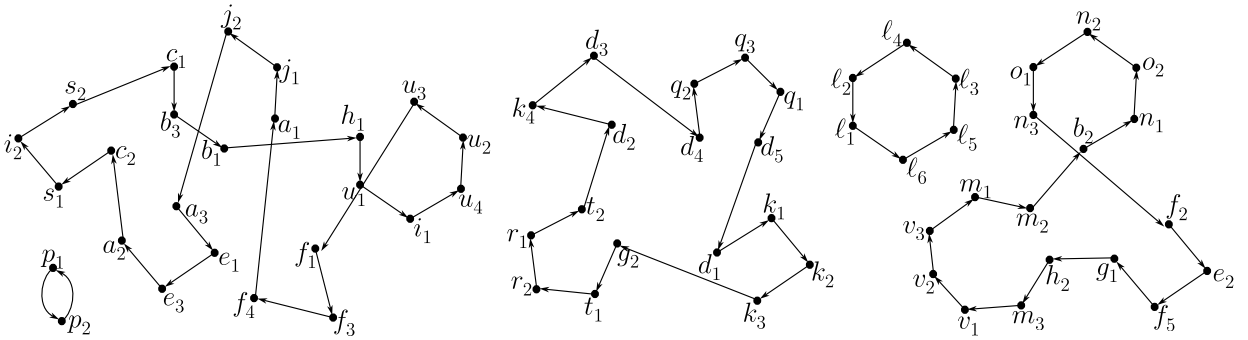

(b) Edge cover obtained from $\mathcal{C}$ for $R = \{a_1, a_3, b_1, b_2, c_1, c_2, d_1, d_2, d_4, f_1, f_2, g_1, g_2, h_1, h_2\}$.




(c) Minimum directed 2-factor $\mathcal{C}'$ restricted to $D'$.



(d) Strongly Eulerian digraph $D''$.



(e) New 2-factor obtained from shortcutting an Eulerian tour in $D''$.

**Fig. 5.** Auxiliar digraphs and structures built by Algorithm 5.

**Algorithm 5** SteinerMulticycleApprox_Asymmetric($G$, $w$, $\mathcal{T}$).

**Input:** a complete digraph $D$, a weight function $w\colon A(D) \to \mathbb{Q}_+$, and a partition $\mathcal{T}$ of $V(D)$
**Output:** a directed 2-factor $C$ in $D$ that respects $\mathcal{T}$
1: $C \leftarrow$ MinimumDirected2Factor($D$, $w$)
2: **while** $\eta_{\mathcal{T}}(C) > 0$ **do**
3: $R \leftarrow$ Representatives($C$, $\mathcal{T}'$)
4: Let $D'$ be the complete digraph induced by $R$ on $D$
5: Let $w'$ be the restriction of $w$ to $A(D')$
6: $C' \leftarrow$ MinimumDirected2Factor($D'$, $w'$)
7: Let $D''$ be the digraph induced by $C' \cup C$
8: $C \leftarrow$ DirectedShortcut($D''$)
9: **return** $C$

in Fig. 5, suppose $a_1$, $a_2$, and $a_3$ are alone in a cycle in any optimal 2-factor, and vertex $a_1$ was chosen as the representative of $C_1$, while vertex $e_1$ is chosen as the representative of $C_2$ (hence $a_2$ and $a_3$, which are also in $C_2$, would not be representatives). In this case, no shortcut of any optimal 2-factor would result in a 2-factor on the chosen representatives: $a_1$ would be isolated in a shortcut of any optimal 2-factor on the chosen representatives. This means we cannot guarantee that the optimal cost is an upper bound on the minimum weight $w(C')$ of a 2-factor on the representatives. So we needed to develop a way to guarantee that the shortcut of an optimal 2-factor on $R$ is a 2-factor, keeping the property that $C'$ joins a good amount of cycles of $C$ that do not respect $\mathcal{T}$. In the other extreme, one could consider including in $R$ all vertices in unhappy terminal sets because then the shortcut on $R$ of any optimal solution would be a 2-factor. But this 2-factor might not join unhappy terminal sets: indeed, all terminal sets might be unhappy in $C$, and in this case $R$ would be the whole set of vertices and $C' = C$, leading the algorithm to loop forever.

**Theorem 6.** *Algorithm 5 is an* $O(\lg n)$*-approximation for the Asymmetric Steiner Multicycle problem, where* $n$ *is the number of vertices in the given digraph.*

**Proof.** Let $(D, w, \mathcal{T})$ be an instance of the Asymmetric Steiner Multicycle, where $D$ has $n$ vertices. We first show that the solution produced by Algorithm 5 is indeed feasible for $(D, w, \mathcal{T})$. It follows from its construction that the digraph $D''$ (computed at line 7) is a strongly Euclidean spanning subdigraph of $D$, and so $C$ is indeed a directed 2-factor in $D$ with the same components as $D''$. By the condition in line 2, the set $C$ returned by Algorithm 5 respects $\mathcal{T}$, and thus $C$ is a valid solution for $(D, w, \mathcal{T})$.

We now prove that the weight of each minimum weight 2-factor computed at line 6 is upper bounded by the weight of an optimal solution for $(D, w, \mathcal{T})$. Consider the complete digraph $D'$ and the weight function $w'$ used in line 6, and let $C'$ be the minimum weight directed 2-factor of $D'$ obtained in line 6. Consider an optimal 2-factor $C^*$ for the instance $(D, w, \mathcal{T})$, that is, a minimum weight 2-factor in $(D, w)$ that respects $\mathcal{T}$. Now consider a shortcutting on $C^*$ to go only through the vertices of $R$, say $C^*_R$. Note that $C^*_R$ is certainly a 2-factor in $D'$ because no cycle in $C^*_R$ has only one terminal in $R$. Also, $C^*_R$ has weight at most $\mathrm{opt}(D, w, \mathcal{T})$, since $w$ satisfies the triangular inequalities. As $C'$ is a minimum 2-factor in $D'$, we have that $w(C') \leq w(C^*_R)$, leading to $w(C') \leq \mathrm{opt}(D, w, \mathcal{T})$.

Consider an iteration of the while loop in line 2. Let $C$ be the directed 2-factor at the beginning of this iteration, $R$ be the set from line 3, and $\hat{C}$ be the directed 2-factor obtained in line 8. The following assertion holds.

**Claim 7.** $\eta_{\mathcal{T}}(\hat{C}) \leq \frac{3}{4}\eta_{\mathcal{T}}(C)$.

**Proof.** It suffices to argue that the difference $\Delta := \eta_{\mathcal{T}}(C) - \eta_{\mathcal{T}}(\hat{C})$ is at least half the number of lonely cycles in $C$. Let $C$ be a lonely cycle in $C$ and let $C'$ be the cycle in $C'$ containing the single representative in $R \cap V(C)$. If $C'$ contains a representative of a cycle that is not a lonely cycle in $C$, then $C$ contributes with 1 to $\Delta$. Otherwise, every vertex in $C'$ is a representative of a lonely cycle in $C$, and so $C$ contributes with $(|C'| - 1)/|C'|$ to $\Delta$. As $|C'| \geq 2$, we conclude that every lonely cycle in $C$ contributes with at least $1/2$ to $\Delta$. Because there are at least $\eta_{\mathcal{T}}(C)/2$ lonely cycles in $C$ by property (iii) of $R$, we have $\eta_{\mathcal{T}}(C) - \eta_{\mathcal{T}}(\hat{C}) \geq \frac{1}{4}\eta_{\mathcal{T}}(C)$, which implies $\eta_{\mathcal{T}}(\hat{C}) \leq \eta_{\mathcal{T}}(C) - \frac{1}{4}\eta_{\mathcal{T}}(C) = \frac{3}{4}\eta_{\mathcal{T}}(C)$. ♢

As $\eta_{\mathcal{T}}(C_0) \leq n$ for the initial directed 2-factor $C_0$ from line 1, it follows from the previous claim that the maximum number of iterations of the while loop in line 2 is $O(\lg n)$. This implies that the weight of the solution produced by Algorithm 5 is at most $O(\lg n)\,\mathrm{opt}(D, w, \mathcal{T})$. □

## 5. Final remarks

If there is only one terminal set, then the Steiner Multicycle turns into the TSP. For the metric TSP, there exists a $\frac{3}{2}$-approximation, so the first natural question is whether there is also a $\frac{3}{2}$-approximation for the metric Steiner Multicycle, or at least some approximation with a ratio better than 3.

The difficulty in the Steiner forest is also a major difficulty in the Steiner Multicycle problem: how to find out what is the right way to cluster the terminal sets. Indeed, if the number $k$ of terminal sets is bounded by a constant, then one can use brute force to guess the way an optimal solution clusters the terminal sets, and then, in the case of the Steiner Multicycle, apply any approximation for the TSP to each instance induced by one of the clusters. This leads to a $\frac{3}{2}$-approximation for any metric instance

with bounded number of terminal sets. It also leads to better approximations for hereditary classes of instances for which there are better approximations for the TSP.

It would be nice to find out whether or not the cost of a minimum weight perfect matching on the set of odd vertices of a minimum weight Steiner forest is at most the optimum value for the Steiner Multicycle.

Observe that, for the $\{1,2\}$-Steiner Multicycle, we can achieve the same approximation ratio than the modified algorithm for the $\{1,2\}$-TSP, but for the more general metric case, our ratio is twice the best ratio for the metric TSP. This comes from the fact that the backbone structure used in the solution for the metric TSP (the MST and the minimum weight 2-factor) can be computed in polynomial time. For the $\{1,2\}$-Steiner Multicycle we can still use the 2-factor, but the two adaptations of the MST for the metric Steiner Multicycle (the Steiner forest and the survivable network design) are hard problems, for which we only have 2-approximations, not exact algorithms.

In fact, for the $\{1,2\}$-TSP, better approximation algorithms are known: there is an $\frac{8}{7}$-approximation by Berman and Karpinski [4], and a $\frac{7}{6}$-approximation and a faster $\frac{8}{7}$-approximation by Adamaszek et al. [1]. The latter algorithms rely on some tools that we were not able to extend to the $\{1,2\}$-Steiner Multicycle. On the other hand, the $\frac{8}{7}$-approximation due to Berman and Karpinski seems to be more amenable to an adaptation.

Recently, constant-factor approximations were presented for the asymmetric TSP [26,27]. Thus a natural direction for further research is to design constant-factor approximation algorithms also for the Asymmetric Steiner Multicycle.

## CRediT authorship contribution statement

**Cristina G. Fernandes:** Conceptualization, Investigation, Methodology, Validation, Writing – original draft, Writing – review & editing. **Carla N. Lintzmayer:** Conceptualization, Investigation, Methodology, Validation, Writing – original draft, Writing – review & editing. **Phablo F.S. Moura:** Conceptualization, Investigation, Methodology, Validation, Writing – original draft, Writing – review & editing.

## Declaration of competing interest

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

## Data availability

No data was used for the research described in the article.

## Acknowledgements

C. G. Fernandes was partially supported by CNPq (Proc. 310979/2020-0 and 423833/2018-9). C. N. Lintzmayer was partially supported by CNPq (Proc. 312026/2021-8) and by L'ORÉAL-UNESCO-ABC For Women In Science. P. F. S. Moura was partially supported by Internal Funds KU Leuven (C14/22/026). This study was also financed in part by CAPES - Finance Code 001, by FAPESP grant 2019/13364-7, and by CNPq grant 404315/2023-2. CNPq is the National Council for Scientific and Technological Development, CAPES is the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil, and FAPESP is the Fundação de Amparo à Pesquisa do Estado de São Paulo. The authors would like to thank the anonymous reviewers for their valuable comments and suggestions, which helped to improve the quality of the paper.

## Appendix A. Minimum-weight triangle-free 2-factor

Hartvigsen, in his PhD thesis [14, Section 3, Chapter 3], described an algorithm that finds, in a given graph, a triangle-free simple 2-matching with the maximum number of edges. Recently, he presented a new algorithm to solve this problem [13]. In this appendix, we detail how to use any of these algorithms to find a minimum-weight triangle-free 2-factor in a complete graph with all edge weights 1 or 2. Let us start by clarifying the notation involved, as it is used differently throughout the literature.

Let $H$ be a graph (not necessarily complete, and without weights). A subgraph of $H$ whose maximum degree is 2 is sometimes called a *2-matching*, and it differs from a 2-factor as it allows for degree-1 and degree-0 vertices. That is, a 2-matching is a collection of vertex-disjoint paths and cycles in $H$.

Sometimes, in the literature, a 2-matching is used to refer to a weight function that assigns weight 0, 1, or 2 to each edge of a simple graph $H$ so that the sum of the weights of the edges incident to each vertex is at most 2. An edge that is assigned a weight of 2 works essentially as a length-2 cycle. For this reason, sometimes in the literature, the 2-matching as we defined is referred to as a *simple* 2-matching (as it does not allow for these parallel edges). Also, a 2-factor is sometimes called a perfect simple 2-matching. Indeed, a simple 2-matching $F$ is *perfect* if every vertex is incident to exactly two edges from $F$.

There are polynomial-time algorithms that find a minimum-weight 2-factor in a complete graph with arbitrary edge weights. Such an algorithm can be used to find a simple 2-matching in a given graph $H$ with the maximum number of edges: just consider the edges of $H$ as having weight 1, and the non-edges as having weight 2, and throw away the weight-2 edges of the obtained 2-factor.

On the other hand, no polynomial-time algorithm is known to find a minimum-weight (or, equivalently, maximum-weight) *triangle-free* 2-factor in a complete graph with arbitrary edge weights [13].

There are some statements in the literature [1,21], when discussing the $\frac{7}{6}$-approximation for TSP, that might lead one to think that Hartvigsen's algorithm for finding a maximum-size triangle-free simple 2-matching could be used to find a minimum-weight triangle-free 2-factor for general weights. But that does not seem to be the case. What is true, and stated explicitly in [15], is that Hartvigsen's algorithm can be used to find a *maximum*-weight triangle-free 2-factor in a complete graph $G$ with edge weights 0 and 1. In a similar manner, one can also use Hartvigsen's algorithm to find a minimum-weight triangle-free 2-factor in complete graphs with edge weights 1 and 2. In the following lemma we describe with details how to deal with this last case, which is needed by Corollary 1. Its proof can be straightforwardly adapted to find a *maximum-weight* triangle-free 2-factor in complete graphs with edge weights 0 and 1.

**Lemma 3.** *There exists a polynomial-time algorithm for finding a minimum-weight triangle-free 2-factor in complete graphs whose edges have weights 1 or 2.*

**Proof.** Let $G$ be a complete graph of order $n$ with edge weights 1 or 2. Note that all minimum-weight triangle-free 2-factors in $G$ have the same number $a$ of weight-2 edges, and their weight is $\alpha = n + a$.

Let $H$ be an unweighted graph obtained from $G$ by removing all weight-2 edges. Also note that all triangle-free collections of cycles and paths in $H$ with the maximum number of edges have the same number $b$ of paths, and contain exactly $\beta = n - b$ edges.

First note that, given a minimum-weight triangle-free 2-factor $\mathcal{F}$ in $G$ with $a$ weight-2 edges and total weight $\alpha = n + a$, one may remove the weight-2 edges from $\mathcal{F}$ to obtain a triangle-free collection of cycles and paths in $H$ with $a$ paths and $n - a$ edges. This means that $b \leq a$ and thus

$$\alpha = n + a \geq n + b = \beta + 2b. \tag{7}$$

Now let $\mathcal{C}$ be a collection of triangle-free cycles and paths in $H$ with the maximum number of edges. This collection can be computed in polynomial time using one of the algorithms due to Hartvigsen [13,14]. Recall that $\mathcal{C}$ has $b$ paths and $\beta$ edges. If $b = 0$, then $\mathcal{C}$ is already a triangle-free 2-factor in $G$ of weight $n$, and thus it has minimum weight.

Suppose now that $b > 1$. If the paths in $\mathcal{C}$ contain altogether at least four vertices, then one may join these $b$ paths in $\mathcal{C}$ into a single cycle using weight-2 edges to obtain a triangle-free 2-factor of weight $\beta + 2b$, which is minimum by (7). Note that this encompasses the case $b \geq 4$ as every path has at least one vertex.

If $b \in \{2, 3\}$ and the paths in $\mathcal{C}$ contain altogether at most three vertices, then $\mathcal{C}$ contains one of the following: (i) two trivial paths, (ii) three trivial paths, or (iii) one trivial path and one path having two vertices (using a weight-1 edge). Observe now that all edges in $E(G)$ linking vertices of distinct paths in $\mathcal{C}$ have weight 2 due to the maximality of $\mathcal{C}$ in $H$. Suppose first that there exists a weight-1 edge $ux \in E(G)$ with $x \in V(C)$ for some cycle $C \in \mathcal{C}$ and $u$ belonging to a path in $\mathcal{C}$. Let $P$ be a path in $G$ with one endpoint at $u$ that is obtained by joining the $b$ paths in $\mathcal{C}$ into a single path using weight-2 edges, and let $v$ be the endpoint of $P$ different from $u$. In this case, one can remove $C$ and all paths from $\mathcal{C}$, and add the cycle formed by $P$, $ux$, $vy$, and $C - xy$, where $xy \in E(C)$. The obtained collection is a triangle-free 2-factor of $G$ of weight at most $\beta - 1 + 1 + 2(b-1) + 2 = \beta + 2b$, which is minimum by (7).

Suppose now that each edge in $G$ from a vertex in a cycle of $\mathcal{C}$ to a vertex in a path of $\mathcal{C}$ has weight 2. One may proceed similarly to the previous case to obtain a triangle-free 2-factor, say $\mathcal{C}'$, of weight at most $\beta - 1 + 2 + 2(b-1) + 2 = \beta + 2b + 1$. If $b = 2$, then every edge in $G$ incident to a vertex of a path in $\mathcal{C}$ has weight 2, except for exactly the weight-1 edge which belongs to one of the paths in case (iii). As a consequence, every triangle-free 2-factor of $G$ must contain at least three weight-2 edges, which implies that it has weight at least $n + 3 = \beta + b + 3 = \beta + 2b + 1$. If $b = 3$, then the paths in $\mathcal{C}$ contain no weight-1 edge, and so each vertex in these paths is only incident with weight-2 edges in $G$. Hence every triangle-free 2-factor of $G$ must contain at least four weight-2 edges, which implies that it has weight at least $n + 4 = \beta + b + 4 = \beta + 2b + 1$. In both cases, we conclude that $\mathcal{C}'$ is a minimum-weight triangle-free 2-factor of $G$.

It only remains to consider the case where $b = 1$ and the single path in $\mathcal{C}$ has at most three vertices. Let $P$ be such a path, and let $u$ and $v$ denote its endpoints (with $u = v$ in one case). Suppose first that there exists one weight-1 edge $ux \in E(G)$ with $x \in V(C)$ for some cycle $C \in \mathcal{C}$. Analogously to a previous case, one can remove $C$ and $P$ from $\mathcal{C}$, and add the cycle formed by $P$, $ux$, $vy$, and $C - xy$, where $xy \in E(C)$. This collection is a triangle-free 2-factor of $G$ of weight at most $\beta - 1 + 1 + 2 = \beta + 2b$, and thus it has minimum weight by (7).

Thus each edge in $G$ linking a vertex in a cycle of $\mathcal{C}$ to an endpoint of $P$ has weight 2. One may proceed similarly to the previous case to obtain a triangle-free 2-factor $\mathcal{C}'$ of weight $\beta - 1 + 2 + 2 = \beta + 2b + 1$. Now note that if $P$ is a trivial path, a path of length 1, or a path of length 2 where $uv$ has weight 2, then every edge in $G$ incident with $u$ or $v$ that is not in $P$ has weight 2. Otherwise, $P$ has length 2 and $uv$ has weight 1, which means every edge from a vertex in $P$ to a vertex not in $P$ in $G$ has weight 2. In both cases, we conclude that every triangle-free 2-factor of $G$ must contain at least two weight-2 edges, and thus a total weight of at least $n + 2 = \beta + b + 2 = \beta + 2b + 1$. Thus $\mathcal{C}'$ is indeed a minimum-weight triangle-free 2-factor of $G$. □

## References

[1] A. Adamaszek, M. Mnich, K. Paluch, New approximation algorithms for (1,2)-TSP, in: I. Chatzigiannakis, C. Kaklamanis, D. Marx, D. Sannella (Eds.), 45th International Colloquium on Automata, Languages, and Programming (ICALP 2018), in: Leibniz International Proceedings in Informatics (LIPIcs), vol. 107, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 2018, pp. 9:1–9:14.

[2] S. Arora, Polynomial time approximation schemes for Euclidean Traveling Salesman and other geometric problems, J. ACM 45 (5) (1998) 753–782, https://doi.org/10.1145/290179.290180.
[3] N. Bansal, S. Bravyi, B.M. Terhal, Classical approximation schemes for the ground-state energy of quantum and classical Ising spin Hamiltonians on planar graphs, Quantum Inf. Comput. 9 (7) (2009) 701–720.
[4] P. Berman, M. Karpinski, 8/7-approximation algorithm for (1,2)-TSP, in: Proc. of the 17th Annual ACM-SIAM Symposium on Discrete Algorithm (SODA), 2006, pp. 641–648.
[5] G. Borradaile, P.N. Klein, C. Mathieu, A polynomial-time approximation scheme for Euclidean Steiner forest, ACM Trans. Algorithms 11 (3) (2015) 19:1–19:20, https://doi.org/10.1145/2629654.
[6] N. Christofides, Worst-case analysis of a new heuristic for the traveling salesman problem, Technical Report 388, Carnegie Mellon University, 1976.
[7] J. Edmonds, Paths, trees, and flowers, Can. J. Math. 17 (1965) 449–467, https://doi.org/10.4153/CJM-1965-045-4.
[8] J. Edmonds, E.L. Johnson, Matchings, Euler tours and the Chinese postman problem, Math. Program. 5 (1973) 88–124.
[9] O. Ergun, G. Kuyzu, M. Savelsbergh, Reducing truckload transportation costs through collaboration, Transp. Sci. 41 (2) (2007) 206–221, https://doi.org/10.1287/trsc.1060.0169.
[10] O. Ergun, G. Kuyzu, M. Savelsbergh, Shipper collaboration, Comput. Oper. Res. 34 (6) (2007) 1551–1560, https://doi.org/10.1016/j.cor.2005.07.026.
[11] C.G. Fernandes, C.N. Lintzmayer, P.F.S. Moura, Approximations for the Steiner Multicycle Problem, in: A. Castañeda, F. Rodríguez-Henríquez (Eds.), LATIN 2022: Theoretical Informatics, Springer International Publishing, Cham, 2022, pp. 188–203.
[12] A.M. Frieze, G. Galbiati, F. Maffioli, On the worst-case performance of some algorithms for the asymmetric traveling salesman problem, Networks 12 (1) (1982) 23–39, https://doi.org/10.1002/net.3230120103.
[13] D. Hartvigsen, Finding triangle-free 2-factors in general graphs, J. Graph Theory (2024) 1–82, https://doi.org/10.1002/jgt.23089.
[14] D. Hartvigsen, An extension of matching theory, Ph.D. thesis, Department of Mathematics, Carnegie Mellon University, Pittsburgh, PA, USA, 1984, https://david-hartvigsen.net/?page_id=33.
[15] D. Hartvigsen, Y. Li, Polyhedron of triangle-free simple 2-matchings in subcubic graphs, Math. Program. 138 (1–2) (2013) 43–82, https://doi.org/10.1007/s10107-012-0516-0.
[16] K. Jain, A factor 2 approximation algorithm for the generalized Steiner network problem, Combinatorica 21 (1) (2001) 39–60.
[17] C.N. Lintzmayer, F.K. Miyazawa, P.F.S. Moura, E.C. Xavier, Randomized approximation scheme for Steiner Multi Cycle in the Euclidean plane, Theor. Comput. Sci. 835 (2020) 134–155, https://doi.org/10.1016/j.tcs.2020.06.022.
[18] L. Lovász, M.D. Plummer, Matching Theory, North-Holland Mathematics Studies, vol. 121, Elsevier, 1986.
[19] S. Micali, V.V. Vazirani, An $\mathcal{O}(\sqrt{|V|}|E|)$ algorithm for finding maximum matching in general graphs, in: 21st Annual Symposium on Foundations of Computer Science (SFCS 1980), IEEE, 1980, pp. 17–27.
[20] J.B. Orlin, Max flows in O($nm$) time, or better, in: Proc. of the 45th Annual ACM Symposium on Theory of Computing (STOC), 2013, pp. 765–774.
[21] C.H. Papadimitriou, M. Yannakakis, The Traveling Salesman Problem with distances one and two, Math. Oper. Res. 18 (1) (1993) 1–11, https://doi.org/10.1287/moor.18.1.1.
[22] V.N.G. Pereira, M.C.S. Felice, P.H.D.B. Hokama, E.C. Xavier, The Steiner Multi Cycle Problem with applications to a collaborative truckload problem, in: 17th International Symposium on Experimental Algorithms (SEA'2018), 2018, pp. 26:1–26:13.
[23] D.J. Rosenkrantz, R.E. Stearns, P.M. Lewis, An analysis of several heuristics for the traveling salesman problem, SIAM J. Comput. 6 (1977) 563–581.
[24] J.J. Salazar-González, The Steiner cycle polytope, Eur. J. Oper. Res. 147 (3) (2003) 671–679, https://doi.org/10.1016/S0377-2217(02)00359-4.
[25] A. Schrijver, Combinatorial Optimization: Polyhedra and Efficiency, Springer-Verlag, 2003.
[26] O. Svensson, J. Tarnawski, L.A. Végh, A constant-factor approximation algorithm for the asymmetric traveling salesman problem, J. ACM 67 (6) (2020), https://doi.org/10.1145/3424306.
[27] V. Traub, J. Vygen, An improved approximation algorithm for the asymmetric traveling salesman problem, SIAM J. Comput. 51 (1) (2022) 139–173, https://doi.org/10.1137/20M1339313.
[28] W.T. Tutte, A short proof of the factor theorem for finite graphs, Can. J. Math. 6 (1954) 347–352.
[29] V.V. Vazirani, Approximation Algorithms, Springer, 2002.
[30] Z. Xu, B. Rodrigues, A 3/2-approximation algorithm for the multiple TSP with a fixed number of depots, INFORMS J. Comput. 27 (4) (2015) 636–645, https://doi.org/10.1287/ijoc.2015.0650.