Finding Triangle-free 2-factors in General Graphs

David Hartvigsen[*]

February 25, 2024

Abstract

A 2-factor in a graph G is a subset of edges M such that every node of G is incident with exactly two edges of M. Many results are known concerning 2-factors including a polynomial-time algorithm for finding 2-factors and a characterization of those graphs that have a 2-factor. The problem of finding a 2-factor in a graph is a relaxation of the NP-hard problem of finding a Hamilton cycle. A stronger relaxation is the problem of finding a triangle-free 2-factor, that is, a 2-factor whose edges induce no cycle of length 3. In this paper, we present a polynomial-time algorithm for the problem of finding a triangle-free 2-factor as well as a characterization of the graphs that have such a 2-factor and related min-max and augmenting path theorems.

Keywords: combinatorial optimization; matchings

1 Introduction

Let $G = (V, E)$ be an undirected graph with no parallel edges or loops. A subset $M \subseteq E$ is called a *simple* 2*-matching* if every node of G is incident with at most 2 edges of M. Hence the edges in a simple 2-matching induce paths and cycles in G. A simple 2-matching M is called a *2-factor* if every node of G is incident with exactly 2 edges of M. The number of edges in a simple 2-matching is its *cardinality*.

A simple 2-matching (2-factor) is called k*-restricted*, for $k \geq 2$ and integer, if its edges induce no cycles of length k or less. We let P_k denote the problem of finding a maximum cardinality k-restricted simple 2-matching. Hence, P_2 is the problem of finding a maximum cardinality simple 2-matching and P_3 is the problem of finding a maximum cardinality simple 2-matching that induces no cycles of length 3 (called *triangles*). Note that, for a given graph $G = (V, E)$, the problems P_k for $|V|/2 \leq k \leq |V| - 1$ are equivalent to the problem of finding a Hamilton cycle in G. Thus, the problems P_k define a hierarchy of relaxations of the Hamilton cycle problem. This hierarchy was introduced by

[*]Hartvigsen.1@nd.edu University of Notre Dame, Notre Dame, IN 46556 USA

Fisher, Nemhauser, and Wolsey [20] and further developed by Cornuéjols and Pulleyblank [13, 14]. (More discussion of this work appears below.)

Problems involving simple 2-matchings have been extensively studied. A key result is a polynomial-time algorithm solving P_2 (using a reduction due to Tutte [50] to the classical problem of finding a maximum cardinality matching (see Edmonds [18])). There is also a well-known "good" characterization of those graphs that have a 2-factor (see Belck [3] and Gallai [22]; for a generalization see Tutte [49]). In general, we refer to this as a Tutte-type theorem. Also known is a min-max theorem that characterizes maximum cardinality simple 2-matchings; see [43] for the statement and a derivation that uses results in [48, 50, 8]. In general, we refer to this as a Tutte-Berge-type theorem. See Schrijver [43] for a thorough history of this body of work.

The problems P_k, for $k \geq 5$, are known to be NP-hard. (See the following section for details.) A polynomial-time algorithm for P_3 was presented in the author's Ph.D. thesis [26]. (It has never been submitted for publication.) The complexity of problem P_4 is open. Complexity results for problems similar to the P_k problems appeared in Hell, Kirkpatrick, Kratochvíl, and Kříž [30].

In this paper, we focus on the problem P_3. We refer to 3-restricted simple 2-matchings (2-factors) as *tri-free simple 2-matchings (2-factors)*. Our main result is a polynomial-time algorithm for P_3. (It is significantly different from the algorithm in [26].) Note that, if a graph contains a tri-free 2-factor, then our algorithm finds one. As by-products of this algorithm, we obtain two characterizations of maximum cardinality tri-free simple 2-matchings. The first characterization is an "augmenting path" theorem in the style of the well-known result for matchings due to Petersen [40] and Berge [7]. The augmenting path theorem for matchings is easy to state and has an elementary (i.e., non-algorithmic) proof. Our augmenting path theorem for tri-free simple 2-matchings is also easy to state, but our proof uses the validity of the algorithm; an elementary proof appears in [42]. The second characterization is a min-max Tutte-Berge-type theorem. We also obtain a characterization of the graphs that have a tri-free 2-factor (a Tutte-type theorem). The statements of the latter two theorems make use of a new type of graph called the "tri-blossom cluster." These graphs are a generalization of graphs with an odd number of nodes (from the study of classical matchings) and blossoms (from the study of simple 2-matchings). (See [43].) Tri-blossom clusters are closely related to the clique tree inequalities introduced by Grötschel and Pulleyblank [25] and the more general bipartition inequalities introduced by Boyd and Cunningham [10] (both of which provide facets for the travelling salesman polytope on complete graphs). The algorithm follows the general format of Edmonds' algorithm for matchings, but is elaborated in a number of ways. A key elaboration consists of two new constructs called "models" and "substructures."

Related work is surveyed in Section 1.1. A high-level, intuitive overview of the algorithm and some of our results is given in Section 1.2. Finally, the organization of the paper is discussed in Section 1.3.

1.1 Related work

In this section we present an overview of work related to the P_k hierarchy of problems. In addition to work on the P_k problems, we discuss problems involving 2-factors, edge weights, special classes of graphs, a closely related matching problem, and jump systems (a generalization of matroids). Let us start with a few definitions.

Let Q_k denote the problem of finding a k-restricted 2-factor in a graph. For a graph $G = (V, E)$, let $w : E \rightarrow \mathbb{R}$. We define the *weight* of a simple 2-matching M in G as $\sum(w(e) : e \in M)$. Let *weighted-P_k* (*weighted-Q_k*) denote the problem of finding a maximum weight k-restricted simple 2-matching (2-factor) in a graph. Cycles of length 4 and 5 are referred to as *squares* and *pentagons*, respectively.

Let us first consider what is known about the complexity of weighted-P_k and weighted-Q_k. Polynomial-time algorithms for weighted-P_2 and weighted-Q_2 are known along with the associated polytopes (see Edmonds [17]; an algorithmic implementation was reported in [19]; details appear in [43]). The complexities of weighted-P_3 and weighted-Q_3, in general graphs, are open. It has been observed that weighted-Q_4 is NP-hard for bipartite graphs by Király (see [21]), Russell [42], and Cunningham [15]; hence, weighted-Q_4 is NP-hard for general graphs. Papadimitriou showed (see [13] for the proof) that Q_5 is NP-hard. It was shown in [28] that P_k, for $k \geq 5$, is NP-hard for a special class of graphs (see below), hence it is NP-hard for general graphs, which implies it is NP-hard for weighted-P_k, for $k \geq 5$. Finally, Cunningham and Wang [16] have studied, on complete graphs, the polytopes whose extreme points are the incidence vectors of k-restricted 2-factors (although no complete characterization is known for $k \geq 3$).

The weighted-Q_k problems can be used to find increasingly accurate, as k increases, approximate solutions to the NP-hard problem of finding a maximum weight Hamilton cycle in a complete graph, where the weights are non-negative (see [20]). In particular, a solution to weighted-Q_k, in this situation, can be transformed into a Hamilton cycle whose weight is at least $\frac{k}{k+1}$ times the maximum weight of a Hamilton cycle. Hence, it is of interest to look for a polynomial-time algorithm for weighted-Q_3 (an open problem). This paper (which solves P_3, hence Q_3) may provide a step in that direction.

Results related to these hierarchies have been reported for the special cases of cubic and subcubic graphs; i.e., graphs with degree 3 at every node and graphs with maximum degree 3, respectively. Vornberger [51] was the first to consider these sorts of problems. He discovered, for cubic graphs, polynomial-time algorithms for Q_3, weighted-Q_3 and the problem of finding a 2-factor in a cubic graph with no squares. He also showed the following two problems in cubic graphs are NP-hard: finding a 2-factor with no pentagons and finding a minimum (or maximum) weight 2-factor with no squares. Based on this work, polynomial-time algorithms for P_3 and P_4 on subcubic graphs appeared in [28] (which imply the analogous results for the problems Q_k on subcubic graphs). Additionally, [28] contains a proof (again expanding on work in [51]) that P_5, for $k \geq 5$, is NP-hard on subcubic graphs (implying the same result for general graphs, as reported above). That paper also contains a Tutte-type theorem and a Tutte-Berge-type theorem for P_3 and P_4 on subcubic graphs.

In [29], polynomial-time algorithms were given for weighted-Q_3 and weighted-P_3 on subcubic graphs, along with the associated polytopes. (The algorithm and polytope for weighted-P_3 are considerably more complex than the algorithm and polytope for weighted-Q_3.) Subsequently, for weighted-P_3 on subcubic graphs, a slightly more general algorithm and polytope were presented by Bérczi [4] and a different, simpler algorithm was discovered by Kobayashi [32].

Independently of the work in [28], Bérczi and Végh [6] presented a polynomial-time algorithm and Tutte-Berge-type theorem for a generalization of P_4 in subcubic graphs and Bérczi and Kobayashi [5] presented a polynomial-time algorithm for finding a maximum cardinality square-free simple 2-matching in subcubic graphs (as well as a polynomial-time algorithm for a special weighted version of this problem).

Closely related results appear in a paper of Boyd, Iwata, and Takazawa [9]. They present a polynomial-time algorithm for finding a minimum weight 2-factor in a bridgeless cubic graph that covers all the 3-edge cuts; they also present a description of the associated polytope. The paper also contains a polynomial-time algorithm for finding a 2-factor in a bridgeless cubic graph that covers all the 3-edge and 4-edge cuts.

Related results are also known for bipartite graphs. Note that only the problems Q_k and P_k with k even are of interest in this case. It was shown in [27] that P_4 (hence Q_4) is polynomial-time solvable for bipartite graphs. Subsequently, a simpler algorithm was discovered by Pap [39] and a faster algorithm was discovered by Babenko [2]. As noted above, weighted-Q_4 is NP-hard for bipartite graphs. It is also known that Q_6 (hence P_6) is NP-hard for bipartite graphs (see [24]).

A Tutte-Berge-type theorem for P_4 on bipartite graphs was discovered by Király [31]. Frank [21] developed an interesting generalization of this problem. Related work has been done by Pap [39], Makai [36], and Takazawa [44, 45, 46, 47]. See also [31] for additional results.

Two additional problems have been considered where a condition is placed on the excluded small cycles. Nam [37] presented a polynomial-time algorithm for P_4, where the squares in the original graph are node-disjoint. And Kobayashi [34] presented a polynomial-time algorithm for weighted-P_3, where only a given set of pairwise edge-disjoint triangles in the input graph is prohibited.

We next define another closely related problem. For each $v \in V$, let $\delta(v)$ denote the set of edges of G incident with v. Then a *2-matching* is a vector $x \in \{0, 1, 2\}^E$ such that $x(\delta(v)) \leq 2$ for all $v \in V$. (Hence, an equivalent definition of simple 2-matchings is obtained by additionally requiring $x(e) \leq 1$ for all $e \in E$.) A 2-matching is called *perfect* if $x(\delta(v)) = 2$ for all $v \in V$. Given a weight vector $w \in \mathbb{R}^E$, the *weight* of a 2-matching x is wx. A polynomial-time algorithm for finding a maximum weight 2-matching follows from [38] (see [43] for details), a Tutte-Berge-type theorem appeared in [23] (see [43] for more detail), a Tutte-type theorem appeared in [49] (characterizing perfect 2-matchings), and a description of the associated polytope appeared in [17] (see also [41]). A

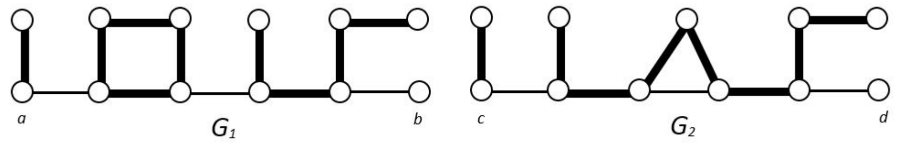


Figure 1: Two examples of augmenting paths

2-matching is called *tri-free* if no three edges with $x(e) = 1$ induce a triangle in G. Cornuéjols and Pulleyblank in [13, 14] studied tri-free 2-matchings. They showed that all the results just mentioned for 2-matchings have analogs for tri-free 2-matchings. (See also [12, 11].) A lower complexity algorithm for finding maximum weight tri-free 2-matchings by Babenko, Gusakov, and Razenshteyn appeared in [1].

Finally, we mention that Cunningham [15] showed that, for $k = 3$, the degree sequences of k-restricted simple 2-matchings induce a jump system and that this is not true for $k \geq 5$. The jump system result was shown to hold for $k = 4$ by Kobayashi, Szabó, and Takazawa [35] (see also [33]).

1.2 Overview of the paper

In this section we present a high-level, informal overview of our algorithm for P_3 and some of our results. Our goal is to help the reader develop some intuition about the basic concepts in our algorithm and main theorems for P_3. Rigorous definitions appear in subsequent sections. Although our algorithm for P_3 is long, it follows the general form of Edmonds' algorithm for matchings (and its computational complexity is typical of such algorithms). As noted above, an algorithm for P_2 can be obtained with a simple and elegant reduction to the classical matching problem. Such a reduction for P_3 is not known. Our algorithm for P_3 is an elaboration of this algorithm for P_2, as interpreted on the original graph, without the reduction. We begin this section by introducing some of the basic concepts used in such an approach for P_2 and then discuss some challenges that arise in adapting this to P_3. We then introduce a new graph, called the *tri-blossom cluster*, that arises in several of our results and is more complex than the analogous and well-known *blossoms* from the study of P_2.

A key notion in Edmonds' algorithm for matchings is the *augmenting path*. This notion has an analog for the problem P_2. Two examples illustrating this for P_2 are given in Figure 1, where the thick edges are in a simple 2-matching. Consider the paths given by the lower edges from node a to node b in G_1 and from node c to node d in G_2. We call such paths *alternating* since the edges are alternately in and out of the simple 2-matching as the path is traversed. (In general, we allow alternating paths to revisit nodes, but not edges. In particular, we allow the first and last nodes to be identical.) An alternating path is called an *augmenting path* if exchanging the edges in and out of the simple 2-matching along the path yields a new simple 2-matching with one more edge. Note that

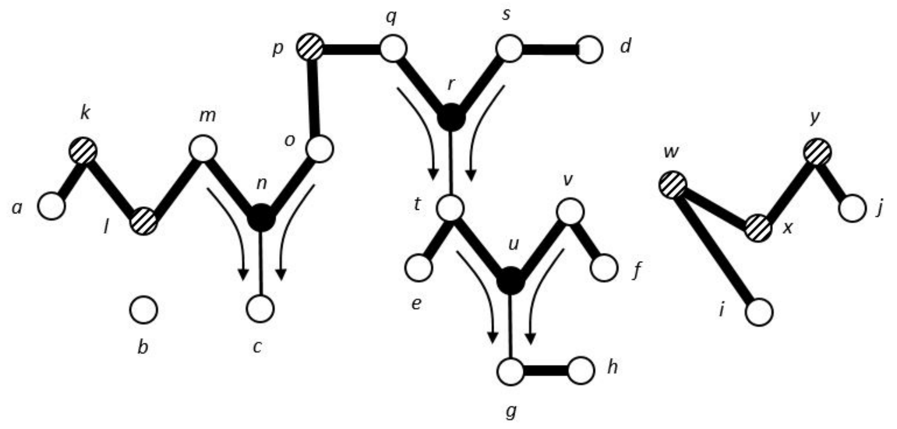


Figure 2: Example of algorithms P_2 and P_3

each of these alternating paths is an augmenting path since the endnodes are *deficient*, that is, they are incident with fewer than two edges of the simple 2-matching, and since the endedges are not in the simple 2-matching. Algorithms for P_2 can be viewed as a search for augmenting paths.

Such an algorithm also immediately yields a theorem stating that a simple 2-matching has maximum cardinality if and only if it admits no augmenting path. However, this result also has an elementary (i.e., non-algorithmic) proof making use of the reduction of P_2 to classical matchings.

Our algorithm for P_3 is based on a similar search. However, our algorithm must contend with the issue illustrated in graph G_2 in Figure 1, where the simple 2-matching resulting from the exchange contains a triangle. So, we alter the definition of *augmenting path* to be an alternating path such that an exchange yields a tri-free simple 2-matching with one more edge. Then our algorithm for P_3 is a search for such paths. As for P_2, our algorithm demonstrates that a tri-free simple 2-matching has maximum cardinality if and only if it admits no augmenting path (using the altered definition). In this case, however, we do not have an elementary proof of this result. (See [42] for an elementary proof.)

One of Edmonds' key algorithmic insights for finding augmenting paths for matchings was to "grow" an *alternating forest*. Let us illustrate an analogous construction for the problem P_2.

Consider Figure 2, which is an example of the state of the algorithm for P_2 at a point in its execution on a graph G. All the nodes of G are shown plus all the edges in a current simple 2-matching M (again illustrated with thick edges). Nodes are of three types, white, black, and striped. A few edges not in M are shown; others are not shown. Note that $\{a, b, c, \ldots, j\}$ is the set of deficient nodes. Consider the three black nodes n, r and u. Each of these nodes is shown incident with three edges of G, two of which are in M. Note that each such 3-edge subgraph has adjacent arrows that follow alternating paths, such as q, r, t. Note also that if two such subgraphs share a node (see node t), then these alternating paths can be combined to obtain a longer alternating path (e.g., $\{q, r, t, u, g\}$). Finally, note that these maximal alternating paths have one endnode that is deficient, called the *root*, and that the edges in these paths combine to yield trees. Let us consider all deficient nodes, other than c and g, to be one-node trees, where the single node is the root. Think of each root as having a trivial alternating path to itself. Thus the white nodes are the set of nodes for which we have identified an alternating path that is either trivial or starts with an edge in M and ends with an edge not in M, where this final edge is incident with a root.

The algorithm considers, one at a time, the edges not in the matching that either connect two white nodes or connect a white node and a striped node. If we consider an edge that connects two white nodes in different trees, then we have identified an augmenting path connecting the roots of those trees and an exchange is performed. For example, if edge oq is in the graph, then we have an augmenting path from c to g.

If we consider an edge that connects two white nodes in the same tree, then, in order to maintain a tree structure, we perform a "shrinking," as in Edmonds' algorithm. (This idea of shrinking is another key concept in Edmonds' algorithm.) For example, if the edge sv is in the graph, then we can shrink the cycle with nodes s, v, u, t, r to a single new node. With this, we gain new alternating paths for the shrinking black nodes (e.g., $\{r, s, v, u, g\}$) and the new shrunk node becomes part of a slightly more complex forest structure.

Finally, if we consider an edge from a white node to a striped node, then we can "grow" a tree. For example, suppose there is an edge bl. Then we make l a black node and make k a white node. We consider this a new three-edge subgraph built around the black node, but we only need to add one arrow following the alternating path $\{k, l, b\}$, since we already have a path from m to c.

These are the main ideas in an Edmonds-style algorithm for P_2 (although details have been left out, particularly involving shrunk nodes).

Our algorithm for P_3 is an elaborated version of this algorithm for P_2. Our algorithm constructs a structure analogous to the forest for P_2. This structure includes three-edge subgraphs defined by black nodes, as for P_2. But it includes a number of additional types, many having their own internal alternating paths. These subgraphs, called *substructures*, are combined to give a forest-like structure with longer, combined alternating paths. The types of substructures are described with a list of graphs called *models*, where each substructure is isomorphic to one of these models. We enumerate the models in Section 5.2 and give rules for how the substructures can be combined in Section 5.3, before presenting the algorithm.

As in the P_2 algorithm, our P_3 algorithm considers, one by one, edges not in the structure as we look for augmenting paths (using our altered definition), nodes to shrink, and opportunities to grow the structure with new substructures. (In fact, the algorithm employs a more general class of subgraphs that play the role of these edges, which we call *temporary substructures*.) For example, consider the graph in Figure 2 and the edge oq, as we did before. Note that

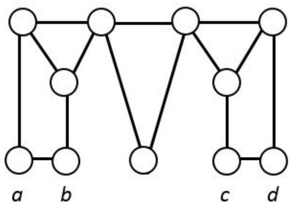


Figure 3: The role of tri-blossoms

performing an exchange on the alternating path from c to g would create the triangle o, p, q. So, instead of performing an exchange, we identify the triangle as a new substructure, where node p becomes white with an arrow indicating the alternating path $\{p, q, o\}$. (Note that adding the edges pq and qo to the forest structure already constructed is not quite a forest; but this turns out to be easy to deal with.)

Suppose we consider an edge ix in Figure 2. In the P_2 algorithm, we would make x a black node and make w and y white nodes with arrows indicating alternating paths from them to the root i. But in our P_3 algorithm, performing an exchange along the path $\{y, x, i\}$ would create the triangle w, x, i in the new simple 2-matching. So, instead, we identify the triangle w, x, i as a new substructure; we make w a white node and add an arrow indicating the alternating path $\{w, x, i\}$ and leave y unchanged. We also make x into a new type of node called "grey," indicating its special role. There are numerous other situations similar to this that must be addressed in our algorithm for P_3.

The following example illustrates another difference in solving P_3 over P_2. Consider the graph in Figure 3. Let G_1 denote the graph with edges ab and cd deleted, and let G_2 denote the graph as shown. Observe that a maximum cardinality simple 2-matching M in G_1 has value 9 (where M contains the central triangle), whereas a maximum cardinality tri-free simple 2-matching in G_1 has value 8. This type of graph plays a key role in our theorems and algorithm. It is an example of a general class of graphs we call *tri-blossom clusters.* These graphs are a generalization of the well-known *blossoms* that play an analogous role in the study of P_2. These graphs also play a key role when we consider 2-factors. In particular, note that G_2 has a 2-factor, but has no tri-free 2-factor. In general, when there is a difference between the maximum cardinalities of a simple 2-matching and a tri-free simple 2-matching in a graph, it is due to the effect of one or more tri-blossom clusters.

1.3 Organization of the paper

The paper is organized as follows. In Section 2, we present an augmenting path theorem that characterizes maximum cardinality tri-free simple 2-matchings. In Section 3 we introduce the tri-blossom cluster, which plays a key role in several of our main results. In Section 4 we present statements of Tutte-Berge-type and

Tutte-type theorems for tri-free simple 2-matchings and 2-factors, respectively. Part of the proof of the Tutte-Berge-type theorem is given and the Tutte-type theorem is seen to follow easily from the Tutte-Berge-type theorem. Section 5 contains definitions of the basic structures that are used in our algorithm for P_3 along with their key properties. The Main Algorithm for P_3 and four subroutines are given in Section 6. The validity of the algorithm is proved in Section 7. This immediately implies proofs of the augmenting path theorem and the remainder of the Tutte-Berge-type theorem. We also show in Section 7 that the algorithm has polynomial-time complexity. Section 8 contains some acknowledgements.

2 An augmenting path theorem

In this section we state an augmenting path theorem for P_3. It follows immediately from the validity of our algorithm for P_3 (see Section 7).

Consider a graph $G = (V, E)$ with a tri-free simple 2-matching M. A node v is called *deficient in* M if it is incident with 0 or 1 edge of M. An *alternating path* in G is defined as a sequence of edges $\{v_1v_2, v_2v_3, v_3v_4, \ldots, v_{m-1}v_m\}$ of G, where: no edge occurs twice; and the edges are alternately in and not in M as the sequence of edges is considered in order. Hence, a node may be traversed more than once in an alternating path; in particular, the first and last node may be identical. A single node is also considered to be a (trivial) alternating path. (When convenient, we also describe alternating paths as a sequence of nodes.) For a non-trivial alternating path P, the operation that yields the edge set $M' = M \Delta P$ is called an *exchange on* P. (Note that Δ denotes the symmetric difference operator and we treat P as a set of edges, hence an exchange simply exchanges edges in and out of M along P. Note also that, in general, M' need not be tri-free; in fact, it need not even be a simple 2-matching.) If the endedges of P are not in M and if M' is a tri-free simple 2-matching, then P is called an *augmenting path* for M. Observe that if P is an augmenting path, then the endnodes are different and both deficient, or the endnodes are identical and incident with no edge in M. In addition, $|M'| = |M| + 1$.

We have the following result, previously shown by Russell [42].

Theorem 1. *Let G be a graph with a tri-free simple 2-matching M. Then M has maximum cardinality if and only if there exists no augmenting path for M.*

3 Tri-blossom clusters

In this section we introduce the tri-blossom cluster, which is an important type of graph that occurs in our main theorems and (implicitly) in the algorithm. (See Figure 4, which contains three examples of tri-blossom clusters. See also the graph in Figure 3, with the edges ab and cd deleted.) We also present an associated inequality.

A connected graph G is called a *tri-blossom cluster* if it satisfies the following properties:

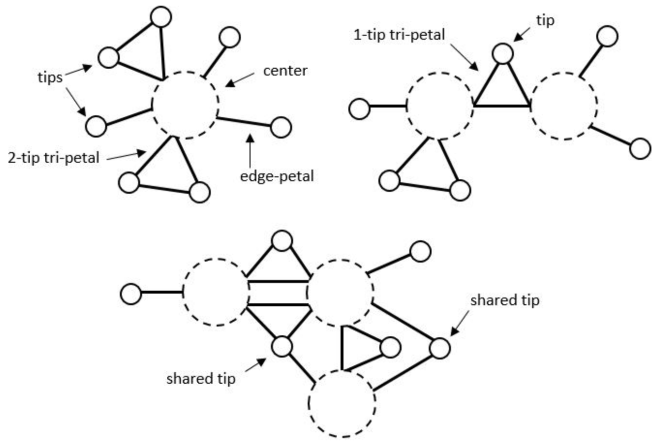


Figure 4: Three tri-blossom clusters

T1: G is the union of connected, node-induced, edge-disjoint subgraphs each of which is a *center* or *petal* and satisfies the properties below. (In Figure 4, centers are subgraphs (details not shown) contained inside the large dashed circles.)

T2: Each petal is either a single edge or a triangle.

Definition: A petal with one edge is called an *edge-petal* and a petal with three edges is called a *tri-petal*.

T3: Each center has at least two nodes.

T4: No two centers have a common node.

T5: A node in a center can be contained in at most one petal.

T6: Each center shares nodes with an odd number of petals.

T7: Each edge-petal contains exactly one node in a center; each tri-petal contains either one node in a center or two nodes in different centers.

Definition: A node in a petal and not in a center is called a *tip* for that petal. A tri-petal with one tip is called a *1-tip tri-petal*; a tri-petal with two tips is called a *2-tip tri-petal*.

T8: The subgraph obtained by deleting the tips is connected.

Observation: Properties T6 and T7 imply that a tri-blossom cluster must have at least one center and at least one petal. Note that a node can be a tip in more than one petal.

Observation: The special case of a tri-blossom cluster with one center and only edge-petals, known as a *blossom*, plays an analogous role in the study of P_2.

For a tri-blossom cluster $B = (V, E)$, let n denote the number of nodes in the centers of B and let p_e, p_{1t}, and p_{2t} denote the numbers of edge-petals, 1-tip tri-petals, and 2-tip tri-petals of B, respectively.

Let M be a tri-free simple 2-matching for B. For a node v in B, we let $deg_M(v)$, called the *degree of* v *in* M, denote the number of edges of M incident with v. For a node v in a center of B, we define the *value* of M at v to be $deg_M(v)$. For an edge-petal or 1-tip tri-petal P, we define the *value* of M at P to be $deg_M(v)$, where v is the single tip of P. For a 2-tip tri-petal P, with tips v_1 and v_2, we define the *value* of M at P to be $deg_M(v_1) + deg_M(v_2)$. Hence the value of M is at most 2 at a center node, at most 1 at an edge-petal, at most 2 at a 1-tip tri-petal, and at most 3 at a 2-tip tri-petal. These upper bounds are called the *targets* for the corresponding center nodes and petals.

We want to derive an upper bound on $\sum_{v \in V} deg_M(v)$. Let us first develop such a bound for a simpler case. A tri-blossom cluster with a single center is called a *tri-blossom.* (See the upper left graph in Figure 4.) Let $B = (V, E)$ be a tri-blossom with a tri-free 2-matching M.

Summing the targets over the petals and the nodes of the center, we obtain the following upper bound:

$$\sum_{v \in V} deg_M(v) \leq 2n + p_e + 3p_{2t} \tag{1}$$

Note: As noted above, a tip for B can be in more than one petal. So, for example, if a tip is in three edge-petals of B, then this tip contributes 3 to the right hand side of expression (1).

Because B has an odd number of petals, $p_e + 3p_{2t}$ is odd. Since $\sum_{v \in V} deg_M(v)$ is even, the following is a tighter upper bound:

$$\sum_{v \in V} deg_M(v) \leq 2n + p_e + 3p_{2t} - 1 \tag{2}$$

A tri-free simple 2-matching M is said to *saturate* a tri-blossom if inequality (2) is satisfied at equality.

Observation: A tri-free simple 2-matching M saturates a tri-blossom B if and only if the values of M at all the center nodes and petals are at their targets except one, whose value is one below its target.

We next develop a similar bound for the more general tri-blossom clusters.

Let $B = (V, E)$ be a tri-blossom cluster and let M be a tri-free 2-matching for B. Let $C_1, \ldots, C_k$ be the centers of B, where $k \geq 2$. For each center C_i we construct a tri-blossom $B_M(C_i)$, which is a subgraph of B, with center C_i as follows.

Let T be an edge-petal or a 2-tip tri-petal of B that shares a node with C_i. Then we make T a petal of the same type for $B_M(C_i)$. Let T be a 1-tip tri-petal of B that shares nodes with centers C_i and C_j. Let a, b, and c be the three edges of T, where the edge c shares its endnodes with both C_i and C_j, the edge a shares an endnode with C_i, and the edge b shares an endnode with C_j. Then M contains at most two edges of T.

We next apply the following transformation; it creates two edge-petals from T, one for $B_M(C_i)$ and one for $B_M(C_j)$:

1. If M contains only c, make c an edge-petal for $B_M(C_i)$ and b an edge-petal for $B_M(C_j)$. Disregard a and b for $B_M(C_i)$ and disregard a and c for $B_M(C_j)$.

2. If M does not contain c, make a an edge-petal for $B_M(C_i)$ and b an edge-petal for $B_M(C_j)$. Disregard b and c for $B_M(C_i)$ and disregard a and c for $B_M(C_j)$.

3. If M contains a and c, make a an edge-petal for $B_M(C_i)$ and c an edge-petal for $B_M(C_j)$. Disregard b and c for $B_M(C_i)$ and disregard a and b for $B_M(C_j)$.

4. If M contains b and c, make c an edge-petal for $B_M(C_i)$ and b an edge-petal for $B_M(C_j)$. Disregard a and b for $B_M(C_i)$ and disregard a and c for $B_M(C_j)$.

Observe that the resulting graphs $B_M(C_i)$ are, indeed, tri-blossoms, hence, each $B_M(C_i)$ satisfies expression (2). Also, observe that the graphs $B_M(C_i)$ contain all the edges of M. Thus, if we let n denote the total number of nodes in the centers, and let k denote the number of centers, then, by summing the expressions (2) over the $B_M(C_i)$, we obtain the following expression for the tri-blossom cluster B:

$$\sum_{v \in V} deg_M(v) \leq 2n + p_e + 2p_{1t} + 3p_{2t} - k. \tag{3}$$

To see this, note that each 1-tip tri-petal in B that contains nodes from, say, centers C_i and C_j, becomes two edge-petals, one for $B_M(C_i)$ and one for $B_M(C_j)$. Each of these edges contributes 1 to the corresponding right hand side of expression (2) in the p_e term for each of $B_M(C_i)$ and $B_M(C_j)$. In the right hand side of expression (3), these contributions are replaced and combined in the $2p_{1t}$ term.

A tri-free simple 2-matching M is said to *saturate* a tri-blossom cluster B if inequality (3) is satisfied at equality.

Observation 1: A tri-free simple 2-matching M saturates a tri-blossom cluster B if and only if M saturates each of the tri-blossoms $B_M(C_i)$.

We let $Val(B)$ denote the quantity on the right hand side of expression (3) for a tri-blossom cluster B.

4 A min-max theorem

In this section we state our min-max (or Tutte-Berge-type) theorem for tri-free simple 2-matchings. We also prove that the max is less than or equal to the min. Showing equality is more difficult. It is proved in Section 7 at the same time we prove the validity of our algorithm.

For a graph $G = (V, E)$ with $V' \subseteq V$, let $\delta(V')$ denote the set of edges of G with exactly one endnode in V'; let $\gamma(V')$ denote the set of edges of G with both endnodes in V'; let $G(V') = (V', \gamma(V'))$; let $E(G) = E$; let $E[V_1, V_2]$, for V_1, V_2 disjoint, non-empty subsets of V, denote the set of edges with one endnode in V_1 and the other in V_2; and let $G - V'$ denote the graph obtained from G by deleting the nodes in V' (together with all edges incident with a node in V').

For a graph $G = (V, E)$, let $U \subseteq V$ and $W \subseteq V - U$. Let $\mathcal{T}$ be a set of pairwise edge-disjoint triangles in $G(V - U)$, where each triangle contains one, two, or three nodes in W. Let $\mathcal{T}_3$ denote the triangles of $\mathcal{T}$ that contain 3 nodes in W.

Let $\mathcal{C}$ contain an arbitrary subset of the connected components of $G(V - U - W)$. For each $C \in \mathcal{C}$, consider the subgraph that is the union of C, the triangles of $\mathcal{T}$ that share one or two nodes with C, and the remaining edges from C to W (with their endnodes). Let $\mathcal{C}^*$ denote the collection of such subgraphs that are tri-blossom clusters where the incident triangles of $\mathcal{T}$ are the tri-petals and the edges from C to W, that are not in a triangle of $\mathcal{T}$, are the edge-petals. Note that the tips of all such petals are in W. Let $CN(\mathcal{C}^*)$ denote the nodes in the centers of the graphs in $\mathcal{C}^*$ (i.e., the nodes of $V - U - W$ contained in the graphs of $\mathcal{C}^*$). Let $R = V - U - W - CN(\mathcal{C}^*)$.

Example: Figure 5 illustrates a node partition U, W, $CN(\mathcal{C}^*)$, R. As usual, small circles illustrate nodes; in this example, the large circles with a dashed border contain connected subgraphs. Edges incident with the nodes of U are not shown. The five triangles in the figure are the set $\mathcal{T}$, where two of the triangles are in $\mathcal{T}_3$. Observe that there are four connected components of $G(V - U - W)$, defined by the subgraphs: a; b and c; d and e; and f. Let us arbitrarily choose $\mathcal{C}$ to contain the three connected components defined by a; b and c; and d and e. Two of these connected components, defined by a, b, and c, yield two tri-blossom clusters, which become $\mathcal{C}^*$. Note that the connected component defined by d and e does not yield a a tri-blossom cluster because d and e are each incident with an even number of petals. Thus, we have $CN(\mathcal{C}^*)$ equal to the nodes in the subgraphs a, b, and c, which leaves R equal to the nodes in the subgraphs d, e, and f. If we had chosen the connected component f of $G(V - U - W)$ to also be in $\mathcal{C}$, then we would have had a third tri-blossom cluster defined by f in $\mathcal{C}^*$ (and we would have the nodes of f in $CN(\mathcal{C}^*)$ instead of R).

Select a set $R^* \subseteq W$ with the following properties: (1) each node of R^* occurs in exactly one triangle of $\mathcal{T}$ and that triangle contains no other node in R^*; (2) that triangle is in $\mathcal{T}_3$ or is a 2-tip tri-petal of a tri-blossom cluster in $\mathcal{C}^*$; and (3) each node in R^* is adjacent only to the other two nodes in its triangle

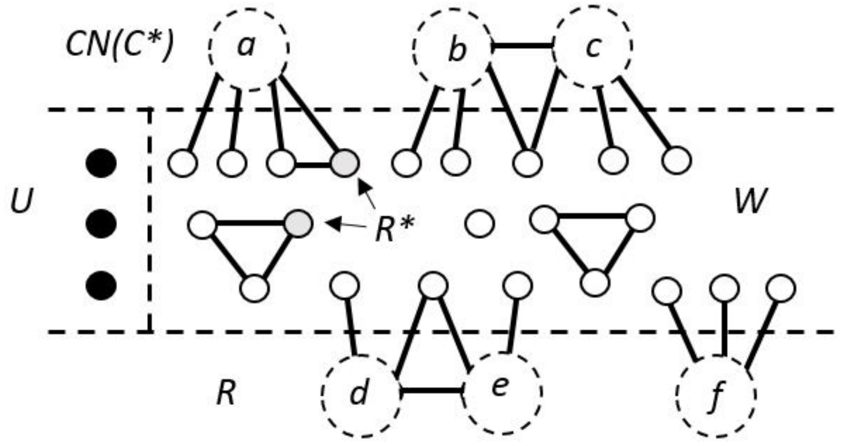


Figure 5: Node partition

of $\mathcal{T}$ and nodes in $U \cup R \cup R^*$. For each $r \in R^*$, let T_r denote the triangle of $\mathcal{T}$ that contains it and let r, r_a, r_b denote the nodes of T_r. Let $\mathcal{C}^{**}$ denote the tri-blossom clusters obtained by replacing each tri-petal in $\mathcal{C}^*$ of the form T_r with the edge-petal $r_a r_b$. Note that R^* need not be uniquely determined.

Let $\mathcal{T}_3^*$ denote the triangles in $\mathcal{T}_3$ that do not contain a node in R^*.

Example: In Figure 5 there are two nodes in R^*. One is in a 2-tip tri-petal of a tri-blossom cluster in $\mathcal{C}^*$ and the other is in a triangle of $\mathcal{T}_3$. There are two tri-blossom clusters in $\mathcal{C}^{**}$. One is obtained from the tri-blossom cluster defined by a by replacing its tri-petal with an edge-petal obtained by deleting its node in R^*. The other is the same one as before, defined by b and c. There is one triangle in $\mathcal{T}_3^*$ (the one furthest to the right).

Let U, W, $\mathcal{T}$, $CN(\mathcal{C}^*)$, and R^* be a selection of nodes and triangles of G as described above. We define the following partition of the edges of G.

- E_1 is the set of edges incident with one or two nodes of U.
- E_2 is the set of edges in the tri-blossom clusters of $\mathcal{C}^{**}$.
- $E_3 = \gamma(R \cup R^*) \cup E[R \cup R^*, V - U - (R \cup R^*)]$.
- E_4 is the set of remaining edges: $\gamma(W - R^*) - E_2$.

The following definitions are closely based on the above partition. In particular, for any tri-free simple 2-matching M of G, we show, towards proving our min-max theorem, that $\sum_{v \in V} deg_{M \cap E_i}(v) \leq \alpha_i$, for $i = 1, \ldots, 4$ (see Proposition 1).

- $\alpha_1 = 4|U|$.
- $\alpha_2 = \sum_{C \in \mathcal{C}^{**}} Val(C)$.
- $\alpha_3 = 2|R \cup R^*| + |E[R \cup R^*, V - U - (R \cup R^*)]|$.
- $\alpha_4 = 2|E_4|$.
- $\alpha_5 = |R^*| + 2|\mathcal{T}_3^*|$

Theorem 2. *Let G be a graph. The maximum cardinality of a tri-free simple 2-matching equals the minimum value of*

$$\frac{1}{2}(\alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 - \alpha_5)$$

taken over all choices of U, W, $\mathcal{T}$, $\mathcal{C}$, and R^*, using the definitions given above.

Note: Suppose the set $\mathcal{T}$ is empty. Then R^* is empty, $\alpha_5 = 0$, and the statement of Theorem 2 reduces to the statement of a min-max theorem for simple 2-matchings given in Schrijver [43]. (See Theorem 32.1 on page 562, where we take $b = 2$, $c = 1$, and we split the summation to obtain $\alpha_2 + \alpha_3$).

The following theorem follows immediately from Theorem 2. It is our Tutte-type theorem.

Theorem 3. *A graph G has a tri-free 2-factor if and only if*

$$|V| \leq \frac{1}{2}(\alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 - \alpha_5)$$

taken over all choices of U, W, $\mathcal{T}$, $\mathcal{C}$, and R^*, using the definitions given above.

Example: Consider the graph in Figure 3. Let $U = \emptyset$, let W be the five nodes at the bottom, and let $\mathcal{T}$ be the triangle in the middle. Let $CN(\mathcal{C}^*)$ denote the node set obtained by deleting the nodes in W. The triangles to the left and right are each a center for a tri-blossom cluster; the triangle in the middle is a 1-tip tri-petal for the cluster and the remaining four vertical edges are edge-petals. Note that $R = R^* = \emptyset$. We have $\alpha_1 = \alpha_3 = \alpha_5 = 0$ and $\alpha_2 + \alpha_4 = 16 + 4$. We see that the condition in Theorem 3 is violated since $|V| = 11 \not\leq \frac{1}{2}(16 + 4)$. Hence this graph has no tri-free 2-factor.

We next state and prove the first step in proving Theorem 2.

Proposition 1. *Let G be a graph. The maximum cardinality of a tri-free simple 2-matching is at most the minimum value of*

$$\frac{1}{2}(\alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 - \alpha_5)$$

taken over all choices of U, W, $\mathcal{T}$, and R^*, using the definitions given above.

Proof. Let M be a tri-free simple 2-matching for G. We show that

$$\sum_{v \in V} deg_M(v) \leq \alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 - \alpha_5$$

. We begin by showing the following.

Claim 1. *$\sum_{v \in V} deg_{M \cap E_i}(v) \leq \alpha_i$, for $i = 1, \ldots, 4$.*

Proof. Consider the case $i = 1$. The number of edges of M in E_1 is clearly maximized if each edge of M has one endnode in U and one endnode not in U, and each node in U is incident with two edges in M. Hence, $\sum_{v \in V} deg_{M \cap E_1}(v) \leq 4|U|$.

Consider the case $i = 2$. This follows immediately from our previous calculation that is summarized in expression (3).

Consider the case $i = 3$. This follows from the observation that $2|R \cup R^*|$ is the maximum value of $\sum_{v \in R \cup R^*} deg_{M \cap E_3}(v)$ and $|E[R \cup R^*, V - U - (R \cup R^*)]|$ is the maximum additional contribution to $\sum_{v \in V} deg_{M \cap E_3}(v)$ if all the edges of $E[R \cup R^*, V - U - (R \cup R^*)]$ are also in M.

Consider the case $i = 4$. This follows by simply assuming that all edges of E_4 are in M. This concludes the proof of the claim.

□

Let us finish proving the proposition by considering the "correction factor" α_5. Every triangle of $\mathcal{T}_3^*$ has all three of its edges contributing 2 to α_4. However, only two of the three edges can be in M, hence we subtract 2 from $\alpha_1 + \alpha_2 + \alpha_3 + \alpha_4$ for each such triangle.

Finally, consider a node $r \in R^*$ and its associated triangle T_r. Suppose the edge $r_a r_b$ is an edge-petal for a tri-blossom cluster $C \in \mathcal{C}^{**}$. In our count for α_3 we count both rr_a and rr_b as being in M. If this is the case, then $r_a r_b$ cannot be in M. This then implies that C cannot be saturated by M since the petal $r_a r_b$ and the endnode of $r_a r_b$ in a center of C are both below their targets of 1 and 2, respectively. Hence, the maximum value of $\sum_{v \in V} deg_{M \cap E(C)}(v)$ is at least 2 less than $Val(C)$ (since this sum is, of course, even) so, in this case, we can subtract 2 from $\alpha_1 + \alpha_2 + \alpha_3 + \alpha_4$ for each such node r. Suppose instead that only one of rr_a and rr_b is in M. Then it is possible that the $\sum_{v \in V} deg_{M \cap E(C)}(v)$ is $Val(C)$ and we can subtract 1 from $\alpha_1 + \alpha_2 + \alpha_3 + \alpha_4$ for each such node. So we subtract the minimum of these two possibilities with α_5. Finally, suppose the edge $r_a r_b$ is in $\gamma(W - R^*)$. In our count for α_3 we count both rr_a and rr_b as being in M. If this is the case, then $r_a r_b$ cannot be in M so we can subtract 2 from $\alpha_1 + \alpha_2 + \alpha_3 + \alpha_4$ for each such node r. Conversely, suppose $r_a r_b$ is in M. Then at most one of rr_a and rr_b can be in M. Since we counted 1 for each of these edges in α_3, we can subtract 1 from $\alpha_1 + \alpha_2 + \alpha_3 + \alpha_4$. So we again subtract the minimum of these two possibilities with α_5.

□

5 Structures for the algorithm

In this section we present some key concepts and structures used in the algorithm. Our key concepts include the standard matching concept of *shrinking*, employed in Edmonds' classical matching algorithm [18], as well as the new concepts of *models* and *substructures*.

5.1 Basic definitions

Let U be a non-empty subset of nodes of a graph $G = (V, E)$ and let M be a simple 2-matching for G. A graph G' is obtained from G by *shrinking* U as follows: Delete the nodes of U from V (with any incident edges) and add a new node u; then, for each edge $u'v \in E$, where $u' \in U$ and $v \notin U$, add to G' an edge uv. The node u is called a *shrunk node* of G' and each edge in G' has a corresponding edge in G. Observe that G' may have parallel edges. If there are parallel edges between nodes u and v of G', we may refer to a specific such edge as uv, when the context allows. We refer to the edges of G' whose corresponding edges of G are in M as the edges of G' in M. In the course of the algorithm, we perform this shrinking operation recursively.

Let $G' = (V', E')$ be obtained from G by a sequence of shrinkings and let u be a shrunk node in G'. Then the nodes of G that are deleted in the course of shrinking u are referred to as the *nodes inside* u. Similarly, the edges of G that connect two nodes inside u are said to be the *edges inside* u. We also refer to the subgraph of G induced by the nodes and edges inside u as the *graph inside* u. If uv is an edge in G', $u'v'$ is the corresponding edge in G, and u is shrunk, then we refer to u' as the *endnode of* uv *inside* u. All nodes and edges that are not inside u are said to be *nodes* and *edges outside* u.

During the main part of the algorithm we have a tri-free simple 2-matching M in the original graph G and a graph $\tilde{G}$ obtained from G by a sequence of shrinkings. As we will see when proving the validity of the algorithm, the graph inside each shrunk node induces a center of a tri-blossom cluster in G.

Each deficient non-shrunk node of $\tilde{G}$ is called a *root node of* $\tilde{G}$. In addition, each shrunk node of $\tilde{G}$ with a deficient node inside is also called a *root node of* $\tilde{G}$.

5.2 Models

A key to the algorithm is a collection of graphs, called *models*, which are depicted in Figures 6 to 9 along with some key properties (based on the terminology and conventions F1-F6 listed below). Think of each model as being obtained from a graph with a tri-free simple 2-matching M^* by possibly shrinking some node sets.

The following important points are made in Figures 6 to 9.

- Some depictions of a model can represent a few variations, where some nodes are allowed to be either shrunk or not.

- Each model, and each of its variations, is either *standard* or *temporary*. (This notion is important in the algorithm.) All models are *standard* except where indicated in the figures.
- Most of the temporary models have two special nodes called *endnodes*.

In discussing models, we use the following terminology and conventions (F1-F6) in reference to Figures 6 to 9.

Terminology and conventions for figures:

F1: Shrunk nodes are illustrated with a large circle with a white interior and a dashed border (such as node a in M3). Non-shrunk nodes are illustrated by small circles. Each non-shrunk node is either *white*, *black*, *grey*, or *striped*. (These types of nodes play various roles in the algorithm.) For example, a in M2 is white; c in M8 is black; c in M13 is grey; and the nodes in M4 are striped.

F2: Thick solid lines (such as ab in M2) illustrate edges in M^* and thin solid lines (such as ab in M6) illustrate edges not in M^*.

F3: Nodes with a grid interior (such as node c in M5) and the dashed thick lines incident with them (such as ac in M5) represent edges, but they are **not** in the models and play no role in the definition of models. They are explained in Section 5.4 with the definition of *border edges* and Proposition 2.

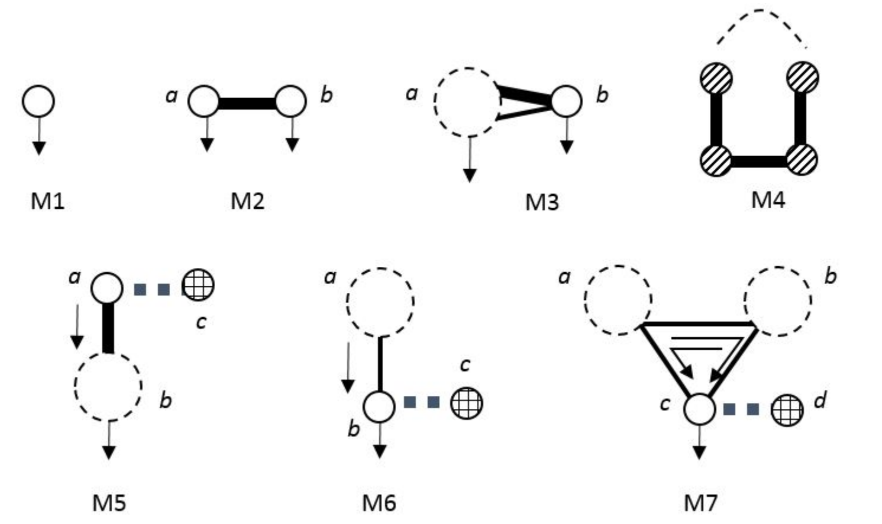
F4: Some models have an illustration that represents several variations, where only one variation is shown. In particular, in some cases, the white nodes in the figures can be shrunk nodes. The details are discussed in Figures 6 to 9.

F5: If v is a white or shrunk node in model graph N, then v is either an *in-node of N* or an *out-node of N* (not both). Out-nodes are illustrated by an arrow (solid or dashed) pointing away from the graph (i.e., not along an edge of the graph). In-nodes are illustrated by a solid (non-dashed) arrow pointing along a path in the graph; this path ends at an out-node.

Examples: Node a is an out-node in M2; b is an out-node in M5; b is an out-node in M9; and e is an out-node in M17 and M20. Also, a is an in-node in M5 with the arrow pointing along the path $\{a, b\}$; and a is an in-node in M7 with the arrow pointing along the path $\{a, b, c\}$. An out-node need not have such a path that ends at it.

F6: The dashed thin arrows in models M17 and M20 play a special role and are discussed below when *substructures* are introduced in Section 5.3. (See substructure property S11.)

Models satisfy the following additional properties:



M1: A single white node.

M2: 0, 1, or 2 of the nodes can be shrunk. If two are shrunk, the model is called ***temporary*** with ***endnodes*** *a* and *b*.

M3: Node *b* can be white or shrunk. The two edges shown are incident with different nodes inside shrunk node *a*. This model is called ***temporary*** with ***endnodes*** *a* and *b*.

M4: A cycle with at least 4 edges, all in M*. All nodes are striped.

M5: Node *a* can be white or shrunk.

M6: Node *b* can be white or shrunk.

M7: Node *c* must be white.

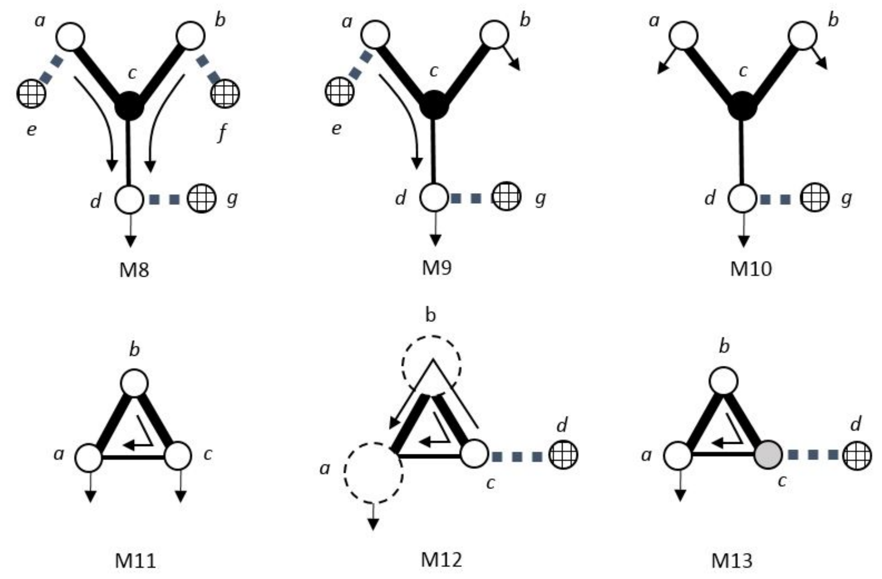
Figure 6: Models M1-M7

P1: The inside structure of shrunk nodes is not specified except as follows. The two edges connecting nodes a and b in M3 are incident with different nodes inside a; in M18, the two edges connecting nodes b and c are incident with different nodes inside b and the two edges in M^* incident with b are incident with the same node inside b. In all other cases, if two edges are incident with the same shrunk node (e.g., ab and ac in M7), then they are incident with the same node inside that shrunk node.

P2: For each model, the illustrated white, black, grey, and striped nodes are distinct.

5.3 Substructures

The algorithm constructs a collection of subgraphs of $\tilde{G}$, called the *substructures of* $\tilde{G}$, say $S_1, ..., S_m$. Roughly speaking, the substructures in $\tilde{G}$ are an edge-



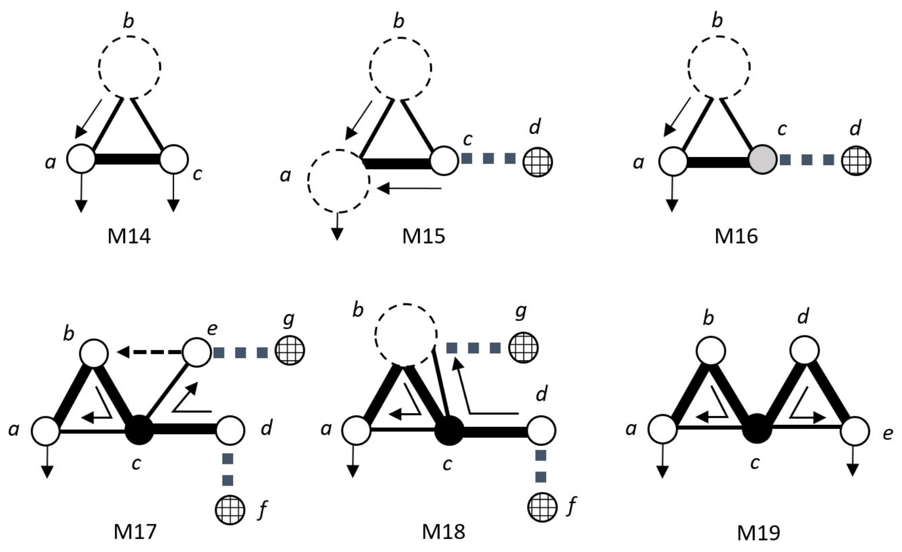
M8, M9, M10: Nodes *a,b* must be white in M10; otherwise *a, b,* and *d* can be white or shrunk. If *b* is shrunk, in M9, then this model is called ***temporary*** with ***endnodes*** *b* and *d*.

M11: 0, 1, 2, or 3 nodes can be shrunk. If three nodes are shrunk, this model is called ***temporary*** with ***endnodes*** *a* and *c*.

M12: Node *c* must be white.

M13: At most one of *a* and *b* is shrunk. Node *c* is grey.

Figure 7: Models M8-M13



M14: 0, 1, or 2 of *a* and *c* can be shrunk. If both nodes are shrunk, this model is called ***temporary*** with ***endnodes*** *a* and *c*.

M15: Node *c* must be white.

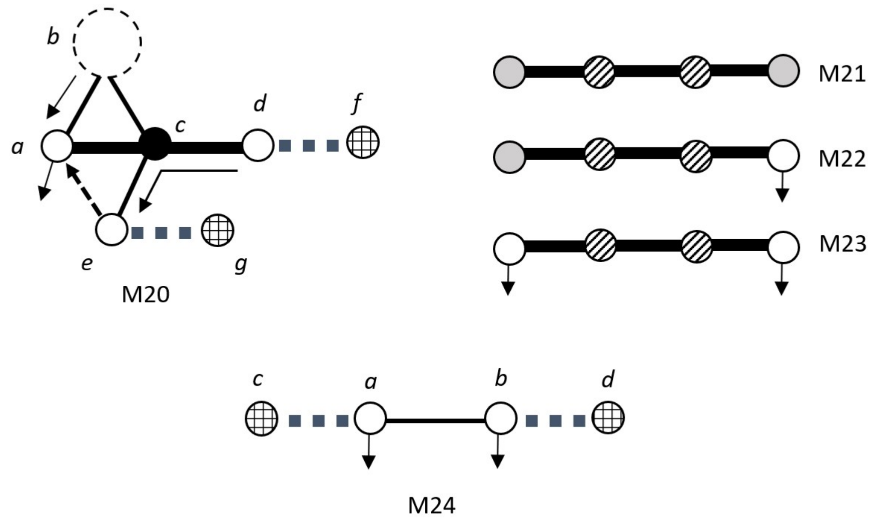
M16: Node *a* must be white. Node *c* is grey.

M17: 0, 1, or 2 of the nodes *a* and *b* can be shrunk. If both are shrunk, this model is called ***temporary***; a transformation is performed in Step 2, part 2 of the the Main Algorithm. 0, 1, or 2 of the nodes *d* and *e* can be shrunk.

M18: Node *a* must be white; node *d* can be shrunk. The two edges from *c* to *b* are incident with different nodes inside *b*.

M19: At most one of *a* and *b* can be shrunk. At most one of *d* and *e* can be shrunk.

Figure 8: Models M14-M19



M20: Node *a* can be shrunk. If it is shrunk, then this model is called ***temporary***; a transformation is performed in Step 2, part 2 of the Main Algorithm. 0, 1, or 2 of the nodes *d* and *e* can be shrunk.

M21, M22, M23: Each model is a path in *M**. The left and right nodes for M21, M22, and M23 are grey/grey, grey/white, and white/white, resp. All interior nodes are striped. M21 and M22 contain ≥0 striped nodes and M23 contains ≥1 striped node. The figure illustrates three possibilities, each with two striped nodes.

M24: 0, 1, or 2 of the nodes *a* and *b* can be shrunk. This model is called ***temporary*** with ***endnodes*** *a* and *b.*

Figure 9: Models M20-M24

disjoint collection of subgraphs of $\tilde{G}$, where each substructure is isomorphic to a model. Pairs of substructures can share white, shrunk, or grey nodes. The arrows from in-nodes to out-nodes (inherited from the corresponding models) yield paths from white and shrunk nodes to roots. Later in the paper, we see that these paths have corresponding alternating paths in G. The algorithm performs exchanges on these paths to increase the size of the current tri-free simple 2-matching. As we see in substructure property S10, the substructures have a corresponding tree structure, which has an analogous function to the alternating trees in Edmonds' algorithm [18] for matchings. The graphs G_1 in Figure 11, G_3 in Figure 12, and $\tilde{G}$ in Figure 13 illustrate collections of substructures. These examples are discussed in more detail after we list a number of properties that more precisely describe substructures.

Properties of substructures

S1: The substructures are pairwise edge-disjoint.

S2: Each node of $\tilde{G}$ is contained in at least one substructure.

S3: Each edge of M in $\tilde{G}$ is contained in a substructure.

S4: Each substructure S_i in $\tilde{G}$ is isomorphic to a model graph. (Recall that the thick dashed edges and the nodes with a grid interior in Figures 6 to 9 are not part of the model graphs.) Each node in $\tilde{G}$ has the same type in each substructure that contains it and this type is in agreement with the allowed types for each of the corresponding models; hence each node in $\tilde{G}$ has a well-defined type. Let e be an edge in a substructure S_i. Then e is in M if and only if the edge corresponding to e in the model for S_i is in the model's simple 2-matching. The edges in a substructure are incident with nodes inside shrunk nodes as in the corresponding model, according to property P1 of models.

Definition: Each white node of a substructure S_i is also called an *in-node of* S_i or an *out-node of* S_i as determined by the corresponding node in its model graph; and each in-node of S_i has a path to an out-node as determined by the corresponding path in its model graph.

Observation: Let P be the path from an in-node to an out-node in a substructure. If the first node on P is white, then the first edge on P is in M. If the last node on P is white, then the last edge on P is not in M.

S5: Each edge of $\tilde{G}$ with each endnode white or shrunk is contained in a substructure.

S6: A substructure with model M1 is a root and incident with no edge of M.

S7: If distinct substructures S_i and S_j share a node, then the node is shrunk, white, or grey in both substructures.

S8: Each grey node in $\tilde{G}$ is contained in precisely two substructures, one has model M13 or M16 and the other has model M21 or M22.

Observation: From the definition of the models (see Figures 6 to 9), the black, grey, and striped nodes are all incident with 2 edges of M. Hence, the non-shrunk roots are a subset of the white nodes of $S_1, ..., S_m$.

S9: Each white root node or shrunk root node of $\tilde{G}$ is an in-node in no substructure. Each white non-root node or shrunk non-root node is an in-node in precisely one substructure.

Observation: A white node or a shrunk node of $\tilde{G}$ can be an out-node in any number of substructures.

Definition: Construct the following digraph $D = (V', A')$. Let V' be the set of white nodes and shrunk nodes in $\tilde{G}$ and start with A' empty. For every non-root node $u \in V'$, let S_i be the unique substructure in which u is an in-node. Let v be the corresponding out-node in S_i obtained by following the arrow path in S_i (obtained from the corresponding model). Add to A' an arc from u to v. Thus each arc in D has a corresponding substructure S_i and path in $\tilde{G}$; we call nodes u and v an *in-node, out-node pair* in S_i.

S10: The digraph D is a directed forest (i.e., its underlying undirected graph is acyclic). (By property S9, every node has at most one out arc, but, based on this alone, the underlying graph could contain a cycle.)

Observation: We show in Section 5.4 that there exist alternating paths in G associated with every directed path in D. They play an important role in the algorithm in its search for augmenting paths. These paths are analogous to the alternating paths defined by the tree structure in Edmonds' matching algorithm [18].

S11: If a substructure has model M17, then D contains a directed path from the node corresponding to e to the node corresponding to b (indicated by the dashed arrow in Figure 8). Similarly, if a substructure has model M20 (see Figure 9), then D contains a directed path from the node corresponding to e to the node corresponding to a.

S12: There are three ways two substructures can be combined by the above rules that are prohibited; see Pr1, Pr2, and Pr3 in Figure 10. Graphs Pr1, Pr2, and Pr3 contain the substructures with models M2, M5, and M6 (on nodes b and u), respectively, combined with a substructure with model M8 or M9. In each case the nodes u, v, b induce a triangle in G.

S13: In a substructure with model M9, the path in D from node b cannot go to a. (This condition is enforced when substructures with model M9 are created in Subroutine 4. This condition plays a role in the proof of Theorem 4 in Section 7 as it prevents the creation of a triangle in M by an exchange along an alternating path. Also, looking ahead to definitions

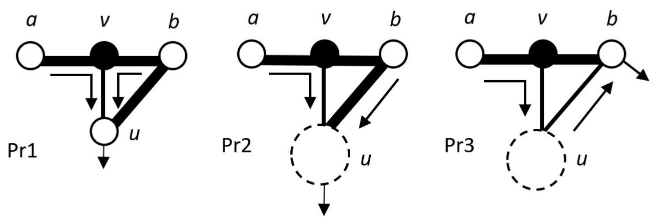


Figure 10: Prohibited substructure combinations

later in this section, if a substructure S_i with model M9 is temporary, then an alternating path passing from d to b through S_i would then traverse the path $\{a, c, d\}$, thus using edge cd twice.)

Note: The next three properties of substructures are implied by the previous properties. Though we do not make explicit use of them, they are stated here for clarity.

- In a substructure with model M11, the path in D from c can go to a or b, OR the path in D from a can go to c, but not both (which would imply a dicycle in D).
- In a substructure with model M14, the path in D from c can go to a or b, OR the path in D from a can go to c, but not both (which would imply a dicycle in D).
- In a substructure with model M19, the path in D from node a can go to d or e, OR the path in D from e can go to a or b, but not both (which would imply a dicycle in D).

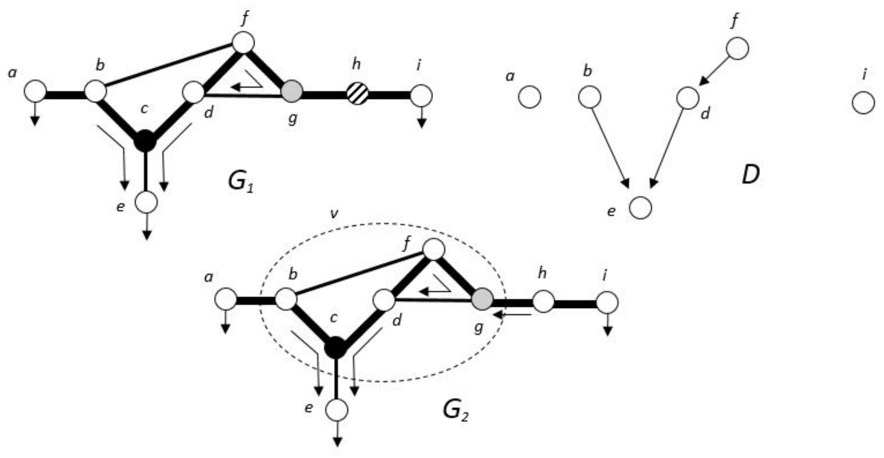
We next state a couple of definitions that make use of the above properties of substructures.

Definition: A substructure is called *standard* or *temporary* in agreement with the type of its corresponding model. Suppose S_i is a substructure whose model is temporary and has two endnodes, say x and y. Then the nodes of S_i that correspond to x and y are called the *endnodes* of S_i.

Definition: For every non-root node v in D, there exists a unique directed path to a root. (This follows from Properties S9 and S10.) We call this root the *root associated with* v.

Note: When referring to nodes in a substructure, we often use the labels of the corresponding nodes in the associated model.

Examples of substructures:



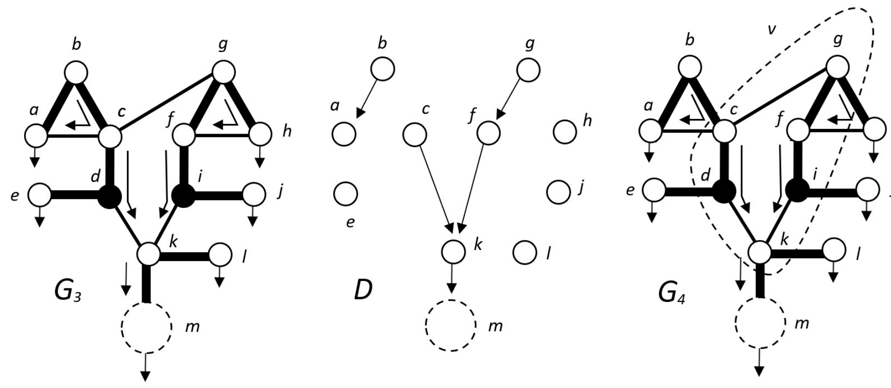
Graph G_1 contains the following substructures:

Nodes {*a,b*} induce a substructure with model M2.
Nodes {*b,c,d,e*} induce a substructure with model M8.
Nodes {*d,f,g*} induce a substructure with model M13.
Nodes {*g,h,i*} induce a substructure with model M22.
Nodes {*b,f*} induce a substructure with model M24.

Graph G_2 (with shrunk node v) contains the following substructures:

Nodes {*a,v*} and {*h,i*} induce substructures with model M2.
Nodes {*v,e*} induce a substructure with model M6.
Nodes {*v,h*} induce a substructure with model M5.

Figure 11: Substructures in a graph G_1, the digraph D for G_1, and a shrinking G_2 on G_1



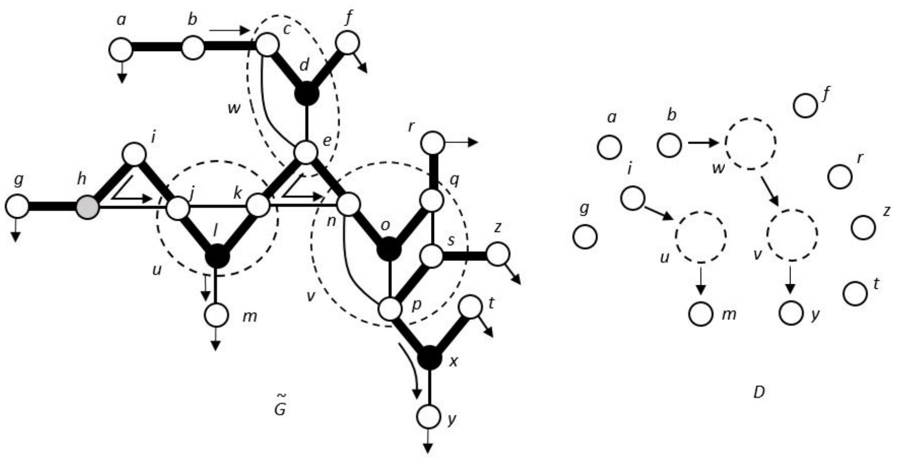
Graph G_3 contains the following substructures:

- Nodes {*k,l*} induce a substructure with model M2.
- Nodes {*k,m*} induce a substructure with model M5.
- Nodes {*c,d,e,k*} and {*f,i,j,k*} induce substructures with model M9.
- Nodes {*a,b,c*} and {*f,g,h*} induce substructures with model M11.
- Nodes {*c,g*} induce a substructure with model M24.

Graph G_4 (with shrunk node v) contains the following substructures:

- Nodes {*e,v*}, {*v,j*}, and {*v,l*} induce substructures with model M2.
- Nodes {*v,h*} induce a substructure with model M3.
- Nodes {*v,m*} induce a substructure with model M5.
- Nodes {*a,b,v*} induce a substructure with model M11.

Figure 12: Substructures in a graph G_3, the digraph D for G_3, and a shrinking G_4 on G_3



An example of $\tilde{G}$ from the algorithm and the associated graph *D*. Nodes *w, u, and v* of $\tilde{G}$ are shrunk with the inside edges and nodes displayed.

Nodes {*w,u,v*} induce a temporary substructure with model M11.
Nodes {*a,b*}, {*f,w*}, {*r,v*}, and {*z,v*} induce substructures with model M2.
Nodes {*b,w*} induces a substructure with model M5.
Nodes {*i,h,u*} induce a substructure with model M13.
Nodes {*g,h*} induce a substructure with model M22.
Nodes {*u,m*} induce a substructure with model M6.
Nodes {*v,t,x,y*} induce a substructure with model M9.

Figure 13: Substructures in a graph $\tilde{G}$ and the digraph D

Consider the graph G_1 in Figure 11. It plays the role of $\tilde{G}$ and contains the substructures described in the figure, which satisfy the above properties.

In this example, all the nodes are non-shrunk, but some of the white nodes could be shrunk as long as the properties for substructures are satisfied. In this case, any subset of the white nodes could be shrunk, as long as at most one of the white nodes in the M13 substructure is shrunk.

The graph D in Figure 11 is based on G_1 and satisfies property S10.

As we see in the algorithm, the node set $\{b, c, d, f, g\}$ can be shrunk. This is indicated in graph G_2 in Figure 11 by the dashed oval, which is the shrunk node v. The new substructures obtained from this shrinking are described in the figure.

This shrinking illustrates a key step in the algorithm. The reason this graph can be shrunk is discussed in the algorithm.

Next, consider the graph G_3 in Figure 12. It plays the role of $\tilde{G}$ and contains the substructures described in the figure.

In this example, all the nodes are non-shrunk, but any subset of the white nodes could be shrunk, as long as the properties for substructures are satisfied.

The graph D in Figure 12 is based on G_3 and satisfies property S10.

As we see in the algorithm, the node set $\{c, d, f, g, i, k\}$ can be shrunk. This is indicated in graph G_4 in Figure 12 by the dashed oval, which is the shrunk node v. The new substructures obtained from this shrinking are described in the figure.

Finally, consider the graph $\tilde{G}$ in Figure 13 with the associated graph D. It is another illustration of how substructures can appear in the algorithm. Noteworthy in this example are the three nodes w, u, v that induce a temporary substructure with model M11.

5.4 Properties of shrunk nodes in $\tilde{G}$

Suppose we are running the algorithm with a graph $\tilde{G}$, a tri-free simple 2-matching M, and a set of substructures that satisfy the above properties (S1-S13). In this section, we make some definitions and state a few additional properties that are maintained during the algorithm. These properties concern the shrunk nodes and types of alternating paths through shrunk nodes. We then show how these alternating paths can be used to build longer alternating paths through the substructures. These longer paths are tracked in the algorithm and used to construct augmenting paths.

We begin by listing two properties of shrunk nodes in the algorithm.

A1: Let v be a shrunk **non-root** node in $\tilde{G}$ during the algorithm. Then every node inside v is incident with two edges of M. (This follows from the definition of root node.)

A2: Let v be a shrunk **root** node in $\tilde{G}$ during the algorithm. Then there exists a unique node inside v that is incident with exactly one edge of M. All other nodes inside v are incident with two edges of M.

Definitions: Let v be a shrunk node in $\tilde{G}$. If v is an in-node in a substructure S_i, then the first edge from v along the path in S_i to an out-node of S_i is called the *base edge* of v and the node inside v that is incident with this base edge is called the *base node* of v. The one exception to this is if S_i has model M7. In this case, we let edges ac and bc serve as the base edges for a and b, respectively. As noted in the properties of substructures, if v is not an in-node in any substructure, then it is a root. In this case, the unique deficient node (see Property A2) inside v is its *base node* and it has no base edge.

Examples: In Figure 11, ce is the base edge and c is the base node for shrunk node v in G_2. In Figure 12, km is the base edge and k is the base node for shrunk node v in G_4. Suppose we delete the edge km and the node m in the three graphs in Figure 12. Then node k is the base node for shrunk node v in G_4 and it has no base edge. In Figure 13, lm is the base edge for shrunk node u, ek is the base edge for shrunk node w, and px is the base edge for shrunk node v.

Our next property provides a bit more detail concerning the occurrence of edges of M in a shrunk node.

A3: Let v be a shrunk node in $\tilde{G}$ during the algorithm and let v' be a non-base node inside v. Then at most one of the two edges of M incident with v' is outside v.

We next state a property that says each shrunk node in the algorithm is one of five types.

A4: Let v be a shrunk node in $\tilde{G}$ during the algorithm. Let v' denote the base node of v and let $v'x$ be the base edge of v, if it has one. Then v is one of the following types.

Type 1: v' is incident with two edges of M, both inside v, and v has a base edge (which is not in M).

Type 2: v' is incident with two edges of M, neither is inside v, and one is the base edge.

Type 3: v' is incident with two edges of M, where one is inside v and the other is the base edge.

Type 4: v' is incident with one edge of M, where this edge is outside v and v has no base edge (hence v is a root).

Type 5: v' is incident with one edge of M, where this edge is inside v and v has no base edge (hence v is a root).

Examples: In Figure 11, the shrunk node v in G_2 is of type 1. In Figure 12, the shrunk node v in G_4 is of type 2. In Figure 13, the shrunk node u is of type 1, the shrunk node w is of type 2, and the shrunk node v is of type 3. If we were to delete the node m and incident edge km in Figure 12, then v would become a shrunk node of type 4 and a root. If we were to delete the node x (and the three incident edges) in Figure 13, then v would become a shrunk node of type 5 and a root.

We next present the notion of *border edges* and follow this with a few examples and a proposition that describes their key properties. This explains the role of the dashed thick edges in Figures 6 to 9.

Definition: Let S_i be a substructure in $\tilde{G}$. An edge $x'y'$ of $\tilde{G}$ is called a *border edge* of S_i if $x'y' \in M$, x' is in S_i, y' is not in S_i, $x'y'$ is adjacent in G to an edge in S_i, and x' is not an in-node of the substructure that contains $x'y'$.

Examples of border edges: In Figure 11, consider the substructure in G_1 with model M8 that is induced by nodes b, c, d, e. The edges ab and df are border edges for this substructure, but there is no border edge incident with node e. Also, edge gh is a border edge for the substructure induced by nodes d, f, g. Next, consider G_3 in Figure 12. The edge kl is a border edge for the substructure induced by nodes k, m and for the substructures induced by c, d, e, k and f, i, j, k. However, edge km is not a border edge of substructures c, d, e, k and f, i, j, k since k is an in-node in substructure k, m. Note also that in G_4 in Figure 12, edge ed is not a border edge of substructure k, m since it is not adjacent to edge km in G.

The following proposition is proved in Section 7.

Proposition 2. *Let S_i be a substructure in $\tilde{G}$ with model F. Let x' be a node of S_i and let x be the corresponding node in F. (Refer to Figures 6 to 9.)*

1. *If there is no dashed thick edge incident with x in the figure for F, then S_i has no border edge at x'.*

   For the next two points, assume there is a dashed thick edge incident with x in the figure for F.

2. *If F is model M13 or M16, then $x = c$ (see Figures 7 and 8) and S_i must have a border edge at x'.*

3. *In all other cases, it is possible that S_i has a border edge at x'.*

We next introduce the notions of *pendant edges* and *cross edges.* As we see in the algorithm, border edges can become pendant edges when a shrinking occurs. (And later, when proving the algorithm works, we see that pendant edges become edge-petals or contained in tri-petals of some tri-blossom clusters.)

Definitions: Let v be a shrunk node in $\tilde{G}$ during the algorithm. Let $G(v)$ denote the subgraph of G induced by (1) the nodes and edges inside v; (2) the two edges (and their endnodes) in each substructure with model M3 where v plays the role of a (see Figure 6); (3) the base edge, if v has type 1; and (4) all other edges in M with one endnode inside v and one outside v, with the exception of the base edge when it is in M. (Note that the base edge is in M if and only if v has type 2 or 3.) Given $G(v)$, let us call an edge that satisfies (3) or (4) a *pendant edge* of $G(v)$. Let uv' be a pendant edge with v' inside v such that u has degree 1 in $G(v)$. An edge uv'' of G is called a *cross edge* for $G(v)$ if it satisfies the following: it is not in M; the node v'' is inside v; and there is an edge $v'v''$ of G that was contained in a substructure when it first shrunk. Hence, the nodes u, v', v'' form a triangle in G.

Examples of $G(v)$, pendant edges, and cross edges: Consider the graph G_2 in Figure 11 with shrunk node v. Then $G(v)$ contains the graph inside v plus the pendant edges ab, hg, and ec. If there were an edge hf (clearly not in M), it would be a substructure with model M24 and it would be a cross edge for $G(v)$. Similarly, if there were an edge ed or ac. Consider the graph G_4 in Figure 12 with shrunk node v. Then $G(v)$ contains the graph inside v plus the edges hg and hf (which are in a substructure with model M3), and the pendant edges bc, ed, ji, and lk. Note that mk is not a pendant edge since it is a base edge for this type 2 shrunk node.

We next state a property that says each shrunk node in the algorithm must have certain types of alternating paths.

A5: Let v be a shrunk node in $\tilde{G}$ during the algorithm. If v has type 1, let x be the endnode of the base edge that is not inside v; otherwise, let x be the base node. The algorithm maintains the following types of alternating paths for all five types of shrunk node v.

Type 1: For each node u inside v, there exists an alternating path from u to x in $G(v)$ that starts with an edge in M and ends with an edge not in M. In the case that v has type $2, \ldots, 5$ and u is the base node of v, then the alternating path is the trivial one consisting only of u.

Type 2: Let $uv' \in M$ be a pendant edge of $G(v)$, where v' is inside v. Then there exists an alternating path from u to x in $G(v)$ that starts with uv' and ends with an edge not in M.

Type 3: Let uv' be a pendant edge of $G(v)$, where v' is inside v, with a cross edge uv''. Then there exists an alternating path from u to x in $G(v) \cup uv''$ that starts with uv'' and ends with an edge not in M. Performing an exchange on this path leaves the triangle u, v', v'' with exactly two edges in M.

Note: As defined above, we use the terms *type 1, type 2 , . . .* to refer to varieties of alternating paths and shrunk nodes. It should be clear from the context which situation we are referring to.

Examples of alternating paths: Consider shrunk node v in G_2 in Figure 11: $\{b, c, e\}$ and $\{g, f, b, c, e\}$ are alternating paths of type 1; $\{h, g, d, f, b, c, e\}$

is an alternating path of type 2; and, if there were a cross edge ac, then $\{a, c, b, f, g, d, c, e\}$ would be an alternating path of type 3. (Note that triangle $\{a, b, c\}$ ends up with two edges in M after an exchange on this path.) Similarly, in G_4 in Figure 12: $\{c, d, k\}$ and $\{g, h, f, i, k\}$ are alternating paths of type 1 for v. Note that the alternating path of type 1 from g contains the node h, which is outside v, but is contained in $G(v)$.

Let us point out that such alternating paths can, in some cases, include a cycle in G.

Examples of alternating paths with cycles: For shrunk node v in G_2 in Figure 11, we have that $\{c, d, g, f, b, c, e\}$ is an alternating path of type 1; and, if there were a cross edge ed, then $\{e, d, c, e\}$ would be an alternating path of type 3. For shrunk node v in G_4 in Figure 12, we have that $\{l, k, i, f, h, g, c, d, k\}$ is an alternating path of type 2.

Note that, in A5, a type 3 alternating path is not the same as a type 1 alternating path plus the edge uv''. Consider $\tilde{G}$ in Figure 13: If there were a cross edge zp, then its alternating path of type 3 would be $\{z, p, s, q, o, p\}$. Note that this path is not the same as edge zp plus the alternating path of type 1 from p, which is simply the node p. If we were to perform an exchange along the path $\{z, p\}$, (which would typically be continued to a root along the path $\{p, x, y\}$) we would create a triangle in the updated M.

Definitions of longer alternating paths:

We next make use of the type 1, 2, and 3 alternating paths for shrunk nodes (see Property A5) to define longer alternating paths through the substructures and present some associated notation. Included are some special alternating paths for the temporary substructures. These paths play an important role in the algorithm, which searches for exchanges on alternating paths that increase the size of the current tri-free simple 2-matching.

Let $\bar{P}_v^u$ be a di-path in D from node u to node v. (For example, (with different node labels) see the path $\{f, d, e\}$ in D in Figure 11, the path $\{g, f, k\}$ in D in Figure 12, and the path $\{w, v, y\}$ in D in Figure 13.) Let u_b denote the base node for u if it is shrunk, and let it denote u if it is white. Define v_b similarly for v. For each arc xy in $\bar{P}_v^u$ consider the edges of $\tilde{G}$ in the corresponding path through the substructure that corresponds to xy. Let $\tilde{P}_v^u$ be the path in $\tilde{G}$ defined by these sets of edges. For each shrunk node w of $\tilde{P}_v^u$, different from u, do the following: Let z_1 be the endnode inside w of the edge of $\tilde{P}_v^u$ incident with w and closer to u and let z_2 be the endnode inside w of the edge of $\tilde{P}_v^u$ incident with w and closer to v, if such an edge exists; otherwise, we have $w = v$ and we let $z_2 = v_b$. If $z_1 \neq z_2$ then extend the path $\tilde{P}_v^u$ through the shrunk node w from z_1 to z_2 using the appropriate interior path as implied by the properties defined above (where the final edge is deleted if w has type 1); that is, use a type 1 or 2 path: if the edge of $\tilde{P}_v^u$ incident with w and closer to u is not in M use the type

1 path, otherwise use the type 2 path. Note that if $z_1 = z_2$ and the two edges of $\tilde{P}_v^u$ incident with w are both in M or both not in M, then we are extending $\tilde{P}_v^u$ around a cycle inside w (as we discussed above). Let P_v^u denote the extended path in G just defined. Note that P_v^u is a path in G from u_b to v_b. Furthermore, by the observation after Property S4 and the properties of alternating paths through shrunk nodes, P_v^u is alternating. Given $\bar{P}_v^u$ in D, if v is a root, then we abbreviate P_v^u with P^u. Similarly, we refer to $\bar{P}^u$. Finally, let us extend this notation as follows: If u is shrunk and a is a node inside u, we let P^a denote the type 1 path from a to the base of u plus the path P^u.

Additional examples of alternating paths: Consider the graphs in Figure 13. We have, for example, $\bar{P}_v^b$ is the di-path $\{b, w, v\}$ in D. We have $\tilde{P}_v^b$ is sequence of edges bw, wu, uv in $\tilde{G}$. And we have P_v^b is the path $\{b, c, e, k, n, o, p\}$ in G. Similarly, we have P^j is the path $\{j, l, m\}$ and P^o is the path $\{o, n, p, x, y\}$.

We also define alternating paths for temporary substructures as follows. Each of the cases below constructs an alternating path based on the two endnodes u and v of the substructure. We denote this path P^{uv}. (An example is given for temporary substructures with model M11.)

1. Consider a temporary substructure with model M2: Let a and b play the roles of u and v, respectively. Let a_b and b_b be the base nodes of nodes a and b, respectively. Consider the type 2 alternating path starting with ab and then passing through a (where the final edge is deleted if a has type 1). Similarly, consider the type 2 alternating path from ab through b. Combine these to yield an alternating path from a_b to b_b.

2. Consider a temporary substructure with model M3. Let a and b play the roles of u and v, respectively. Let a' be the endnode inside a of the edge of the substructure not in M and let a_b be the base node of a. If b is shrunk, let b' be the endnode inside b of the two edges ab and let b_b be its base node. Otherwise, let $b = b' = b_b$. When node a is shrunk in the algorithm (from a substructure with model M11 or M14), we define a special alternating path for this situation. The path starts with $b'a'$ and can be taken to a_b. If b is shrunk, add to this alternating path the type 1 alternating path from b' to b_b inside $G(b)$. This yields an alternating path from a_b to b_b.

3. Consider a temporary substructure with model M9: Let b and d play the roles of u and v, respectively. Let b_b be the base node of b. If d is shrunk, let d_b be the base node for d, and let d' be the endnode of cd inside d. Otherwise, let $d_b = d$. Consider the type 2 alternating path starting with cb and continuing inside $G(b)$ to b_b (where the final edge is deleted if b has type 1). Add to this the edge cd and, if d is shrunk, the type 1 alternating path inside $G(d)$ from d' to d_b (where the final edge is deleted if d has type 1) to obtain an alternating path from d_b to b_b.

4. Consider a temporary substructure with model M11: Let a and c play the roles of u and v, respectively. Let a_b, b_b, and c_b denote the base nodes

of a, b, and c, respectively. Consider the type 2 alternating path from ab through b to b_b (which contains a cycle inside $G(b)$); the type 2 alternating path from bc through $G(c)$ to c_b; and the type 2 alternating path from ab through $G(a)$ to a_b (where the final edge on the path through $G(a)$ is deleted if a has type 1; similarly c). Combine these to yield an alternating path from a_b to c_b.

Example: Consider the temporary substructure u, v, w in $\tilde{G}$ in Figure 13. It has endnodes that are already labeled u and v. Then P^{uv} is the path $\{l, j, k, e, d, c, e, n, p\}$.

5. Consider a temporary substructure with model M14: Let a and c play the roles of u and v, respectively. Let a_b and c_b denote the base nodes of nodes a and c. Consider the type 2 alternating paths from ac through $G(a)$ and through $G(c)$ (where the final edge on the path through $G(a)$ is deleted if a has type 1; similarly for c). Combine these alternating paths to obtain an alternating path from a_b to c_b.

6. Consider the temporary substructures with models M17 and M20. These situations are handled in Step 2, part 2 of the Main Algorithm. (In both cases we treat edge ce as a substructure with model M24.)

7. Consider a temporary substructure with model M24. Let a and b play the roles of u and v, respectively. If node a is shrunk, let a_b denote the base node of a and let a' denote the endnode of edge ab inside a. If a is white (not shrunk), set $a_b := a$ and $a' := a$. Define b_b and b' analogously for b. Consider the type 1 alternating paths from a' to a_b and from b' to b_b (where, if a is shrunk and has type 1, the final edge on the path through $G(a)$ is deleted; similarly for b). Combine these with $a'b'$ to yield an alternating path from a_b to b_b. An exception to this is if a and b play the roles of u and v, respectively, in the definition of a type 3 alternating path. That is, using the notation in that definition, suppose we have that $b = v$ is shrunk, and there is a node $b'' = v'$ inside b such that $b''a = v'u$ is a pendant edge for $G(b) = G(v)$, and $ab' = uv''$ is a cross edge for $G(b) = G(v)$; hence the nodes a, b'', b' (equivalently, nodes u, v', v'') form a triangle in G. If ab' is not also a cross edge from $G(a)$, then combine the type 3 alternating path from ab' through $G(b)$ to b_b with the type 1 alternating path through $G(a)$ from a' to a_b to obtain an alternating path from a_b to b_b. Proceed analogously if ab' is a cross edge for $G(a)$ and not for $G(b)$. If ab' is a cross edge for both $G(a)$ and $G(b)$, use the two type 3 alternating paths.

6 The algorithm

In this section we present an algorithm for finding a maximum cardinality tri-free simple 2-matching in a graph. The algorithm makes use of the structures defined in the previous section. A proof of the algorithm's validity is given in Section 7.

The Main Algorithm is given in Section 6.1. Various subroutines used in the Main Algorithm are given in subsequent sections.

During the algorithm we have a simple graph G, a tri-free simple 2-matching M in G, a graph $\tilde{G}$ obtained from G by shrinking nodes (that satisfy properties A1-A4), a set of substructures of $\tilde{G}$, say $S_1, \ldots, S_m$, that satisfy the properties of substructures (S1-S13), and alternating paths through the shrunk nodes (that satisfy property A5).

In general, the features of our algorithm mirror the features of Edmonds' algorithm [18] for matchings. In Edmonds' algorithm, a current matching is maintained along with an alternating forest that has white, shrunk, and black nodes. In a sequence of iterations, the algorithm seeks to perform shrinkings, grow the forest, or perform an exchange on an alternating path that increases the cardinality of the current matching. In our algorithm, we maintain a current tri-free simple 2-matching. We also maintain a collection of substructures together with the directed tree D, which serve as an analog of Edmonds' alternating forest. We perform shrinkings (in Subroutine 1). We grow (or modify) the substructures (in Subroutines 2, 3, and 4). And we perform exchanges on alternating paths (in the Main Algorithm) to increase the cardinality of our current tri-free simple 2-matching. However, all of these procedures are more complex for our problem. For example, in Edmonds' algorithm, when an edge is considered that connects two white nodes and creates a cycle with the edges in the alternating forest, a shrinking is performed. In our algorithm, the temporary substructures play the role of this edge. However, when we consider a temporary substructure and discover a "cycle" formed with the arcs in D, we may perform a shrinking or we may modify the substructures.

Note: In the algorithm we use the notation for alternating paths defined on page 33.

6.1 The Main Algorithm

Main Algorithm: Maximum cardinality tri-free simple 2-matching

Input: A simple graph G and a tri-free simple 2-matching M in G.
Output: A maximum cardinality tri-free simple 2-matching in G.

Step 0: If $|M| = |V|$, then output M; end. Otherwise, set the substructures as follows: Let each node incident with no edge in M be a substructure with model M1. Let each maximal path in M be a substructure with model M2 or M23. Let each cycle in M be a substructure with model M4. Set $\tilde{G} := G$. For each deficient node v, set $P^v := v$.

Step 1 Consider the edges of $\tilde{G}$ that (1) are not in M; (2) are not in a substructure; and (3) have each endnode either white or shrunk. Make each such edge into a substructure with model M24.

Step 2: Perform this step if there exists a temporary substructure; otherwise, go to Step 3. (The models for temporary substructures and their endnodes are defined in Figures 6 to 9.) Select one temporary substructure, call it S_t, as follows: If possible, choose S_t so that it does not have model M24. (This is needed for cases B1 and B2 in Subroutine 2.) Otherwise, choose an S_t with model M24. Let u and v denote the endnodes of S_t.

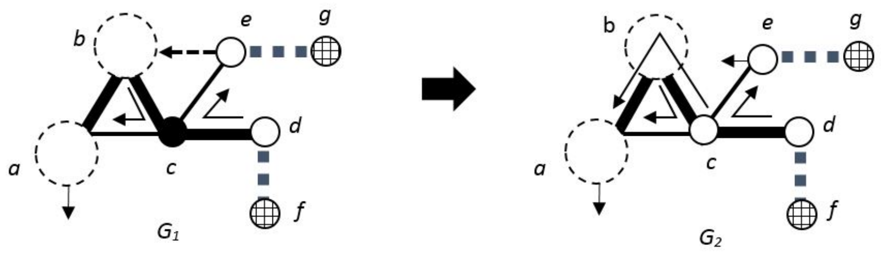
1. If S_t has model M24, run Subroutine 2 (on page 65). Upon return, if a transformation was performed in the subroutine, go to Step 1. If a transformation was not performed, skip below to part 3 of Step 2 of the Main Algorithm.

2. If S_t has model M17 or M20 we perform a transformation of S_t as depicted in Figures 14 and 15, respectively. To begin, convert the triangle abc to a substructure with model M12 or M15, respectively; hence, in both cases, we treat c as a white node (see graphs G_2 in the figures). In the first case, let a' be the endnode of edge ab inside a and let b' be the endnode of ab inside b. Let P^* denote the type 2 alternating path through shrunk node b that starts with ab and ends at node b' (which includes a cycle inside b). Set $P^c := cb \cup P^* \cup P^{a'}$. In the second case, again let a' be the endnode of edge ab inside a. Set $P^c := ca \cup P^{a'}$. In both cases, treat ce as a temporary substructure with model M24 with endnodes c and e. Reset S_t to be this new temporary substructure with endnodes $u := c$ and $v := e$. (We allow this at this point in the algorithm, even if there remain other temporary substructures that do not have model M24.) Since, in both cases, $u = c$ and $v = e$ do not satisfy the conditions for performing an exchange in Step 2, part 3, we will be performing a shrinking in Step 2, part 4. Thus we anticipate this by treating dc as a substructure with model M5 in the new graphs G_2 (where the arrow from d shows its alternating path after the shrinking). (This temporarily violates our properties of substructures, but allows us to retain the subtree of D "above" node d along with all the substructures that contain these nodes. These substructures are unaffected by the shrinking.)

3. If endnodes u and v have different roots associated with them; or u and v have the same root, the root is white (not shrunk) and incident with no edges in M (i.e., the root is a substructure with model M1), and P^u and P^v share only the root node: Let $P = P^u \cup P^{uv} \cup P^v$.

   (*) Consider each substructure S_j with model M14, M15, M16, or M20 that is in $\tilde{G}$ where P contains an edge adjacent in G to edge ab and P contains no edge of the triangle a, b, c. (In all these cases, observe that no edge inside b is contained in P). Perform the following operation on P for each S_j:

   Extension of P: Let P' be the alternating path $ac \cup cb \cup$ {the type 1 alternating path from the base node of b} (which extends around a



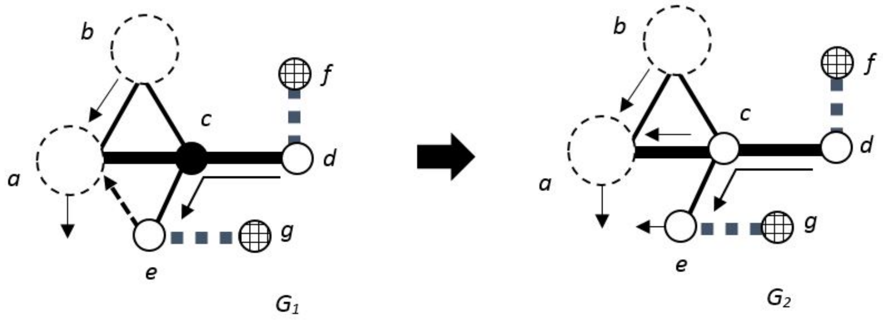
In G_2:

Nodes $\{a,b,c\}$ induce a substructure with model M12.

Nodes $\{c,e\}$ induce a substructure with model M24.

Nodes {d,c} induce a substructure with model M5 (since node *c* will shrink).

Figure 14: Transformation of temporary substructure with model M17



In G_2:

Nodes $\{a,b,c\}$ induce a substructure with model M15.

Nodes $\{c,e\}$ induce a substructure with model M24.

Nodes {d,c} induce a substructure with model M5 (since node *c* will shrink).

Figure 15: Transformation of temporary substructure with model M20

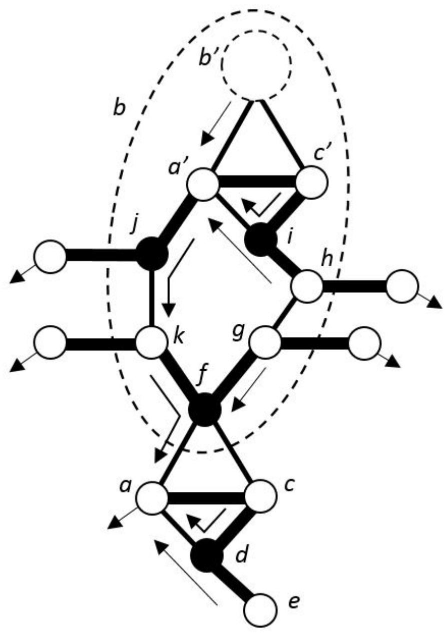


Figure 16: Performing an extension of P in the algorithm

cycle inside b and ends with edge ba). Reset P to be the alternating path obtained by combining P and P'.

For each shrunk node x in $\tilde{G}$, let $N(x)$ denote the nodes that shrunk into x when x was shrunk (during an execution of part 4 of this step). Consider the substructures with nodes in $N(x)$ just before x was shrunk. Consider each such substructure S_k that is in $\tilde{G}$ where P contains and edge adjacent in G to edge ab and P contains no edge of the triangle a, b, c. As above, perform an extension of P for each such S_k. Similarly, apply the extension of P procedure recursively for shrunk nodes in $N(x)$.

If an extension of P was performed since the last pass through line (*), then return to line (*).

Perform an exchange on the alternating path P to obtain an updated M. (We augment the size of M.) Throw away the substructures, unshrink the nodes, and go to Step 0 with the updated M.

Observe that, for each S_j and S_k used for an extension of P, each triangle a, b, c ends up having two edges in the updated M. Thus none of the edges in this triangle can occur in a triangle of the updated M. This is used the proof of the algorithm's validity. Consider the following examples.

Example: Consider the graph $\tilde{G}$ in Figure 13. Suppose we have chosen the substructure u, v, w with model M11 as S_t. Then u and v have different roots (m and y, respectively). We perform an exchange on the alternating path $\{m, l, j, k, e, c, d, e, n, p, x, y\}$.

Example: Consider the graph in Figure 16. The large dashed oval is a shrunk node b in $\tilde{G}$ with the graph inside b shown as it was before b shrunk. Shrunk node b is in a substructure with nodes a, b, c and model M14. Each black node is contained in a substructure with model M8 or M9, as indicated by the arrows, both inside and outside the shrunk node b. (Note that b shrunk when we considered edge gh, which was a substructure with model M24. After b shrunk, edge fc became a substructure with model M24 in Step 1 of the Main Algorithm. The substructure with nodes a, b, c was then formed in a call to Subroutine 2, case B2, where edge ac was a substructure with model M2.) The triangle a', b', c' is another substructure with model M14, contained inside b. The four edges of M with one endnode inside b are substructures with model M2 that contain the shrunk node b. Suppose we are just entering part 3 of Step 2 in the Main Algorithm and that the condition of part 3 is satisfied. Suppose this graph depicts part of the substructures of $\tilde{G}$ at this time. Suppose the exchange contains the subpath $\{e, d, a\}$. If we performed an exchange on P at this time, the updated M would contain the triangle a, c, d. Hence, this step dictates that we next perform an extension of P on the substructure a, b, c. The path P becomes the alternating path: $\{a, c, f, g, h, i, a', j, k, f, a\}$. (The portion of this path inside b was determined when b was first shrunk (due to edge gh).) Note that performing an exchange on this new P would eliminate the triangle a, c, d from the updated M, but it would create a new triangle a', c', i inside b. This step then directs us to perform another extension of P on the substructure a', b', c'. Now, performing an exchange on the new P eliminates the triangle a', c', i from the updated M.

4. Otherwise, run Subroutine 1. (We identify a new shrunk node.) Upon return, go to Step 1.

Step 3: Search for an edge uv in $\tilde{G}$ that satisfies the following: uv is not in M and not in a substructure; u is white or shrunk and v is grey or striped; and, if v is grey, in a substructure with model M22, then that substructure contains two or more edges. If such an edge does not exist, go to Step 4. Otherwise, select one such edge uv, and perform 1 or 2 below.

1. If v is grey, run Subroutine 3. (We add/alter some substructures.) Upon return, go to Step 1.

2. If v is striped, run Subroutine 4. (We add/alter some substructures.) Upon return, go to Step 1.

Step 4: Search for a substructure with model M5 or M6, where nodes a and b are shrunk. If there is no such substructure, go to Step 5. Otherwise, select one such substructure and add a new edge $a'b' \notin M$ to G, where a' is an arbitrary node inside a and b' is an arbitrary node inside b. Hence, this new edge has endnodes a and b in $\tilde{G}$. Call this edge a substructure with model M24. Run Subroutine 1 with this new edge playing the role of uv. (The subroutine shrinks a and b together, with possibly some other nodes of $\tilde{G}$.) (Note that edge $a'b'$ is shrunk into the new shrunk node that contains a and b.) Upon return, remove the edge $a'b'$ from G. (It is contained in no alternating path through the new shrunk node.) Go to Step 1.

Step 5: Output M. End.

6.2 Subroutine 1: Shrinking

Let us begin by describing the notation we use in Subroutine 1.

For each substructure S_i, let $V(S_i)$ denote the nodes of $\tilde{G}$ in S_i. Given input nodes u and v (the endnodes of our chosen temporary substructure S_t), add an edge uv to the undirected graph underlying D and let C denote the set of nodes of $\tilde{G}$ in the resulting cycle. Then delete uv from D; the only purpose of adding this edge to D is to identify C. (Recall that S_t need not be a substructure with model M24.) Let $C(S_i) := C \cap V(S_i)$ for all S_i.

Example: For the graph G_1 in Figure 11, let S_t be the substructure bf with model M24. Hence nodes b and f play the roles of endnodes u and v in S_t. Observe, based on adding an edge $uv = bf$ to the graph underlying D, that $C = \{b, e, d, f\}$. If S_i refers to the substructure of G_1 with nodes $\{d, f, g\}$, then $V(S_i) = \{d, f, g\}$ and $C(S_i) = \{d, f\}$. For the graph G_3 in Figure 12, let S_t be the substructure cg with model M24. Hence nodes c and g play the roles of endnodes u and v in S_t. Then, from D, we see that $C = \{c, k, f, g\}$. (Typically, the nodes in C, plus potentially some additional nodes, are shrunk in the algorithm. We have a special situation in G_2 in Figure 11. In this case we produce a type 1 shrunk node and the node e is not shrunk. This is handled in Subroutine 1*.)

For each substructure S_i, the nodes in $V(S_i) - C(S_i)$ are partitioned into $Ext(S_i)$ (the *externally selected nodes* of S_i), $Int(S_i)$ (the *internally selected nodes* of S_i), and any remaining nodes. In addition, the nodes in $C(S_i) \cup Ext(S_i)$ are either *processed* or *unprocessed.* These sets are denoted $Proc(S_i)$ and $Unproc(S_i)$, respectively.

During a run of Subroutine 1, each substructure S_i is considered iteratively. Subroutine 1, using a call to Subroutine 1*, determines the nodes to add to $Ext(S_i)$ and $Int(S_i)$. These are the nodes, together with $C(S_i)$, over all the substructures, that are shrunk inside a new shrunk node at the end of the call of Subroutine 1. (When proving the validity of the algorithm, we show that the sets $Ext(S_i)$ and $Int(S_i)$ only grow as the S_i are repeatedly processed.) Let us call the final shrunk node v. As each S_i is processed, we identify new alternating paths of type 1 and a set of edges that could play the role of pendant edges for the final $G(v)$. (We say "could play" since the sets $Ext(S_i)$ and $Int(S_i)$, which will be shrunk, are not finalized until the end of the subroutine.) A pendant edge at the end of the subroutine has one endnode in $C(S_i) \cup Int(S_i) \cup Ext(S_i)$ and the other not in that set. The border edges for S_i can serve as candidates for such edges, although there can be other candidates in S_i itself when $C(S_i) \cup Int(S_i) \cup Ext(S_i)$ does not include all the nodes of S_i. These potential pendant edges are used to identify the alternating paths of type 2. We also use the potential pendant edges to identify the associated potential cross edges and the associated alternating paths of type 3. When identifying the potential pendant edges and potential cross edges in Subroutine 1^*, we drop the adjective "potential" for expediency.

Note: Consider a substructure S_i during an execution of Subroutine 1^*, where xy is a (potential) pendant edge with x in $C(S_i) \cup Int(S_i) \cup Ext(S_i)$. We sometimes say that the type 2 alternating path for xy is "determined elsewhere" in the subroutine. Immediately after the statement of Subroutine 1^* there is note that explains these references in more detail. The phrase appears in bold in the subroutine to aid the reader.

Subroutine 1: Shrinking

Step 0: For each substructure S_i, initialize: $Unproc(S_i) := C(S_i)$ and $Proc(S_i)$, $Int(S_i)$, $Ext(S_i) := \emptyset$.

Step 1: If this is the first pass through this step in the current call to this subroutine and there exists a substructure S_i with model M8 such that nodes a, b, and d are in $C(S_i)$ (see Figure 7), then choose S_i. (We will be identifying a type 1 shrunk node with base node d. Figure 11 contains an example of this, where the substructure with model M8 on nodes b, c, d, e in G_1 is shrunk (with some other nodes). This forms the substructure ve with model M6 in G_2, where v is a shrunk node with type 1.) Otherwise, select a substructure S_i that contains a node in $Unproc(S_i)$.

1. Run Subroutine 1^* to determine $Int(S_i)$, new alternating paths, and how to shrink S_i, depending on the situation. If more than one alternating path is determined, for a node or edge, retain only the first one identified. (This is an arbitrary choice, since all identified alternating paths are valid.) After returning from Subroutine 1^*, add the nodes in $[C(S_i) \cup Ext(S_i)] \cap Unproc(S_i)$ to $Proc(S_i)$ and remove them from $Unproc(S_i)$.

2. If, in Subroutine 1^*, a node was identified for $Int(S_i)$ that was not previously in $Int(S_i)$, do the following: For each such node x and all S_j, where $j \neq i$ and $x \in V(S_j)$, add x to $Ext(S_j)$ and to $Unproc(S_j)$.

3. If there exists a substructure S_j with $Unproc(S_j) \neq \emptyset$, repeat Step 1 (of Subroutine 1). Otherwise, go to Step 2 (of Subroutine 1).

Step 2: For each substructure S_i, shrink $C(S_i) \cup Int(S_i) \cup Ext(S_i)$, retain the alternating paths, and update the form of S_i as determined in the last consideration of S_i in Subroutine 1*.

End.

As noted above, Subroutine 1* identifies alternating paths of types 1, 2, and 3 for nodes we are shrinking. In particular, when we add a node to $Int(S_i)$, we identify a type 1 alternating path for that node. When we identify a pendant edge, we identify a corresponding type 2 alternating path that starts with that edge. And when we identify a cross edge for S_i (which forms a triangle with a pendant edge and one edge of S_i), we identify a type 3 alternating path that starts with the cross edge. We actually identify alternating paths to a root (with a few exceptions, explained in the subroutine), since a shrunk node may later shrink inside another shrunk node. Truncating these paths at the base edge for type 1 shrunk nodes or, otherwise, at the base node, yields the paths of types 1, 2, and 3. For each case, we also specify how S_i is transformed when the shrinking is performed in Step 2 of Subroutine 1. S_i may entirely shrink, it may retain its current type, or it may switch to one or more different types of substructures.

Note: In Subroutine 1*, when we reference P^u, P^v, or some variant of these (such as P^u_a), we assume u and v have been labeled so that these paths are well defined, with respect to the labeling of the nodes in the corresponding models.

Examples: For the graph G_3 in Figure 12, suppose we have selected, in Step 2 of the Main Algorithm, the substructure cg with model M24 as the temporary substructure S_t and we let u and v denote the endnodes, c and g, of S_t. We next execute Step 2, part 4 of the Main Algorithm with a call to Subroutine 1. In the subroutine, suppose we first select the substructure with model M9 on nodes c, d, e, k as S_i. Thus we have $C = \{c, k, f, g\}$; $C(S_i) = \{c, k\}$; and we have $Unproc(S_i) = C(S_i) = \{c, k\}$ and $Proc(S_i)$, $Int(S_i)$, $Ext(S_i) := \emptyset$. We next call Subroutine 1*. (Note that the node labels for each substructure type in Subroutine 1* refer to the labels on the corresponding models. Hence, in this example, we consider the case where nodes a and d are the only nodes in $C(S_i) \cup Ext(S_i)$.) Using the node labels for G_3 in Figure 12, we add d to $Int(S_i)$ and set P^d to be the alternating path $\{d, c, g, h, f, i, P^k\}$. We also determine alternating paths for (potential) pendant edges lk and ed and for (potential) cross edges determined by these pendant edges. For example, the path for lk is $\{l, k, d, c, g, h, f, i, k, P^k\}$ and the path for cross edge ec would be $\{e, c, d, k, P^k\}$. We also note that, if this were our last consideration of this substructure in Subroutine 1, then, in this substructure, we shrink nodes c, d, k and the remaining edge ed becomes a substructure with model M2.

For another example, consider the graph G_1 in Figure 11. Suppose we have selected, in Step 2 of the Main Algorithm, the substructure bf with model M24 as S_t and we let u and v denote the endnodes, b and f, of S_t. Again, we next execute Step 2, part 4 of the Main Algorithm with a call to Subroutine 1.

Observe that $C = \{b, e, d, f\}$. Note that in the first pass through Subroutine 1 we satisfy the special condition in Step 1, so we first choose the substructure $\{b, c, d, e\}$ with model M8 as S_i. Using the node labels for G_1 in Figure 11, we add the node c to $Int(S_i)$ and identify its alternating path $\{c, d, g, f, b, c, e\}$. We note that alternating paths for (potential) pendant edges ab and fd are determined elsewhere in the subroutine. For (potential) cross edge ac, we would identify the alternating path $\{a, c, b, f, g, d, c, e\}$. Similarly for (potential) cross edge fc. We note that ce becomes the base edge for a type 1 shrunk node, when we perform the shrinking in Subroutine 1. We may next choose the substructure $\{d, f, g\}$ with model M13 as S_i in Subroutine 1*. We add the node g to $Int(S_i)$ and identify its alternating path $\{g, f, b, c, e\}$. We identify (potential) pendant edge hg and identify its alternating path $\{h, g, d, f, b, c, e\}$. For (potential) cross edge hf, we would identify its alternating path $\{h, f, g, d, c, e\}$; for (potential) cross edge hd, we would identify its alternating path $\{h, d, c, e\}$; and so on.

Subroutine 1*: Shrinking details

Node labels refer to Figures 6 to 9.

Note: No substructure S_i is shrunk or transformed during an execution of this subroutine. Shrinkings and transformations are performed when the algorithm returns to Subroutine 1 and Step 2 is performed.

Note: We frequently consider a (potential) pendant edge xy, where $x \in C(S_i) \cup Ext(S_i)$. If x is shrunk, then (by Property A5) the type 2 alternating path from xy has already been determined as well as all type 3 alternating paths for cross edges $x'y$ with x' inside x.

S_i **has model M1:** Cannot happen.

S_i **has model M2 or M3:**

- Suppose a (or b) is the only node in $C(S_i) \cup Ext(S_i)$.
    - S_i retains its type.
- Suppose a and b are in $C(S_i) \cup Ext(S_i)$.
    - S_i shrinks.

S_i **has model M4:** Cannot happen.

S_i **has model M5:**

- Suppose a (or b) is the only node in $C(S_i) \cup Ext(S_i)$.
    - S_i retains its type.

- Suppose a and b are in $C(S_i) \cup Ext(S_i)$.
  - ca is a pendant edge.
    - Path for pendant edge ca is **determined elsewhere**.
    - For cross edge cb, use alternating path $cb \cup ba \cup P_a^u \cup P^{uv} \cup P^v$.
  - S_i shrinks.

S_i **has model M6:**

- Suppose a (or b) is the only node in $C(S_i) \cup Ext(S_i)$.
  - S_i retains its type.
- Suppose a and b are in $C(S_i) \cup Ext(S_i)$.
  - cb is a pendant edge.
    - For pendant edge cb, if b is white, use alternating path $cb \cup P_b^u \cup P^{uv} \cup P^v$.
    - For cross edge ca, use alternating path $ca \cup P_a^u \cup P^{uv} \cup P^v$.
  - S_i shrinks.

S_i **has model M7:**

- Suppose a (or b) is the only node in $C(S_i) \cup Ext(S_i)$.
  - S_i retains its type.
- Suppose c is the only node in $C(S_i) \cup Ext(S_i)$.
  - Add a and b to $Int(S_i)$.
  - dc is a pendant edge.
    - Alternating path for pendant edge dc is **determined elsewhere**.
    - For cross edge db use alternating path $db \cup P^b$.
    - For cross edge da use alternating path $da \cup P^a$.
  - S_i shrinks.
- Suppose a and c are the only nodes in $C(S_i) \cup Ext(S_i)$. (Similarly for b and c.)
  - Add b to $Int(S_i)$.
  - dc is a pendant edge.
    - For pendant edge dc use alternating path $dc \cup P_c^u \cup P^{uv} \cup P^v$.
    - For cross edge db use alternating path $db \cup P^b$.
    - For cross edge da use alternating path $da \cup P^a$.

    - S_i shrinks.

- Suppose a, b, and c are in $C(S_i) \cup Ext(S_i)$. Note: It follows that $a, b, c \in C(S_i)$.
    - dc is a pendant edge.
        - For pendant edge dc use alternating path $dc \cup ca \cup P_a^u \cup P^{uv} \cup P_b^v \cup bc \cup P^c$.
        - For cross edge db use alternating path $db \cup P^b$.
        - For cross edge da use alternating path $da \cup P^a$.
    - S_i shrinks.

S_i **has model M8:**

- Suppose a, b, and d are in $C(S_i) \cup Ext(S_i)$. (In this case, d would be the lowest node in C, hence a, b, and d are in $C(S_i)$. This follows from Proposition 8).
    - For all substructures S_j that contain d, remove d from $C(S_j)$.
    - Add c to $Int(S_i)$.
    - Set $P^c := ca \cup P_a^u \cup P^{uv} \cup P^v$.
    - dc becomes the base edge for a type 1 shrunk node.
        - For cross edge da use alternating path $da \cup P^a$. (Similarly for cross edge db.)
    - ea and fb are pendant edges.
        - for pendant edges ea and fb, alternating paths are **determined elsewhere**.
        - For cross edge ec use alternating path $ec \cup P^c$. (Similarly for cross edge fc.)
    - S_i switches to a substructure with model M6.

- Suppose a (or b or d) is the only node in $C(S_i) \cup Ext(S_i)$.
    - S_i retains its type.

- Suppose a and d are the only nodes in $C(S_i) \cup Ext(S_i)$. (Similarly for b and d.)
    - Add c to $Int(S_i)$.
    - Set $P^c := ca \cup P_a^u \cup P^{uv} \cup P^v$.
    - bc, ea, and gd are pendant edges.
        - Alternating path for pendant edge bc is already defined.
        - Alternating path for pendant edge ea is **determined elsewhere**.

    - For pendant edge gd use alternating path $gd \cup P^u_d \cup P^{uv} \cup P^v$.
    - For cross edge ec use alternating path $ec \cup P^c$.
    - For cross edge gc use alternating path $gc \cup P^c$.
    - For cross edge ba use alternating path $ba \cup P^a$.
    - For cross edge bd use alternating path $bd \cup P^d$.
  - S_i switches to substructure with model M5.
- a and b are the only nodes in $C(S_i) \cup Ext(S_i)$ is not possible.

S_i **has model M9:**

- Suppose a, b, and d are in $C(S_i) \cup Ext(S_i)$.
  - Add c to $Int(S_i)$.
  - Set $P^c := ca \cup P^u_a \cup P^{uv} \cup P^v$.
  - ea and gd are pendant edges.
    - Alternating path for pendant edge ea is **determined elsewhere**.
    - For pendant edge gd use alternating path $gd \cup P^u_d \cup P^{uv} \cup P^v$.
    - For cross edge ec use alternating path $ec \cup P^c$.
    - For cross edge gc use alternating path $gc \cup P^c$.
    - S_i shrinks.
- Suppose a (or b or d) is the only node in $C(S_i) \cup Ext(S_i)$.
  - S_i retains its type.
- Suppose a and d are the only nodes in $C(S_i) \cup Ext(S_i)$.
  - Add c to $Int(S_i)$.
  - Set $P^c := ca \cup P^u_a \cup P^{uv} \cup P^v$.
  - bc, ea, and gd are pendant edges.
    - For pendant edge bc use alternating path $bc \cup cd \cup P^d$.
    - Alternating path for pendant edge ea is **determined elsewhere**.
    - For pendant edge gd use alternating path $gd \cup P^u_d \cup P^{uv} \cup P^v$.
    - For cross edge ec use alternating path $ec \cup P^c$.
    - For cross edge gc use alternating path $gc \cup P^c$.
    - For cross edge ba use alternating path $ba \cup P^a$.
    - For cross edge bd use alternating path $bd \cup P^d$.
  - S_i switches to substructure with model M2.
- a and b are the only nodes in $C(S_i) \cup Ext(S_i)$ is not possible. (Because d is not in $C(S_i) \cup Ext(S_i)$, a must be the lowest node in $C(S_j) \cup Ext(S_j)$, for some j, on any di-path in D from a node in C to a root. Hence, the di-path in D from b would have to go through a on the way to a root. But this situation is ruled out by S13.)

- Suppose b and d are the only nodes in $C(S_i) \cup Ext(S_i)$.
  - Add c to $Int(S_i)$.
  - Set P^c along cb: If b is shrunk, it is already determined; otherwise, it is determined when cb is a pendant edge for another substructure that contains b.
  - ac and gd are pendant edges.
    - For pendant edge ac the alternating path is determined: P^a.
    - For pendant edge gd: If d is white, use alternating path $gd \cup dc \cup P^c$. Note that this path is well-defined because we have the property that P^b does not pass through ad. If d is shrunk, the alternating path for gd is already determined.
    - For cross edge ab use alternating path $ab \cup bc \cup cd \cup P^d$.
    - For cross edge ad use alternating path $ad \cup P^d$.
    - For cross edge gc use alternating path $gc \cup P^c$.
  - S_i switches to substructure with model M5 (based on pendant edge ac).

S_i **has model M10:**

- Suppose a, b, and d are in $C(S_i) \cup Ext(S_i)$.
  - Add c to $Int(S_i)$.
  - Set P^c along ca: It is determined when ca is a pendant edge for another substructure that contains a.
  - gd is a pendant edge.
    - For pendant edge gd use alternating path $gd \cup dc \cup P^c$.
    - For cross edge gc use alternating path $gc \cup P^c$.
    - S_i shrinks.
- Suppose a is the only node in $C(S_i) \cup Ext(S_i)$.
  - The node c becomes white.
  - P^c is determined when ca is a pendant edge for another substructure that contains a.
  - The edge ca becomes a substructure with model M5. The edge cb becomes a substructure with model M2. The edge dc becomes a substructure with model M24.
  - Handle the case for b analogously.
- Suppose d is the only node in $C(S_i) \cup Ext(S_i)$.
  - S_i retains its type.

- Suppose a and d are the only nodes in $C(S_i) \cup Ext(S_i)$.
  - Add c to $Int(S_i)$.
  - P^c: $ca \cup P_a^u \cup P^{uv} \cup P^v$.
  - bc and gd are pendant edges.
    - For pendant edge bc use alternating path $bc \cup cd \cup P^d$.
    - For pendant edge gd use alternating path $gd \cup dc \cup P^c$.
    - For cross edge ba use alternating path $ba \cup P^a$.
    - For cross edge bd use alternating path $bd \cup P^d$.
    - For cross edge gc use alternating path $gc \cup P^c$.
  - S_i switches to a substructure with model M2 on edge bc.
  - Handle analogously if b and d are the only nodes in $C(S_i) \cup Ext(S_i)$.
- Suppose a and b are the only nodes in $C(S_i) \cup Ext(S_i)$.
  - Note: The special case where u and v define a substructure with model M24 with endnodes a and b was handled in Subroutine 2.
  - Node c becomes a white node.
  - P^c: Determined when ca is a pendant edge for another substructure that contains a.
  - The edge dc becomes a substructure with model M24. ca becomes a substructure with model M5 and cb becomes a substructure with model M2.

S_i **has model M11:**

- Suppose a (or b) is the only node in $C(S_i) \cup Ext(S_i)$.
  - S_i retains its type.
- Suppose c is the only node in $C(S_i) \cup Ext(S_i)$.
  - Note that bc becomes a pendant edge for the new shrunk node containing c. The alternating path for bc through the new shrunk node to a root is determined when bc is pendant for another substructure that contains c. If the path in D from c goes to b, then this path uses the edge bc twice, so it is not truly an alternating path. However, if an exchange is performed at some point in the algorithm, we use only the portion of this path from bc to the base node of c (which may further shrink within this substructure with model M11), which is a type 2 alternating path. This portion is used only if S_i becomes temporary, in which case S_i shrinks. So we truncate this alternating path for bc to b or its base node, if it is shrunk.
  - S_i retains its type.

- Suppose a and b are the only nodes in $C(S_i) \cup Ext(S_i)$.
  - Nodes a and b shrink to produce a substructure with model M3. Therefore, edge bc would not be a pendant edge for the new shrunk node (by the definition of pendant edges). However, we must consider the following: Suppose this new substructure with model M3 were to be chosen as S_t at some future point in Step 2 of the Main Algorithm. The associated new alternating path P^{uv}_{new} through this new substructure is discussed under "additional examples of alternating paths" in Section 5.4. The portion of P^{uv}_{new} that begins with edge ca is the subpath of $ca \cup ab \cup P^u_b \cup P^{uv} \cup P^v$ that ends at the base node of the new node we are shrinking here.
- Suppose b and c are the only nodes in $C(S_i) \cup Ext(S_i)$.
  - Add a to $Int(S_i)$.
  - S_i shrinks.
- Suppose a and c are the only nodes in $C(S_i) \cup Ext(S_i)$.
  - Add b to $Int(S_i)$.
  - S_i shrinks.
- Suppose a, b, and c are in $C(S_i) \cup Ext(S_i)$.
  - S_i shrinks.

S_i **has model M12:**

- Suppose a (or b) is the only node in $C(S_i) \cup Ext(S_i)$.
  - S_i retains its type.
- Suppose c is the only node in $C(S_i) \cup Ext(S_i)$.
  - Add a and b to $Int(S_i)$.
  - dc is a pendant edge.
    * For pendant edge dc the alternating path is **determined elsewhere**.
    * For cross edge db use alternating path $db \cup P^b$.
    * For cross edge da use alternating path $da \cup P^a$.
  - S_i shrinks.
- Suppose a and b are the only nodes in $C(S_i) \cup Ext(S_i)$.
  - Add c to $Int(S_i)$.

- dc is a pendant edge.
  - For pendant edge dc use alternating path $dc \cup ca \cup ab \cup P^u_b \cup P^{uv} \cup P^v$.
  - For cross edge db use alternating path $db \cup P^b$.
  - For cross edge da use alternating path $da \cup P^a$.
- S_i shrinks.

- Suppose a and c are the only nodes in $C(S_i) \cup Ext(S_i)$.
  - Add b to $Int(S_i)$.
  - dc is a pendant edge.
    - For pendant edge dc the alternating path is **determined elsewhere**.
    - For cross edge db use alternating path $db \cup P^b$.
    - For cross edge da use alternating path $da \cup P^a$.
  - S_i shrinks.
- Not possible to have b and c as the only nodes in $C(S_i) \cup Ext(S_i)$.
- Suppose a, b, and c are in $C(S_i) \cup Ext(S_i)$.
  - dc is a pendant edge.
    - For pendant edge dc the alternating path is **determined elsewhere**.
    - For cross edge db use alternating path $db \cup P^b$.
    - For cross edge da use alternating path $da \cup P^a$.
  - S_i shrinks.

S_i **has model M13:**

- Suppose a is the only node in $C(S_i) \cup Ext(S_i)$ and b is white; or suppose b is the only node in $C(S_i) \cup Ext(S_i)$ and a is white.
  - S_i retains its type.
- Suppose a is the only node in $C(S_i) \cup Ext(S_i)$ and b is shrunk (hence a is white); or suppose b is the only node in $C(S_i) \cup Ext(S_i)$ and a is shrunk (hence b is white).
  - Node c becomes white.
  - Suppose a is white and b is shrunk. Let b_b be its base node. Consider the type 2 alternating path from ab through b to b_b; let P^* denote this path minus the edge ab. Note that a is in a different substructure from S_i, say S_j, that contains an edge will be shrinking and for which ab is a pendant edge. Hence, when S_j is considered in this subroutine, an alternating path from ab to a root will be identified; call it P^{**}. Let $P^c := cb_b \cup P^* \cup P^{**}$.

  - Suppose a is shrunk and b is white. Consider the extension of the type 2 alternating path from ab through a to a root; call it P^*. Note that b is in a different substructure from S_i, say S_j, that contains an edge will be shrinking and for which ab is a pendant edge. Hence, when S_j is considered in this subroutine, an alternating path from ab through b and then through c and a to a root will be identified; let P^{**} denote the portion of this path to b. Let $P^c := cb_b \cup P^* \cup P^{**}$.
  - Triangle abc becomes a substructure with model M12.
  - Update the substructure containing dc with c white.

- Suppose a and b are the only nodes in $C(S_i) \cup Ext(S_i)$.
  - Add c to $Int(S_i)$.
  - Set $P^c := cb \cup P^u_b \cup P^{uv} \cup P^v$.
  - dc is a pendant edge.
    - For pendant edge dc use alternating path $dc \cup ca \cup ab \cup P^u_b \cup P^{uv} \cup P^v$.
    - For cross edge db use alternating path $db \cup P^b$.
    - For cross edge da use alternating path $da \cup P^a$.
  - S_i shrinks.

S_i **has model M14:**

- Suppose a (or b or c) is the only node in $C(S_i) \cup Ext(S_i)$.
  - S_i retains its type.
- Suppose a and b are the only nodes in $C(S_i) \cup Ext(S_i)$.
  - Nodes a and b shrink to produce a substructure with model M3. Therefore, edge ac would not be a pendant edge for the new shrunk node (by the definition of pendant edges). However, we must consider the following: Suppose this new substructure with model M3 were to be chosen as S_t at some future point in Step 2 of the Main Algorithm. The associated new alternating path P^{uv}_{new} through this new substructure is discussed under "additional examples of alternating paths" in Section 5.4. The portion of P^{uv}_{new} that begins with edge cb is the subpath of $cb \cup P^u_b \cup P^{uv} \cup P^v$ that ends at the base node of the new node we are shrinking here.
- Suppose b and c are the only nodes in $C(S_i) \cup Ext(S_i)$.
  - Add a to $Int(S_i)$.
  - S_i shrinks.

- Suppose a and c are the only nodes in $C(S_i) \cup Ext(S_i)$.
  - Add b to $Int(S_i)$.
  - S_i shrinks.
- Suppose a, b, and c are in $C(S_i) \cup Ext(S_i)$.
  - S_i shrinks.

S_i **has model M15:**

- Suppose a (or b) is the only node in $C(S_i) \cup Ext(S_i)$.
  - S_i retains its type.
- Suppose c is the only node in $C(S_i) \cup Ext(S_i)$.
  - Add a and b to $Int(S_i)$.
  - dc is a pendant edge.
    - For pendant edge dc the alternating path is **determined elsewhere**.
    - For cross edge db use alternating path $db \cup P^b$.
    - For cross edge da use alternating path $da \cup ac \cup cb \cup P^b$ (where P^b is from the endnode of cb inside b).
  - S_i shrinks.
- Suppose a and b are the only nodes in $C(S_i) \cup Ext(S_i)$.
  - Add c to $Int(S_i)$.
  - dc is a pendant edge.
    - For pendant edge dc use alternating path $dc \cup cb \cup P^u_b \cup P^{uv} \cup P^v$.
    - For cross edge db use alternating path $db \cup P^b$.
    - For cross edge da use alternating path $da \cup ac \cup cb \cup P^b$ (where P^b is from the endnode of cb inside b).
  - S_i shrinks.
- Suppose a and c are the only nodes in $C(S_i) \cup Ext(S_i)$.
  - Add b to $Int(S_i)$.
  - dc is a pendant edge.
    - For pendant edge dc the alternating path is **determined elsewhere**.
    - For cross edge db use alternating path $db \cup P^b$.

        - ∗ For cross edge da use alternating path $da \cup ac \cup cb \cup P^b$ (where P^b is from the endnode of cb inside b).
    - S_i shrinks.

- Not possible to have b and c as the only nodes in $C(S_i) \cup Ext(S_i)$.
- Suppose a, b, and c are in $C(S_i) \cup Ext(S_i)$.
    - dc is a pendant edge.
        - ∗ For pendant edge dc the alternating path is **determined elsewhere**.
        - ∗ For cross edge db use alternating path $db \cup P^b$.
        - ∗ For cross edge da use alternating path $da \cup ac \cup c \cup P^b$ (where P^b is from the endnode of cb inside b).
    - S_i shrinks.

S_i **has model M16:**

- Suppose b is the only node in $C(S_i) \cup Ext(S_i)$.
    - S_i retains its type.
- Suppose a is the only node in $C(S_i) \cup Ext(S_i)$.
    - Node c becomes white.
    - Alternating path for c: Path starting with ca is determined when ca is a pendant edge for another substructure that contains a.
    - Triangle abc becomes a substructure with model M15.
    - Update the substructure containing dc with c white.
- Suppose a and b are the only nodes in $C(S_i) \cup Ext(S_i)$.
    - Add c to $Int(S_i)$.
    - Set $P^c := ca \cup P^u_a \cup P^{uv} \cup P^v$.
    - dc is a pendant edge.
        - ∗ For pendant edge dc use alternating path $dc \cup cb \cup P^u_b \cup P^{uv} \cup P^v$.
        - ∗ For cross edge db use alternating path $db \cup P^b$.
        - ∗ For cross edge da use alternating path $da \cup ac \cup cb \cup P^b$ (where P^b is from the endnode of cb inside b).
    - S_i shrinks.

S_i **has model M17:**

- Suppose a (or b or d or e) is the only node in $C(S_i) \cup Ext(S_i)$.

  - S_i retains its type.

- Suppose a and b are the only nodes in $C(S_i) \cup Ext(S_i)$.
  - Add c to $Int(S_i)$.
  - Set $P^c := cb \cup P_b^u \cup P^{uv} \cup P^v$.
  - dc is a pendant edge.
    * For pendant edge dc use alternating path $dc \cup ca \cup ab \cup P_b^u \cup P^{uv} \cup P^v$. Reroute all previously defined alternating paths, that used $dc \cup ce$, along this new path.
    * For cross edge db use alternating path $db \cup P^b$.
    * For cross edge da use alternating path $da \cup P^a$.
  - Edge ce is deleted and nodes a, b, and c shrink to a new node with dc as a substructure with model M5.

- Suppose d and e are the only nodes in $C(S_i) \cup Ext(S_i)$.
  - Add c to $Int(S_i)$.
  - Set $P^c := cd \cup P_d^u \cup P^{uv} \cup P^v$.
  - bc, fd, and ge are pendant edges.
    * For pendant edge bc: If e is shrunk, let e' be the endnode of ce inside e. Otherwise, let $e' = e$. Use the alternating path $bc \cup ce' \cup P^{e'}$. If the path in D from e goes to b, then this path uses the edge bc twice, so it is not truly an alternating path. However, if an exchange is performed along this path at some point in the algorithm, we use only the portion of this path from bc to the base node of the new node that we shrink (to create a substructure with model M11 (see the end of this case); this node may further shrink within this substructure with model M11). This is a type 2 alternating path. This path is used only if the new substructure with model M11 becomes temporary in which case the entire substructure shrinks. So we truncate this alternating path for bc to b or its base node, if it is shrunk.
    * For pendant edge fd, alternating path is **determined elsewhere**.
    * For pendant edge ge use alternating path $ge \cup P_e^u \cup P^{uv} \cup P^v$.
    * For cross edge bd use alternating path $bd \cup P^d$.
    * For cross edge fc use alternating path $fc \cup cd \cup P_d^u \cup P^{uv} \cup P^v$.
    * For cross edge gc use alternating path $gc \cup P^c$.
  - Triangle abc becomes a substructure with model M11.

- Suppose b and e are the only nodes in $C(S_i) \cup Ext(S_i)$ and a is white.

  - ge is a pendant edge. Alternating path is **determined elsewhere**.
  - With shrinking of nodes b and e, this becomes a substructure with model M18.

- Suppose b and e are the only nodes in $C(S_i) \cup Ext(S_i)$ and a is shrunk.
  - Add a and c to $Int(S_i)$.
  - Let P^* be the alternating path through a starting with pendant edge ab. Set $P^c := cb \cup P_b^u \cup P^{uv} \cup P_b^v \cup ba \cup P^*$. (Note that b has to be the bottom node of the cycle C.)
  - dc and ge are pendant edges.
    * Alternating path for dc equals P^d.
    * For pendant edge ge use alternating path $ge \cup ec \cup P^c$.
    * For cross edge de use alternating path $de \cup P^e$.
    * For cross edge db use alternating path $db \cup P^b$.
    * For cross edge da use alternating path $da \cup P^a$.
    * With shrinking of nodes a, b, c, and e, this becomes a substructure with model M5 based on edge dc.

- Suppose b, d, and e are the only nodes in $C(S_i) \cup Ext(S_i)$.
  - Add a and c to $Int(S_i)$.
  - Set $P^c := cd \cup P_d^u \cup P^{uv} \cup P^v$.
  - fd and ge are pendant edges.
    * For pendant edge fd use alternating path $fd \cup P_d^u \cup P^{uv} \cup P^v$.
    * For pendant edge ge use alternating path $ge \cup P_e^u \cup P^{uv} \cup P^v$.
    * For cross edge fc use alternating path $fc \cup P^c$.
    * For cross edge gc use alternating path $gc \cup P^c$.
  - S_i shrinks. (Note that the nodes along the path in D from e to a are on C and also shrink.)

- Suppose a, b, and e are the only nodes in $C(S_i) \cup Ext(S_i)$.
  - Add c to $Int(S_i)$.
  - Set $P^c := cb \cup P_b^u \cup P^{uv} \cup P^v$.
  - dc and ge are a pendant edges.
    * Alternating path for pendant edge dc equals P^d.
    * For pendant edge ge: Alternating path is **determined elsewhere**.
    * For cross edge de use alternating path $de \cup P_e^u \cup P^{uv} \cup P^v$.
    * For cross edge db use alternating path $db \cup P^b$.

    * For cross edge da use alternating path $da \cup P^a$.
    * For cross edge gc use alternating path $gc \cup P^c$.
  - Nodes a, b, c, and e shrink. dc becomes a substructure with model M5.

- Suppose a, b, d, and e are the only nodes in $C(S_i) \cup Ext(S_i)$.
  - This case is handled as was the case that b, d, and e are the only nodes in $C(S_i) \cup Ext(S_i)$ (except there is no need to add a to $Int(S_i)$.)

S_i **has model M18:**

- Suppose b (or d) is the only node in $C(S_i) \cup Ext(S_i)$.
  - S_i retains its type.
- Suppose a is the only node in $C(S_i) \cup Ext(S_i)$.
  - Add b and c to $Int(S_i)$.
  - Let b_b be the base node of b. Let P^* be the type 2 alternating path starting with pendant edge ab and ending at b_b. Let P^{**} be the alternating path to a root starting with pendant edge ab for shrinking node a (which is determined when ab is pendant for another substructure that contains a). Set $P^c := cb_b \cup P^* \cup P^{**}$.
  - dc and gb are pendant edges.
    * For pendant edge dc, the alternating path equals P^d.
    * For pendant edge gb, the alternating path is already determined.
    * For three cross edges from d, use existing paths from endnodes.
    * For cross edge gc use alternating path $gc \cup P^c$.
  - S_i becomes a substructure with model M5 based on edge dc.
- Suppose a and b are the only nodes in $C(S_i) \cup Ext(S_i)$.
  - Add c to $Int(S_i)$.
  - Let b_b be the base node of b. Set $P^c := cb_b \cup P^u_b \cup P^{uv} \cup P^v$.
  - dc and gb are pendant edges.
    * For pendant edge dc, the alternating path equals P^d.
    * For pendant edge gb, the alternating path is already determined.
    * For three cross edges from d, use existing paths from endnodes.
    * For cross edge gc use alternating path $gc \cup P^c$.
  - S_i becomes a substructure with model M5 based on edge dc.
- Suppose b and d are the only nodes in $C(S_i) \cup Ext(S_i)$.

- Add a and c to $Int(S_i)$.
- Set $P^c := cd \cup P^u_d \cup P^{uv} \cup P^v$.
- fd and gb are pendant edges.
  * For pendant edge fd, the alternating path is **determined elsewhere**.
  * For pendant edge gb, the alternating path is already determined.
  * For cross edge fc use alternating path $fc \cup P^c$.
  * For cross edge gc use alternating path $gc \cup P^c$.
- S_i shrinks.

- Suppose a, b, and d are the only nodes in $C(S_i) \cup Ext(S_i)$.
  - This case is handled as was the case that b and d are the only nodes in $C(S_i) \cup Ext(S_i)$ (except there is no need to add a to $Int(S_i)$.)

S_i **has model M19:**

- Suppose a is the only node in $C(S_i) \cup Ext(S_i)$ and b is white (or vice versa). Or suppose e is the only node in $C(S_i) \cup Ext(S_i)$ and d is white (or vice versa).
  - S_i retains its type.
- Suppose a is the only node in $C(S_i) \cup Ext(S_i)$ and b is shrunk.
  - Make c a white node.
  - Let P^* be the type 2 alternating path starting with ab for shrunk node b. Let P^{**} be the alternating path to a root starting with ba for shrinking node a (which is determined when ab is pendant for another substructure that contains a). Set $P^c := cb \cup P^* \cup P^{**}$.
  - S_i becomes two substructures: abc has model M12 and cde has model M11.
- Suppose b is the only node in $C(S_i) \cup Ext(S_i)$ and a is shrunk.
  - Make c a white node.
  - Let P^* be the type 2 alternating path starting with ab for shrinking node b (which is determined when ab is pendant for another substructure that contains b). Let P^{**} be the alternating path to a root starting with ba for shrunk node a. Set $P^c := cb \cup P^* \cup P^{**}$.
  - S_i becomes two substructures: abc has model M12 and cde has model M11.
- Suppose e is the only node in $C(S_i) \cup Ext(S_i)$ and d is shrunk; or suppose d is the only node in $C(S_i) \cup Ext(S_i)$ and e is shrunk. Handle as for the previous two cases.

- Suppose d and e are the only nodes in $C(S_i) \cup Ext(S_i)$.
  - Add c to $Int(S_i)$.
  - Set $P^c := cd \cup P^u_d \cup P^{uv} \cup P^v$.
  - bc is a pendant edge.
    - For pendant edge bc: Use path $bc \cup ce \cup ed \cup P^u_d \cup P^{uv} \cup P^v_b$. If this path contains bc a second time, an exchange on the entire path is not well-defined. In this case, if an exchange is performed at some point in the algorithm, we only use the portion of this path from bc to the base node of the new node that we shrink, which is a type 2 alternating path, plus, possibly, an additional portion no further than b; these portions are well-defined and used only if the remaining substructure (with model M11, see below) becomes temporary. We determine the full path in case the new shrunk node shrinks further and the new shrunk node still has model M11.
    - For cross edge bd, use alternating path $bd \cup P^d$.
    - For cross edge be, use alternating path $be \cup P^e$.
  - Triangle abc becomes a substructure with model M11.
- Suppose a and b are the only nodes in $C(S_i) \cup Ext(S_i)$. Handle as for the previous case.
- Suppose b and e are the only nodes in $C(S_i) \cup Ext(S_i)$ and a is white.
  - ab is a new pendant edge if b is white; de is a new pendant edge if e is white.
    - Alternating paths for ab and de are determined when they are pendant edges for substructures with edges not in S_i that contain b and e, respectively.
  - S_i becomes two substructures: de becomes a substructure with model M2 and the remainder becomes a substructure with model M18.
- Suppose b and e are the only nodes in $C(S_i) \cup Ext(S_i)$ and a is shrunk.
  - Add a, c, and d to $Int(S_i)$.
  - Let P^* be the type 2 alternating path starting with ab for shrinking node b (which is determined when ab is pendant for another substructure that contains b). Let P^{**} be the alternating path to a root starting with ba for shrunk node a. Set $P^c := cb \cup P^* \cup P^{**}$.
  - S_i shrinks.
- Suppose a and d are the only nodes in $C(S_i) \cup Ext(S_i)$. Handle as for the previous two cases.

- Suppose a and e are the only nodes in $C(S_i) \cup Ext(S_i)$ and b and d are white.
  - c becomes a grey node and edge ce is removed from the substructures.
  - ba and de are pendant edges.
    - Alternating paths for ba and de are determined when they are pendant edges for other substructures containing a and e, respectively.
  - S_i becomes three substructures: abc becomes a substructure with model M13; cd becomes a substructure with model M22; de becomes a substructure with model M5.
- Suppose a and e are the only nodes in $C(S_i) \cup Ext(S_i)$ and b and/or d is shrunk.
  - Add b, c, and d to $Int(S_i)$.
  - Suppose b is shrunk. Let P^* be the type 2 alternating path starting with ab for shrunk node b. Let P^{**} be the alternating path to a root starting with ba for shrinking node a (which is determined when ab is pendant for another substructure that contains a). Set $P^c := cb \cup P^* \cup P^{**}$. If b is white, use analogous path starting with cd.
  - S_i shrinks.
- Suppose b, d, and e are the only nodes in $C(S_i) \cup Ext(S_i)$.
  - Add a and c to $Int(S_i)$.
  - Set $P^c := cd \cup P_d^u \cup P^{uv} \cup P^v$.
  - S_i shrinks.
- Suppose a, b, and d are the only nodes in $C(S_i) \cup Ext(S_i)$. Handle as for the previous case.
- Suppose a, d, and e are the only nodes in $C(S_i) \cup Ext(S_i)$.
  - Add b and c to $Int(S_i)$.
  - Set $P^c := cd \cup P_d^u \cup P^{uv} \cup P^v$.
  - S_i shrinks.
- Suppose a, b, and e are the only nodes in $C(S_i) \cup Ext(S_i)$. Handle as for the previous case.
- Suppose a, b, d, and e are the only nodes in $C(S_i) \cup Ext(S_i)$.
  - Add c to $Int(S_i)$.
  - Set $P^c := cd \cup P_d^u \cup P^{uv} \cup P^v$.

  - S_i shrinks.

S_i **has model M20:**

- Suppose a (or b or d or e) is the only node in $C(S_i) \cup Ext(S_i)$.
  - S_i retains its type.
- Suppose a and b are the only nodes in $C(S_i) \cup Ext(S_i)$.
  - Add c to $Int(S_i)$.
  - Set $P^c := ca \cup ab \cup P^u_b \cup P^{uv} \cup P^v$.
  - dc is a pendant edge.
    - For pendant edge dc use alternating path $dc \cup cb \cup P^u_b \cup P^{uv} \cup P^v$.
    - For cross edge db use alternating path $db \cup P^b$.
    - For cross edge da use alternating path $da \cup ac \cup cb \cup P^b$.
  - Edge ce is deleted and nodes a, b, and c shrink to a new node with dc as the edge in a substructure with model M5.
- Note that it is not possible that b and e are the only nodes in $C(S_i) \cup Ext(S_i)$. If b and e are in $C(S_i) \cup Ext(S_i)$, then a must also be in $C(S_i) \cup Ext(S_i)$.
- Suppose d and e are the only nodes in $C(S_i) \cup Ext(S_i)$.
  - Add c to $Int(S_i)$.
  - Set $P^c := cd \cup P^u_d \cup P^{uv} \cup P^v$.
  - ac, fd, and ge are pendant edges.
    - For pendant edge ac use alternating path $ac \cup ce \cup P^e$.
    - Alternating path for fd is **determined elsewhere**.
    - For pendant edge ge use alternating path $ge \cup P^u_e \cup P^{uv} \cup P^v$.
    - For cross edge ad use alternating path $ad \cup P^d$.
    - For cross edge fc use alternating path $fc \cup P^c$.
    - For cross edge gc use alternating path $gc \cup P^c$.
  - Triangle abc becomes a substructure with model M14.
- Suppose a and e are the only nodes in $C(S_i) \cup Ext(S_i)$.
  - Add b and c to $Int(S_i)$.
  - Set $P^c := ca \cup P^u_a \cup P^{uv} \cup P^v$.
  - dc and ge are pendant edges.
    - Alternating path for dc is already determined.
    - Alternating path for ge is **determined elsewhere**.
    - For cross edge db use alternating path $db \cup P^b$.

    * For cross edge de use alternating path $de \cup P^e$.
    * For cross edge da use alternating path $da \cup ac \cup cb \cup P^b$.
    * For cross edge gc use alternating path $gc \cup ca \cup P_a^u \cup P^{uv} \cup P^v$.
  - With shrinking of nodes a, b, c, and e, this becomes a substructure with model M5 based on the edge dc.

- Suppose a, d, and e are the only nodes in $C(S_i) \cup Ext(S_i)$.
  - Add b and c to $Int(S_i)$.
  - Set $P^c := cd \cup P_d^u \cup P^{uv} \cup P^v$.
  - fd and ge are pendant edges.
    * Alternating path for fd is **determined elsewhere**.
    * For pendant edge ge use alternating path $ge \cup P_e^u \cup P^{uv} \cup P^v$.
    * For cross edge fc use alternating path $fc \cup P^c$.
    * For cross edge gc use alternating path $gc \cup P^c$.
  - S_i shrinks. (Note that the nodes along the path in D from e to a are on C and also shrink.)

- Suppose a, b, and e are the only nodes in $C(S_i) \cup Ext(S_i)$.
  - Add c to $Int(S_i)$.
  - Set $P^c := ca \cup ab \cup P_b^u \cup P^{uv} \cup P^v$.
  - dc and ge are pendant edges.
    * Alternating path for pendant edge dc equals P^d.
    * For pendant edge ge: Alternating path is **determined elsewhere**.
    * For cross edge de use alternating path $de \cup P^e$.
    * For cross edge da use alternating path $da \cup ac \cup cb \cup P^b$.
    * For cross edge db use alternating path $db \cup P^b$.
    * For cross edge gc use alternating path $gc \cup P^c$.
  - Nodes a, b, c, and e shrink. dc becomes a substructure with model M5.

- Suppose a, b, d, and e are the only nodes in $C(S_i) \cup Ext(S_i)$.
  - This case is handled as was the case that a, d, and e are the only nodes in $C(S_i) \cup Ext(S_i)$ (except there is no need to add b to $Int(S_i)$.)

S_i **has model M21, M22, or M23:**

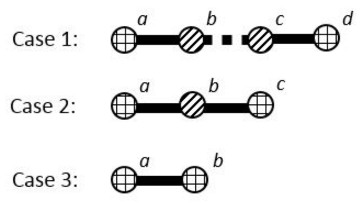


Figure 17: Path substructures

- We consider three general cases for S_i (see Figure 17). In each case, if an endnode of S_i has been previously identified as switching to white from grey (due to an alteration of a substructure with model M13 or M16 during the current execution of Subroutine 1^*), then we treat it here as if it is already white. The interior nodes of each path are striped. The path in Case 1 contains 4 or more nodes. After identifying new substructures in each case, a subpath of S_i may not be explicitly assigned a new type of substructure. Assign such subpaths the appropriate model depending on the type of their endnodes. (For example, in Case 1, if a and d are the only nodes in $C(S_i) \cup Ext(S_i)$, then b and c become white and the subpath between them becomes a substructure with model M2 or M23, depending on its length.)

- **Case 1:**

  - Suppose a is the only node in $C(S_i) \cup Ext(S_i)$.
    - b becomes white.
    - The alternating path for ba is determined when it is a pendant edge for a substructure containing a.
    - ba becomes a substructure with model M5.
  - Similarly if d, or a and d are the only nodes in $C(S_i) \cup Ext(S_i)$.

- **Case 2:**

  - Suppose a is the only node in $C(S_i) \cup Ext(S_i)$, or a and c are the only nodes in $C(S_i) \cup Ext(S_i)$. (We assume a and c are not white with $uv = ac$ a substructure with model M24. This case is handled in Subroutine 2.)
    - b becomes white.
    - The alternating path for ba is determined when it is a pendant edge for a substructure containing a.
    - ba becomes a substructure with model M5.
  - Similarly if c is the only node in $C(S_i) \cup Ext(S_i)$.

- **Case 3:**

  - Suppose a is the only node in $C(S_i) \cup Ext(S_i)$ and b is grey.
    - b becomes white.
    - The alternating path for ba is determined when it is a pendant edge for a substructure containing a.
    - ba becomes a substructure with model M5. Update the other substructure S_j that contains b: If it has model M13, switch it to a substruture with model M11; if it has model M16 switch it to a substructure with model M14.
  - Suppose a is the only node in $C(S_i) \cup Ext(S_i)$ and b is white.
    - The alternating path for ba is determined when it is a pendant edge for a substructure containing a.
    - ba becomes a substructure with model M2.

S_i **has model M24:**

- Suppose a (or b) is the only node in $C(S_i) \cup Ext(S_i)$.
  - S_i retains its type.
- Suppose a and b are in $C(S_i) \cup Ext(S_i)$ and ab does not play the role of uv.
  - S_i shrinks.
- Suppose a and b are in $C(S_i) \cup Ext(S_i)$, ab plays the role of uv, and the nodes c and d are not identified. (The case that they are identified is handled in Subroutine 2.)
  - ca is a pendant edge. (Similarly for pendant edge db.)
    - For pendant edge ca, if a is white, use alternating path $ca \cup P^{uv} \cup P^{v}$.
    - For cross edge cb, use alternating path $cb \cup P^{v}$.
  - S_i shrinks.

End

Note: The phrase "determined elsewhere" is used a number of times (and highlighted in bold) in Subroutine 1* in reference to defining alternating paths of type 2 for pendant edges. Let us explain this in more detail. There are two cases to consider for a substructure S_i. In both cases the pendant edge for S_i is represented by a bold dashed edge in the corresponding model's figure. Let x denote the endnode of this pendant edge that is contained in S_i.

Case 1: x is a white in-node in $C(S_i) \cup Ext(S_i)$. Examples from Subroutine 1*:

- For model M5: $a, b \in C(S_i) \cup Ext(S_i)$; $x = a$; ca is pendant.
- For model M8: $a, d \in C(S_i) \cup Ext(S_i)$; $x = a$; ea is pendant.
- For model M12: $c \in C(S_i) \cup Ext(S_i)$; $x = c$; dc is pendant.

Case 2: x is a white out-node in $C(S_i) \cup Ext(S_i)$. Examples from Subroutine 1^*:

- For model M7: $c \in C(S_i) \cup Ext(S_i)$; $x = c$; dc is pendant.
- For model M17: $b, e \in C(S_i) \cup Ext(S_i)$; $x = e$; ge is pendant.
- For model M20: $a, e \in C(S_i) \cup Ext(S_i)$; $x = e$; ge is pendant.

Suppose we have $x \in C(S_i)$ in Case 1 or 2. Then (1) there must exist an arc zx in D with nodes z and x in C and its head at x; or (2) there must exist a temporary substructure S_j that defines C with either u or v equal to x, say v. For subcase (1), consider the substructure S_k that contains nodes z and x. If we examine all types of substructures in Subroutine 1^* that can play the role of S_k (with $z, x \in C(S_k) \cup Ext(S_k)$ and z an in-node with corresponding out-node x, and pendant edge at x) we see that an alternating path is defined for the pendant edge (and passes through S_k). The only possibility for the subcase (2) (since v is white and incident with a pendant edge) is that the model for S_j is M24; when this occurs, the alternating path for the pendant edge at v is defined in Subroutine 1^* when S_j is considered.

Next suppose $x \in Ext(S_i)$. By Step 1 of Subroutine 1, x must have previously entered $Int(S_j)$ for some $j \neq i$. In examining the cases in Subroutine 1^*, there are only two ways that x can enter $Int(S_j)$ and then be incident with a pendant edge: As node c for model M12 or M15, where $a, b \in C(S_j) \cup Ext(S_j)$, in both situations. Since, in both situations, c is a white in-node, it cannot play the role of x as an in-node in S_i, since x would be an in-node in two substructures, which is not allowed by the definition of D. Hence, we are left with the possibility that x is an out-node of S_i, which is the remainder of Case 2 above. An examination of Subroutine 1^* shows that we must have one of the three examples given above for Case 2. In each case the alternating path is defined for the pendant edge when S_j is considered in Subroutine 1^*.

6.3 Subroutine 2: Substructure transformation cases

Subroutine 2: Substructure Transformation Cases

In this subroutine, we first check if certain conditions hold (called the Transformation Condition and the Flipped Transformation Condition). If one of these conditions holds, then we perform a transformation on some substructures, including the substructure uv with model M24, which serves as input to this subroutine from Step 2, part 1 of the Main Algorithm. The outcome of the transformation is an update of the substructures so that the edge uv becomes

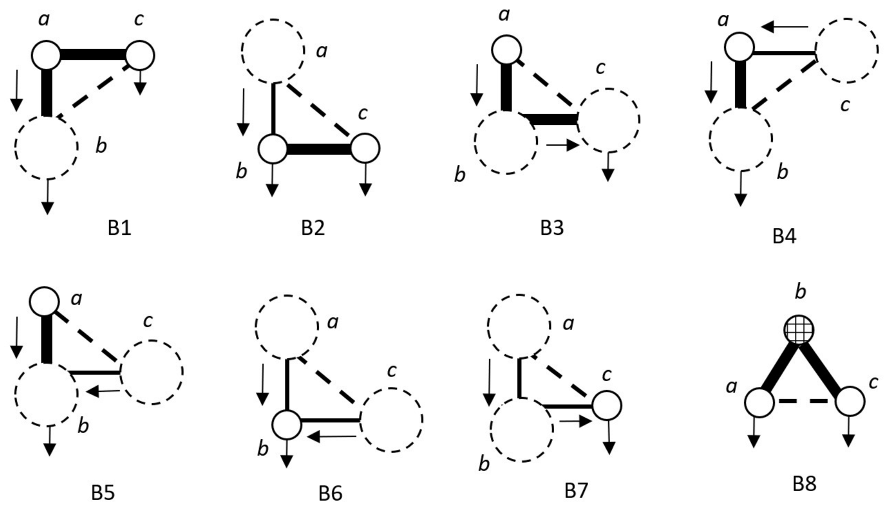
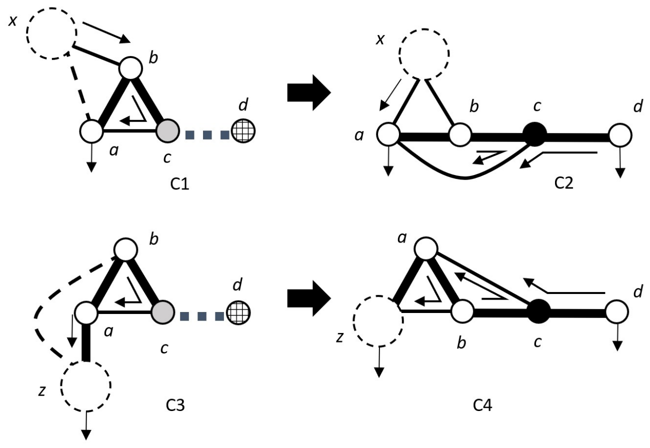


Figure 18: Substructure transformations - 1



Note: The two arrows from node *d* in graphs C2 and C4 indicate that the node can be either an in-node for the substructure containing node *c* or an out-node for that substructure.

Figure 19: Substructure transformations - 2

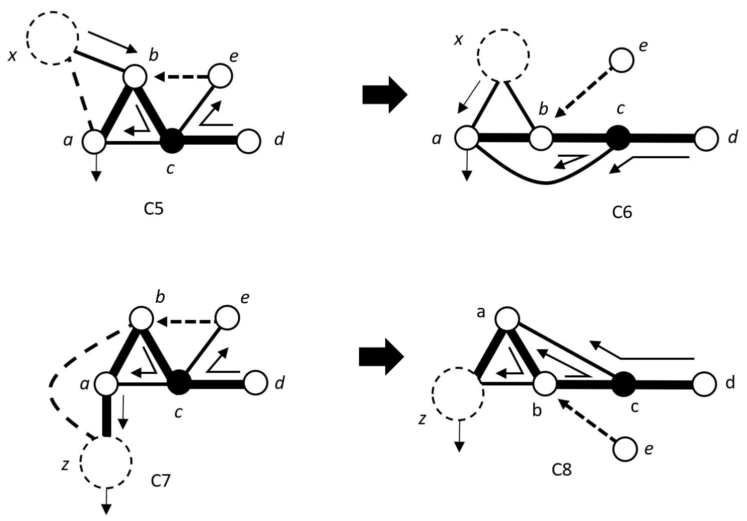


Figure 20: Substructure transformations - 3

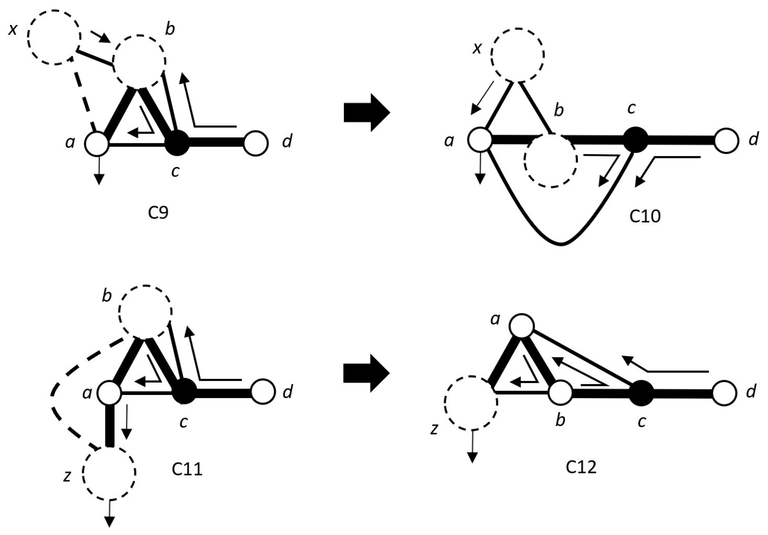


Figure 21: Substructure transformations - 4

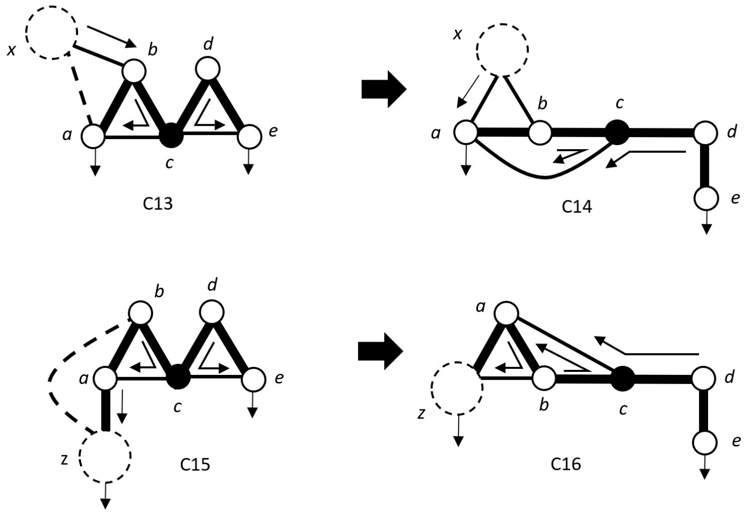
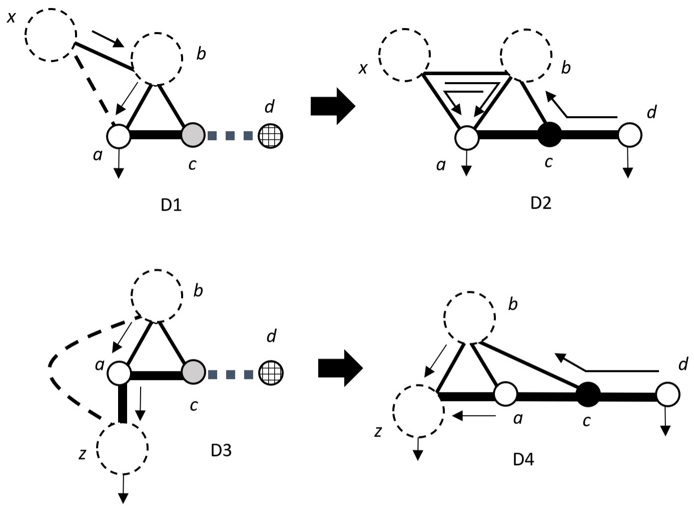


Figure 22: Substructure transformations - 5



Note: The two arrows from node *d* in graphs D2 and D4 indicate that the node can be either an in-node for the substructure containing node *c* or an out-node for that substructure.

Figure 23: Substructure transformations - 6

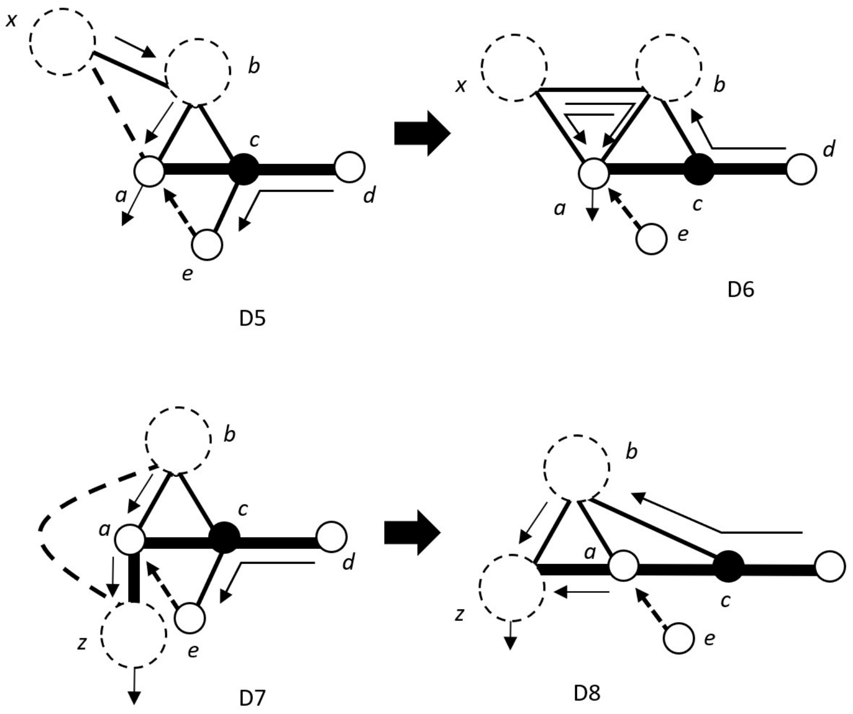


Figure 24: Substructure transformations - 7

an edge in a new, larger substructure. If neither condition holds, we do nothing. After the subroutine finishes, we return to Step 2, part 1 in the Main Algorithm.

The two Transformation Conditions reference Figures 18 to 24, so we begin by describing the content of these figures. (Note: The node labels on the substructures in these figures need not agree with the node labels on the corresponding model graphs in Figures 6 to 9.)

Consider the graphs in Figure 18 and the graphs on the left in Figures 19 to 24. These graphs satisfy the following properties.

- Each of the graphs B1,…, B7 in Figure 18 (without the thin dashed edges) represents a combination of two substructures with models M2, M5, or M6.
- In each graph B1, ..., B7, at least one of the nodes is white (not shrunk).
- The path in M of length 2 in B8 is a substructure with model M23, where b is a striped node, or the path in M in a substructure with model M10, where b is a black node.
- Each of the graphs C1 and C3 in Figure 19 (without the thin dashed edges) represents a combination of three substructures with models M5 or M6; M13; and a portion of M21 or M22 (indicated by the thick dashed edge).
- Each of the graphs C5 and C7 in Figure 20 (without the thin dashed edges) represents a combination of two substructures with models M5 or M6; and M17.
- Each of the graphs C9 and C11 in Figure 21 (without the thin dashed edges) represents a combination of two substructures with models M5 or M6; and M18.
- Each of the graphs C13 and C15 in Figure 22 (without the thin dashed edges) represents a combination of two substructures with models M5 or M6; and M19.
- Each of the graphs D1 and D3 in Figure 23 (without the thin dashed edges) represents a combination of three substructures with models M5 or M6; M16; and a portion of M21 or M22 (indicated by the thick dashed edge).
- Each of the graphs D5 and D7 in Figure 24 (without the thin dashed edges) represents a combination of two substructures with models M5 or M6; and M20.

Transformation Condition: There exist substructures in $\tilde{G}$ that form one of the configurations shown in the graphs in Figure 18 or one of the graphs on the left in Figures 19 to 24 where, in each graph, the thin dashed edge represents the edge uv from the Main Algorithm (a substructure with model M24).

Flipped Transformation Condition: There exist substructures in $\tilde{G}$ that form one of the configurations shown in the graphs C1, C5, C9, C13, D1, and

D5, in Figures 19 to 24 except, in each graph, the thin dashed edge xa represents a substructure with model M6 and the edge xb represents the edge uv from the Main Algorithm (a substructure with model M24). In short, the roles of xa and xb are reversed from the Transformation Condition for these graphs.

If the Transformation Condition holds, then perform one of the following transformations, depending on which configuration we have. In Figures 19 to 24, the transformation for each graph on the left is illustrated by the corresponding graph to the right. If the Flipped Transformation Condition holds, then perform the analogous transformation where the roles of xa and xb are reversed.

- The inputted edge uv, which is a substructure with model M24, is depicted by the thin dashed edges. This edge forms a triangle in $\tilde{G}$, as depicted, and this triangle is also a triangle in G.

B1: Transform into a substructure with model M11. Let b' be the endnode of ab inside b. The alternating path from a becomes $ac \cup cb \cup P^{b'}$. Update, accordingly, other alternating paths that pass through this new substructure.

B2: Transform into a substructure with model M14.

B3: Transform into a substructure with model M12. Let c' be the endnode of bc inside c. The alternating path from b becomes $ba \cup ac \cup P^{a'}$. Update, accordingly, other alternating paths that pass through this new substructure.

B4: Transform into a substructure with model M15. Let b' be the endnode of cb inside b. The alternating path from a becomes $cb \cup P^{b'}$. Update, accordingly, other alternating paths that pass through this new substructure.

B5: Transform into a substructure with model M15.

B6, B7: Transform into a substructure with model M7. For B6, let c' be the endnode of ac inside c. The alternating path from a becomes $ac \cup P^{c'}$. Update, accordingly, other alternating paths that pass through this new substructure. The base edges for a and b remain the same, according to the definition of base edges. For B7, update the alternating paths analogously. The base edge for a changes to ac.

B8: If the path in M is contained in a substructure with model M10, then delete from $\tilde{G}$ the non-matching edge incident with the black node. In both cases, transform into a substructure with model M11. Make node b a white node with alternating path $bc \cup ca \cup P^a$.

C1: Transform into two substructures (see C2) with models M14 and either M8 or M9 (depending on node d). If a is shrunk, let a' be the endnode of ac; otherwise, let $a' := a$. Note that node d in C1 is either white, grey, or striped. If d is white, then it keeps this type and the substructure with model M22 in C1 that contains cd becomes part of a substructure in C2 with model M9. If the path in D from node d goes to node b, then we have a violation of property S13. We fix this as follows: Reroute all alternating paths that contains d along the path $dc \cup ca \cup P^{a'}$. (Note that this shortens all such paths, so this operation cannot "cycle" in the algorithm. Also, performing an exchange on such a path creates a triangle a, b, c in M, which is later removed by the extension P in Step 2 of the Main Algorithm.) If d is grey, then it becomes white in a substructure in C2 with model M8 and alternating path $dc \cup ca \cup P^{a'}$. Also, the other substructure with model M13 or M16 that contains d switches to a substructure with model M11 or M14, respectively. If d is striped, then it becomes white in a substructure in C2 with model M8 and alternating path $dc \cup ca \cup P^{a'}$. The substructure that contained d is updated to one with model M22 or M23, as appropriate. (See substructures property S8.)

C3: Transform into two substructures (see C4) with models M11 and either M8 or M9 (depending on node d). Update d as for the C1 case.

C5: Transform into two substructures (see C6) with models M14 and M8. (The edge ce is removed from the substructures.)

C7: Transform into two substructures (see C8) with models M11 and M8. (The edge ce is removed from the substructures.)

C9: Transform into two substructures (see C10) with models M14 and M8. (The edge cb, which is not in M, is removed from the substructures.)

C11: Transform into two substructures (see C12) with models M11 and M8. (The edge cb, which is not in M, is removed from the substructures.)

C13: Transform into three substructures (see C14) with models M14, M8, and M2. (The edge ce is removed from the substructures.)

C15: Transform into three substructures (see C16) with models M11, M8, and M2. (The edge ce is removed from the substructures.)

D1: Transform into two substructures (see D2) with models M7 and either M9 or M10 (depending on node d). Update d as for the C1 case.

D3: Transform into two substructures (see D4) with models M15 and either M9 or M10 (depending on node d). Update d as for the C1 case.

D5: Transform into two substructures (see D6) with models M7 and M9. (The edge ce is removed from the substructures.)

D7: Transform into two substructures (see D8) with models M15 and M9. (The edge ce is removed from the substructures.)

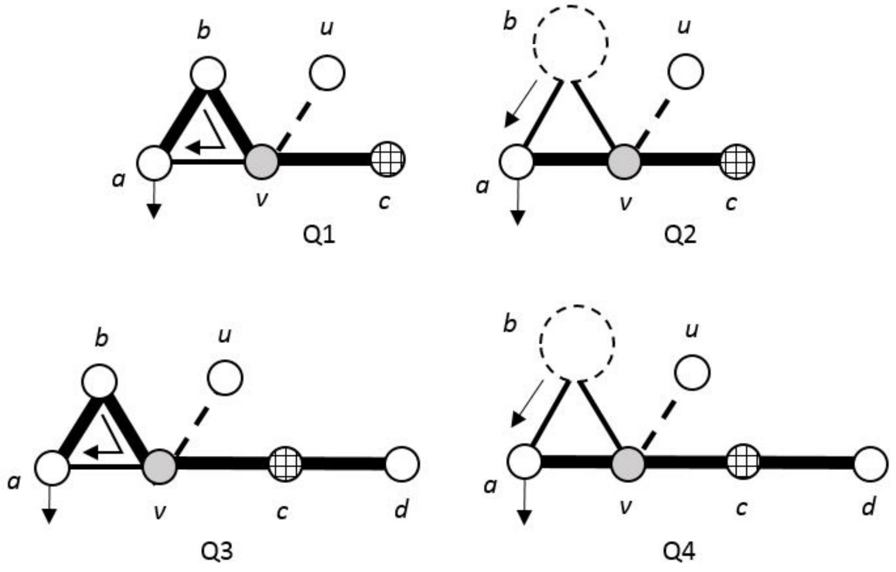


Figure 25: Growing into a grey node-1

Note: Consider the graphs B1,. . ., B7 in Figure 18. If we have one of these configurations with uv as the thin dashed edge, and all the nodes are shrunk, then the situation is dealt with elsewhere in the algorithm: If a and c are shrunk in B1, then ac is a temporary substructure with model M2 that should have been selected at the start of Step 2; similarly, if b and c are shrunk in B2. In the remaining cases, the thin dashed edge is a substruture with model M24. When it is considered as S_t in Step 2, part 1 in the Main Algorithm, the algorithm is directed to Step 2, part 3. The conditions in part 3 are not satisfied, so the algorithm continues to Step 2, part 4, where the triangle is shrunk in Subroutine 1.

End

6.4 Subroutines 3 and 4

In this section we describe Subroutines 3 and 4, which are used to create new substructures. Recall that edge uv was determined in Step 3 of the Main Algorithm.

Subroutine 3: Growth into a grey node

Consider graphs Q1 and Q2 in Figure 25: Triangle abv is a substructure with models M13 and M16, respectively, and vc is an end-edge for a substructure with

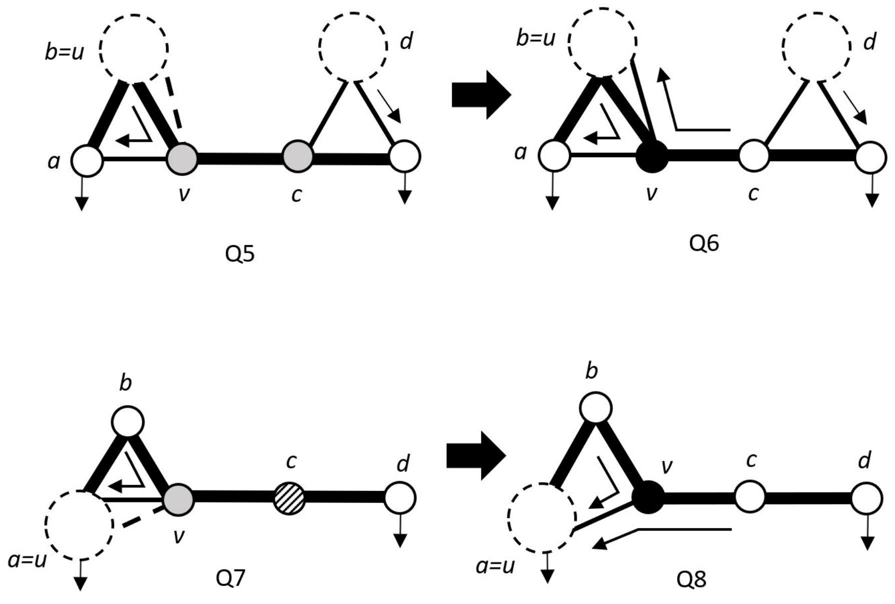


Figure 26: Growing into a grey node-2

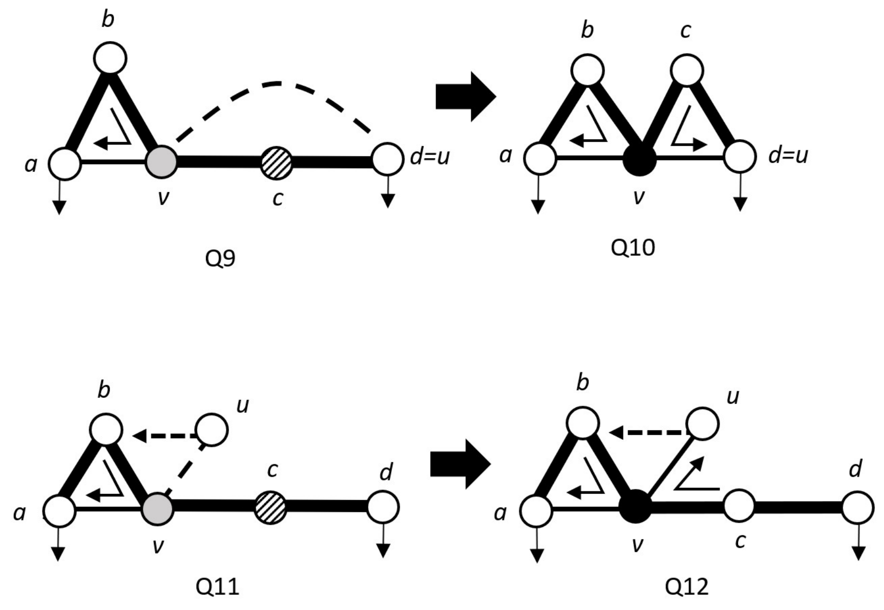


Figure 27: Growing into a grey node-3

model M21 or M22. Specifically, u is white or shrunk, v is grey, and c is striped or grey. (Note that c cannot be white since we do not consider edge uv in that case.) Graphs Q3 and Q4 represent special cases of Q1 and Q2, respectively. In particular, they have the same properties as Q1 and Q2, respectively, except the path substructures with endnode v have precisely two edges and a white node d as their other endnode.

- Suppose we have situation Q1 in Figure 25 where the node u is identified with node b (hence they are shrunk). Then make nodes a, b, v, and c into a substructure with model M18 where v becomes a black node and c becomes a white node. Let u' be the endnode of vu inside u. Then the alternating path for c is $cv \cup vu' \cup P^{u'}$.

- **Additional update:** Appropriately update the original substructure with model M21 or M22 that contains node c. In particular, if c is changed from a grey node to a white node, then the other substructure that contains it changes from a substructure with model M13 or M16 to a substructure with model M11 or M14, respectively.

  Example: In Figure 26, see the transformation of graph Q5 into Q6.

- Suppose we have situation Q2 in Figure 25, where the node u is identified with node b (hence they are shrunk). Then we remove edge bv in triangle abv from the substructures, make ba into a substructure with model M6, and make v a black node in a substructure S_j with model M9. Let u' be the endnode of vu inside u. Node c becomes a white node with alternating path $cv \cup vu' \cup P^{u'}$. Note that $\bar{P}^a$ cannot contain node c since it is just now being made white hence S_j satisfies property S13. Perform an additional update as described above, if needed. Note that the two resulting substructures do not form a prohibited combination since $u' \neq b'$, where b' is the endnode of ab that is inside b.

- Suppose we have situation Q1 in Figure 25, where the node u is identified with node a (hence they are shrunk). Then we remove edge av in triangle abv from the substructures, make ba into a substructure with model M5, and make v a black node in a substructure with model M9. Let u' be the endnode of vu inside u. Node c becomes a white node with alternating path $cv \cup vu' \cup P^{u'}$. Perform an additional update as described above, if needed.

  Example: In Figure 26, see the transformation of graph Q7 into Q8.

- Note that it is not possible that we have situation Q2 in Figure 25, where the node u is identified with node a because a is white (not shrunk).

- Suppose we have situation Q3 in Figure 25, where the node u is identified with node d (hence both are white (not shrunk)). Then we make v a black node, make c a white node, and make the nodes a, b, v, c, d define a substructure with model M19, where the alternating path for c is $cv \cup vd \cup P^d$.

  Example: In Figure 27, see the transformation of graph Q9 into Q10.

- Suppose we have situation Q4 in Figure 25, where the node u is identified with node d (hence both are white (not shrunk)). Then we keep v as a grey node, make c a white node, remove the edge bv from the substructures, make ba into a substructure with model M6, make av into a substructure with model M22, and make the triangle vcd into a substructure with model M13, where the alternating path for c is $cv \cup vd \cup P^d$.

- Suppose we have the situation Q1 in Figure 25, where u is not identified with a or b or a white node d as in Q3.

  - If the path $\bar{P}^u$ contains node b, then make a, b, v, c, u into a substructure with model M17, where v becomes black and c becomes white with alternating path: $cv \cup vu \cup P^u$. Perform an additional update as described above, if needed.

    Example: In Figure 27, see the transformation of graph Q11 into Q12.

  - If the path $\bar{P}^u$ does not contain node b, then remove edge av from the substructures, make v a black node and c a white node, make ab into a substructure with model M2, and make edges bv, uv, and cv into a substructure with model M8, where the alternating path from b becomes $bv \cup vu \cup P^u$ and the alternating path from c becomes $cv \cup vu \cup P^u$. Perform an additional update as described above, if needed.

- Suppose we have the situation Q2 in Figure 25, where u is not identified with b or a white node d as in Q4.

  - If the path $\bar{P}^u$ goes through node a without node b, then make a, b, v, c, u into a substructure with model M20, where v becomes black and c becomes white with alternating path: $cv \cup vu \cup P^u$. Perform an additional update as described above, if needed.
  - If the path $\bar{P}^u$ does not go through node a without node b, then disregard edge bv, make v a black node and c a white node, make ba into a substructure with model M6, make edges av, uv, and cv into a substructure S_j with model M9, where the alternating path from c

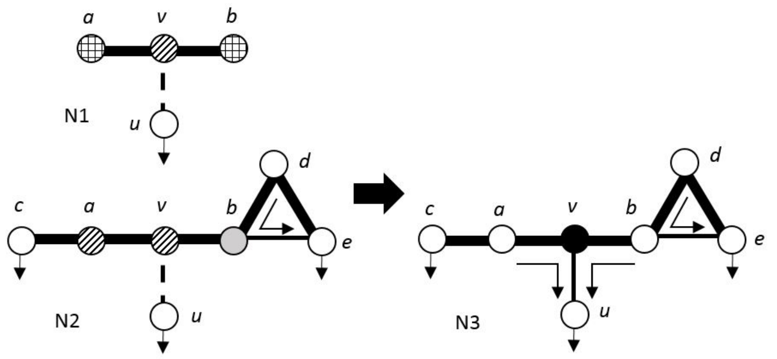


Figure 28: Growing into a striped node

becomes $cv \cup vu \cup P^u$. Note that $\bar{P}^a$ cannot contain node c since c is just now being made white, hence S_j satisfies property S13. Perform an additional update as described above, if needed.

End

Subroutine 4: Growth into a striped node

Consider graph N1 in Figure 28, where u is white or shrunk, v is striped, and a and b are each striped, grey, or white. Note that the path avb occurs in a substructure with model M4, M21, M22, or M23.

1. Make v black.
2. If a is white, make no change to its type. Similarly, if b is white.
3. If a is grey, make it white with alternating path: $av \cup vu \cup P^u$. Similarly, if b is grey.
4. If a is striped, make it white with alternating path: $av \cup vu \cup P^u$. Similarly, if b is striped.
5. Create a new substructure containing v with model M8, M9, or M10, as appropriate. If the substructure that contained avb contains additional edges, appropriately update them. Such an update produces one or two substructures, each with model M2, M22, or M23. If node a (respectively b) has been changed from a grey node to a white node, then the other substructure that contains it changes from a substructure with model M13 or M16 to a substructure with model M11 or M14, respectively.

   Example: Consider graph N2 in Figure 28, where c, a, v, b is a substructure with model M22 and b, d, e is a substructure with model M13. This step results in the substructures in graph N3, where c, a is a substructure

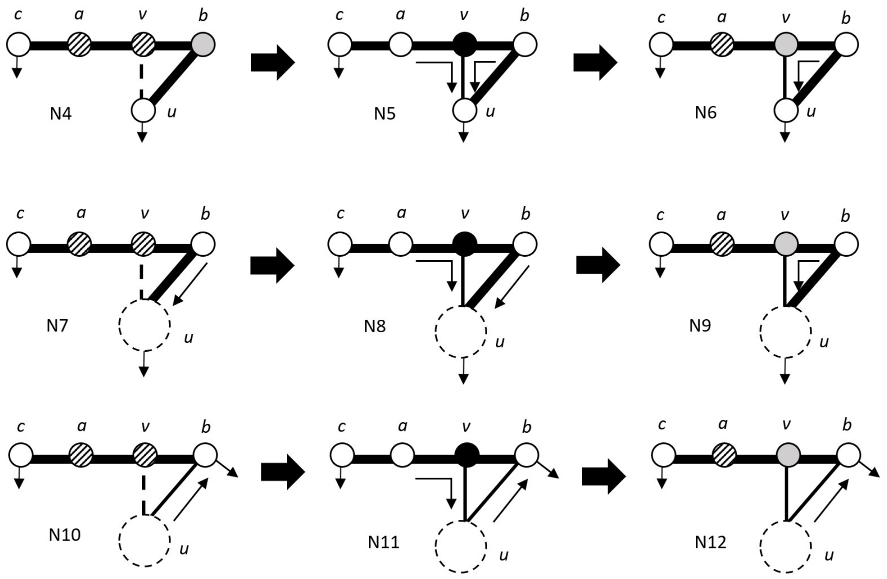


Figure 29: Growing into a striped node and creating a prohibited pair of substructures

with model M2, a, v, b, u is a substructure with model M8, and b, d, e is a substructure with model M11.

6. Check if the new black node substructure forms a prohibited pair of substructures as in Figure 10.

   - Suppose a prohibited pair of type Pr1 or Pr2 has been formed. Change v to a grey node, return the node a to its previous type, update the path substructure containing av, and identify the triangle as a substructure with model M13.
   - Suppose a prohibited pair of type Pr3 has been formed. Change v to a grey node, return the node a to its previous type, update the path substructure containing av, and identify the triangle as a substructure with model M16.

   Examples: Consider the graphs in Figure 29. Applying this step to N4 results in the graph N5, which contains the substructure a, v, b, u with model M8 and substructure b, u with model M2. These two substructures are a prohibited pair as shown in Pr1 in Figure 10. They are then modified as in N6, where v, b, u is a substructure with model M13. The graphs N7, N8, and N9 illustrate what happens when a prohibited pair as shown in Pr2 in Figure 10 is created. And the graphs N10, N11, and N12 illustrate what happens when a prohibited pair as shown in Pr3 in Figure 10 is created.

End

7 Proofs

In this section we prove the validity of the algorithm, show that it has polynomial time complexity, prove Theorem 1, and finish the proof of Theorem 2.

Let us first observe that the initial set of substructures, established in Step 0 of the Main Algorithm, satisfies the defining properties S1-S13 of substructures. Our first objective is to show that the various subroutines always result in a new set of substructures that continue to satisfy these properties. For the most part, this is straightforward. Most of our effort goes into showing this is true for property S10 during the shrinking procedure in Subroutine 1, which is the most complex part of the algorithm.

For each call of Subroutine 1, we have a corresponding node set C. Let $A(C)$ denote the arcs of D between two nodes of C. (These are the arcs of D that correspond to the cycle created by adding the edge uv to the graph underlying D.) Note that, if there exists a substructure S_i with model M8 where a, b, and d are in $C(S_i)$, then we first process S_i and then remove d from all sets $C(S_j)$ that contain d. As a result, d is not involved in any substructures subsequently considered during this call. However, $A(C)$ still contains two arcs of D incident into d. We call such a node d an *M8 bottom node.* This situation leads to the creation of a type 1 shrunk node.

Observe that, for a given S_i during a call of Subroutine 1, the set $Int(S_i)$ is recomputed each time S_i is considered in Subroutine 1*. The following result shows that each time the set is recomputed, it contains the previously computed set.

Proposition 3. *Consider a call of Subroutine 1 from the Main Algorithm and suppose S_i is considered two or more times during this call. In one case, suppose the input to Subroutine 1^* is $C(S_i) \cup Ext_1(S_i)$ and the chosen output is $Int_1(S_i)$. In another case, suppose $C(S_i) \cup Ext_2(S_i)$ is the input and the chosen output is $Int_2(S_i)$. If the $Ext_1(S_i)$ case occurs before the $Ext_2(S_i)$ case, then $Ext_1(S_i) \subset Ext_2(S_i)$ and $Int_1(S_i) \subseteq Int_2(S_i)$.*

Proof. We assume that S_i is selected at least twice in Step 1 of Subroutine 1. After its first selection, after finishing Step 1, part 1, all nodes in $C(S_i) \cup Ext(S_i)$ at that point have been added to $Proc(S_i)$ and removed from $Unproc(S_i)$. Observe that no nodes are removed from $Proc(S_i)$ or from $Ext(S_i)$ at any later point in the subroutine. Hence, when S_i is selected a second time, there must exist a new node in $Ext(S_i) \cap Unproc(S_i)$. This continues to be true for any subsequent selections of S_i, hence, in general, $Ext_1(S_i) \subset Ext_2(S_i)$. An examination of the cases in Subroutine 1^* then shows that we have $Int_1(S_i) \subseteq Int_2(S_i)$.

□

Consider a call of Subroutine 1 at the point just before Step 2 of the subroutine is executed (where we perform a shrinking). We define a special subdigraph, call it D', of D as follows: To begin, set D' to be the tree defined by the arcs in $A(C)$ and the endnodes of those arcs. Next, for each substructure S_i, add to D' all arcs of D (with their endnodes) where the endnodes are an in-node, out-node pair for S_i and either one endnode is in $C(S_i) \cup Ext(S_i)$ and the other endnode is in $Int(S_i)$ or both endnodes are in $Int(S_i)$ (where the M8 bottom node, if one exists, has been removed from each $C(S_i)$).

Proposition 4. *Consider a call to Subroutine 1 at the point just before Step 2 of the subroutine is executed. Then the node set of D' (defined above) equals the union of the white and shrunk nodes in the sets $C(S_i) \cup Ext(S_i) \cup Int(S_i)$, taken over all substructures S_i, plus an M8 bottom node, if one exists. In addition, D' is connected.*

Proof. Consider the following construction during the entire execution of Subroutine 1 up to the point just before Step 2 or the subroutine is executed. To begin, set $E' := A(C)$. Whenever a substructure S_i is considered and a white or shrunk node x is identified for $Int(S_i)$, observe, in Subroutine 1*, that there is at least one arc of D between x and a white or shrunk node y in $C(S_i) \cup Ext(S_i)$; or there is an arc of D between x and another node y in $Int(S_i)$ such that, in both cases, x, y is an in-node, out-node pair for S_i. (Such arcs can go in either direction between x and y.) Add all such arcs to E'. Note that by Proposition 3, this set of arcs, taken over all S_i, equals the arc set of D' (which is defined at the end of this procedure). Hence, the node set of D' equals the union of the white and shrunk nodes in the sets $C(S_i) \cup Ext(S_i) \cup Int(S_i)$, taken over all substructures S_i, plus an M8 bottom node, if one exists. Furthermore, by inspecting the cases in Subroutine 1*, we can see by induction on the number of arcs in E', where we start from $E' := A(C)$, that the arcs in E' (with their endnodes) form a connected digraph at all times (since the arcs in $A(C)$ (with their endnodes) form a connected digraph); hence, D' is connected.

□

Proposition 5. *Let G be a graph with associated graph $\tilde{G}$ and set of substructures that satisfy the properties S1-S13 of substructures. Then the substructures resulting from a call of Subroutine 1, 2, 3, or 4 also satisfy these properties.*

Proof. Let us discuss a few cases in a bit of detail. In all other cases, a straightforward examination of the algorithm shows that all the properties are satisfied at the ends of the various subroutine calls. Note that whenever a substructure with model M9 is formed, we establish that Property S13 holds. An examination of the cases when a substructure with model M8 or M9 is formed shows that the only place a prohibited substructure could be formed is in Subroutine 4. We deal with this possibility in the subroutine by forming substructures with model M13 or M16. Thus, Property S12 holds. Let us finally address Property S10 at the end of a call to Subroutine 1, which requires the newly defined directed graph, call it D'', to be a directed forest. Let us consider how D'' can be formed from its predecessor D. Let D' be the sub-digraph of D defined above. By Proposition 4, the node set of D' consists of the white and shrunk nodes in $C(S_i) \cup Ext(S_i) \cup Int(S_i)$, taken over all substructures S_i, plus an M8 bottom node, if one exists; and D' is connected. Consider the collection of substructures obtained by shrinking the nodes in $C(S_i) \cup Ext(S_i) \cup Int(S_i)$, over all the substructures S_i. If there is no M8 bottom node, then D'' is obtained from D by shrinking the nodes in D' (and adding one arc for each node that is switched from grey or striped to white; these arcs have degree 1 at the switched nodes). Since D' is connected, D'' is a directed forest, as required. Suppose there is an M8 bottom node. Then shrinking the nodes of $C(S_i) \cup Ext(S_i) \cup Int(S_i)$ yields a directed graph that is a forest except for one pair of parallel arcs directed into the M8 bottom node. D'' is obtained by deleting one of these two parallel arcs (and adding one arc for each node that is switched from grey or striped to white; these arcs have degree 1 at the switched nodes), and hence it is a directed forest. Thus, D'' satisfies property S10.

□

The following proposition simply notes that Properties A1, A2, and A3 (from Section 5.4) hold. These properties will help us show that the tri-blossom clusters at the end of the algorithm (defined below) are saturated by the tri-free simple 2-matching at the end of the algorithm.

Proposition 6. *Properties A1, A2, and A3 hold during the algorithm.*

Proof. These properties hold by induction on the number of shrunk nodes during the algorithm and an examination of Subroutine 1.

□

The following proposition shows that Property A4 holds; it concerns the five types of shrunk nodes that occur during the algorithm.

Proposition 7. *Property A4 holds during the algorithm.*

Proof. Let v be a shrunk node in $\tilde{G}$ during the algorithm. Let v' denote the base node of v and let $v'x$ be the base edge of v, if it has one. If v has a base edge in M, or has no base edge, then it follows from Properties S1 - S13 and Properties A1 and A2 that v has one of types 2 - 5. If v has a base edge not in M, then it follows that v has type 1 by the way such shrunk nodes are constructed: either in Step 2 of the Main Algorithm, where the base edge arises from the edge cd in a substructure with model M8; or, at a later point, when v, but not x, shrinks into a larger shrunk node (e.g., in Subroutine 1* when S_i has model M6 and a is the only node in $C(S_i) \cup Ext(S_i)$).

□

The next proposition helps demonstrate in the subsequent proposition that Property A5 holds; it concerns the three types of alternating paths, which are defined in Subroutine 1*.

Proposition 8. *Consider a substructure S_i during a call to Subroutine 1. Suppose $x, y \in C(S_i) \cup Ext(S_i)$ and x, y is an in-node, out-node pair in S_i (hence xy (or yx) is an arc in D). Then xy (or yx) is an arc in $A(C)$.*

Proof. Recall, from Proposition 3, that if a node x is contained in $C(S_i)$ or $Ext(S_i)$ or $Int(S_i)$ at any time during an execution of Step 1 of Subroutine 1, then x remains in that set up to the point just before Step 2 of the subroutine is executed. So, let us suppose an execution of Subroutine 1 is at the point just before Step 2 of the subroutine is executed. Consider two nodes x, y that are an in-node, out-node pair for a substructure S_i (hence, each node is either white or shrunk). If $x, y \in C(S_i)$, then xy (or yx) is in D', by definition of D'. If $x \in C(S_i) \cup Ext(S_i)$ and $y \in Int(S_i)$, then, again by definition, xy (or yx) is in D', and this arc cannot be in $C(S_i)$ because y is not in $C(S_i)$. Similarly, if $x, y \in Int(S_i)$, then xy (or yx) is in D', and this arc cannot be in $C(S_i)$ because neither x nor y is in $C(S_i)$. By definition of D', these three types of arcs, taken over all S_j, comprise all the arcs of D'. There are two other possibilities for x and y; we could have $x, y \in Ext(S_i)$; or $x \in C(S_i)$ and $y \in Ext(S_i)$. Note that, in either case, the corresponding arc xy (or yx) would not be in D', from the definition of D', but it would be in D, since nodes x, y are an in-node, out-node pair for S_i. However, this arc would form a cycle with the arcs in D' since, by Proposition 4, the node set of D' equals the union of the white and shrunk nodes in the sets $C(S_j) \cup Ext(S_j) \cup Int(S_j)$, taken over all substructures S_j, plus an M8 bottom node, if one exists; and D' is connected. This contradicts D being acyclic. The result follows.

□

Proposition 9. *Property A5 holds during the algorithm.*

Proof. First we show that the paths defined in Subroutine 1* are well-defined and alternating. Next, we show these alternating paths pass through the base node (and base edge, if it exists) of the new shrunk node. Using induction, let us assume $\tilde{G}$ has $n \geq 0$ shrunk nodes and the result holds for graphs with at most n shrunk nodes. Consider a new shrunk node as it is being defined during a call to Subroutine 1*. Note that each new alternating path defined is constructed by combining explicitly defined paths with previously defined paths. Previously defined paths have the form P^{uv} and P^x. These are alternating paths by induction. Explicitly defined paths have the form of a few edges of $\tilde{G}$ and paths of the form P^u_x. (For example, consider the case that S_i has model M13 in Subroutine 1* and a and b are the only nodes in $C(S_i) \cup Ext(S_i)$. For pendant edge dc, we define the alternating path $dc \cup ca \cup ab \cup P^u_b \cup P^{uv} \cup P^v$. The explicitly defined portion of the path is $dc \cup ca \cup ab \cup P^u_b$; the remainder uses previously defined paths.) The initial few edges form an alternating path by inspection of the cases. Proposition 8 shows that the addition of P^u_x to these initial few edges defines an alternating path since the edge adjacent to P^u_x in the path has its endnodes in $C(S_i) \cup Ext(S_i)$, which are an in-node, out-pair in S_i, hence they also define an arc in $A(C)$. (In our example, nodes a and b are in

$C(S_i) \cup Ext(S_i)$, are an in-node out-node pair for S_i, and therefore define an arc in $A(C)$; hence the path P_b^u plus ab is a well-defined alternating path.)

To see that every alternating path defined in Subroutine 1* passes through the base node of the new shrunk node, observe that, in all but two cases, the alternating paths end with a subpath of the form P^x, where x is being shrunk into the new shrunk node and x is either a node of $\tilde{G}$ or a node of G inside a shrunk node of $\tilde{G}$. The two exceptions, where the alternating paths end with a subpath that does not go to a root of $\tilde{G}$, are the following:

1. S_i has model M11, c is the only node in $C(S_i) \cup Ext(S_i)$, and the path in D from c goes to b.

2. S_i has model M17, d and e are the only nodes in $C(S_i) \cup Ext(S_i)$, and the path in D from e goes to b.

In both of these cases, the alternating paths end at node b (in model M11 or M17), if it is white, or the base of b, if it is shrunk. Hence, the final subpath in these cases has the form P_b^x.

For each path P^x (or P_b^x), there is a corresponding path $\bar{P}^u$ (or $\bar{P}_v^u$) in D. Because the nodes being shrunk form a connected subtree D' of D (by Proposition 4), each path ends at a common root (or passes through the node b as discussed in the above two exceptions). Thus, there is a first node y common to these paths. Again, because D' is connected, the node y will shrink into the new shrunk node. If y is not a root, let yz denote the first arc in $\bar{P}^y$. This arc yz in D defines a path through a substructure after the shrinking and the first edge on this path becomes (by definition) the base edge of the new shrunk node and its endnode inside y is the new base node (by induction if y is shrunk). Furthermore, all the alternating paths pass through this new base node and base edge (by induction if y is shrunk). If y is a root, then it becomes the new base node if it is white, or, if it is shrunk, its base node becomes the new base node (again by induction).

□

We next present the proof of Proposition 2.

Proof. Suppose x in F is not incident with a dashed thick edge in the corresponding figure and x' is incident with exactly one edge of M in S_i. Furthermore, suppose x' is a white node in S_i. (E.g., x could be node b in model M2 or M9). If x' is a root, then it is incident with no other edge of M, and point 1 holds for this case. If x' is not a root, then it is an in-node for some other substructure, say S_j (by Property S9). Note that (as observed after Property S4), for any such S_j, the path from the white in-node x' to an out-node starts with an edge, say e, of M. Hence, the edge e is not a border edge by the definition and, again, point 1 holds for this case.

Again, suppose x in F is not incident with a dashed thick edge in the corresponding figure and x' is incident with exactly one edge of M in S_i. Now, suppose x' is a shrunk node in S_i and that e is the edge of M in S_i incident with

x'. Let x'' be the endnode of edge e inside x'. If e is the only edge of M incident with x'' in G, then point 1 follows for this case since there is no edge of M with one endnode at x'' and the other not in S_i, (hence, by Property A4, x' is a type 4 shrunk node with its base node at x''). So let us suppose x'' is incident with a second edge, say f, of M in G. If f is inside x', then, again, point 1 follows for this case. So let us suppose that f is outside x'. By Properties A3 and A4, x' is a type 2 shrunk node, f is the base edge (since for all models, the arrow in the figures at x' points away from S_i), and x' is not a root. Hence, as for the previous case, x' must be an in-node in some other substructure and, again, point 1 holds for this case.

Clearly, if x' is incident with two edges of M in S_i (e.g., x is node c in model M8 or node b in model M11), then x in F is incident with no dashed thick edge, S_i cannot have a border edge at x', and point 1 holds for this case.

Suppose x in F is not incident with a dashed thick edge in the corresponding figure and x' is incident with no edges of M in F. If S_i has model M1, point 1 immediately holds for this case by Property S6. In all other cases, x' is a type 1 shrunk node and an in-node for S_i; e.g., x could be node a in M6 or node a in M7. By Property A4, the base node of x' is incident with two edges of M that are inside x', hence no border edge for S_i can be incident with x' and we have finished showing point 1 holds.

There is one final case to show point 1 holds: Suppose x in F is not incident with a dashed thick edge in the corresponding figure and x' is incident with exactly one edge of M in S_i. Furthermore, suppose x' is a shrunk node in S_i and that it is incident with an edge e of S_i *not* in M. These conditions are satisfied only if S_i has model M3, where $a = x$. Observe from Subroutine 1* that a substructure with model M3 is formed in one of two ways: by shrinking nodes a and b in a substructure with model M11 or M14. Call this substructure S_j. Let x'' be the endnode of edge e in S_j. Consider the pass through Subroutine 1 when S_j is shrunk at the end of the subroutine. Let us apply the above arguments as follows: If S_j has model M11, then node a in S_j plays the role of x''; and if S_j has model M14, then node b plays the role of x''. We see that S_j can have no border edge at x'' in either of these cases. And this property continues to hold when a and b shrink together and the resulting substructure ultimately becomes S_i.

For the remaining cases, assume that x in F is incident with a dashed thick edge in the corresponding figure. Point 2 immediately follows from Property S8. For point 3, observe that $\tilde{G}$ can contain a substructure with model M2, say with nodes x' and y, where y is not in S_i. The edge $x'y$ is a border edge for S_i, hence point 3 holds. (There are numerous other ways S_i could have a border edge incident with x'.)

□

We next consider the trickiest part of the proof of the algorithm's validity: showing that performing exchanges along alternating paths to increase the size of the simple 2-matching creates no triangles in M.

Theorem 4. *Performing an exchange along the alternating path P in Step 2, part 3 of the Main Algorithm (where any required extensions of P have been performed) results in a tri-free simple 2-matching.*

Proof. For the following discussion, let M denote the tri-free simple 2-matching before an exchange on P and let M' denote the simple 2-matching after the exchange on P. We show that M' is tri-free. Consider the edges of $\tilde{G}$ that occur in no substructure and the edges inside a shrunk node that were in no substructure when the node shrunk. These edges are not in M (by property S3), do not occur in P, and therefore cannot be in M'. Hence, we focus on the remaining edges of G. The proof is divided into three stages. In the first stage we consider each substructure and identify a number of situations where an edge cannot occur in a triangle of M'. The second stage builds on this by showing that no triangle with all its edges in $\tilde{G}$ can occur in M'. The third stage shows that no triangle with one or more edges inside a shrunk node can occur in a triangle of M'.

Definition: During this pass through Step 2 of the Main Algorithm, we refer to the temporary substructure with endnodes u and v as the *uv-substructure.*

Stage 1

In this stage, we consider each type of substructure and identify situations where no edges in the substructure can occur in a triangle of M'. We also consider one case (for substructures with model M3) where an edge inside a shrunk node cannot occur in a triangle of M'.

Let S_i be a substructure in $\tilde{G}$ before the exchange:

Suppose S_i has model M2: Clearly, if P contains ab, then it cannot occur in a triangle of M'. This occurs only if S_i is a temporary substructure. We examine the case that P does not contain ab in Stage 2.

Suppose S_i has model M3: Note that there are two ways S_i could have first arisen: by shrinking node a (or shrinking a node inside a) when considering a substructure with model M11 or M14. Let T denote the triangle in that substructure with model M11 or M14. Observe that P (including possible extensions) can contain an edge of S_i in several ways. In one case, S_i is the uv-substructure, it arose from a substructure with model M11, and P contains the edge of S_i that is not in M and the edge of T that is shrunk. In a second case, S_i is not the uv-substructure, it arose from a substructure with model M11, and P contains both edges of S_i, but not the edge of T that is shrunk. In a third case, S_i is the uv-substructure, it arose from a substructure with model M14, and P contains the edge of S_i that is in M and no other edge of T. In a fourth case, S_i is not the uv-substructure, S_i arose from a substructure with model M14, and P contains only the shrunk edge of T. In a fifth case, S_i arose from a substructure with model M14, P contains no edge of T but contains an edge adjacent in G to edge ab and we performed an extension of P on the substructure with model M14 in Step 2 of the Main Algorithm. In each of these cases, M' contains either two edges of T or no edges of T, hence, none of these three edges (hence, neither edge of S_i) can be in a triangle in M'; this includes all triangles, besides T, that contain an edge of T. A sixth case is that S_i arose from a substructure with model M14 and no edge of P is incident in G to a node of T. In this last case, M' contains one edge of T, but this edge cannot be in a triangle of M' because this edge was not in a triangle of M.

Suppose S_i has model M4: S_i is a cycle of edges in M with length at least 4, hence, no edge of S_i can occur in P and no edge of S_i can occur in a triangle of M'.

Suppose S_i has model M5: Clearly, if P contains ab, then it cannot occur in a triangle of M'. We examine the case that P does not contain ab in Stage 2.

Suppose S_i has model M6: Clearly, if P does not contain ab, then it cannot occur in a triangle of M'. We examine the case that P contains ab in Stage 2.

Suppose S_i has model M7: If P contains an edge of S_i, then M' must contain two edges in S_i. It follows that no edge of S_i can occur in a triangle of M'.

Suppose S_i has model M8, M9, or M10: We examine these cases in Stage 2.

Suppose S_i has model M11: If S_i is the uv-substructure, then no edge of S_i is in M'. Otherwise, for all possible paths P (whether they contain an edge of S_i or not), S_i contains two edges of M'. Hence, no edge of S_i can occur in a triangle of M'.

Suppose S_i has model M12: If P passes along the path $\{c, b, a\}$, then S_i contains no edges in M'. Otherwise, S_i contains two edges of M'. Hence, no edge of S_i can occur in a triangle of M'.

Suppose S_i has model M13: P either goes along the path $\{b, c, a\}$ or contains no edge of S_i. In either case S_i contains two edges of M'. Hence, no edge of S_i can occur in a triangle of M'.

Suppose S_i has model M14 or M15: P either contains the edge ab only; the edge ac only (when it is the uv-substructure for the M14 case); all three edges of the triangle (where an extension of P was performed); or no node of the triangle. Hence, the triangle has either two or no edges in M', hence, no edge of triangle can be in a triangle of M'; or the triangle contains one edge in M', but this edge cannot be in a triangle of M' since it was in no triangle of M and P contains no node of the triangle (due to the condition for an extension of P to be performed).

Suppose S_i has model M16: P either contains the edge ab only; all three edges of the triangle (where an extension of P was performed); or no node of the triangle. Hence, the triangle has two edges in M' and no edge of the triangle can be in a triangle of M'; or the triangle contains one edge in M', but this edge cannot be in a triangle of M' since it was in no triangle of M and P contains no node of the triangle (due to the condition for an extension of P to be performed).

Suppose S_i has model M17, M18, or M19: An alternating path passing through these substructures is handled similarly to case M13. Note that the edges not in the triangles of S_i with model M17 or M18 also cannot occur in a triangle of M': In all cases, where P passes through S_i or not, one of these edges is in M' and this edge is part of a path with 3 edges in M'.

Suppose S_i has model M20: P contains, from S_i, either (1) the edge ab and no other nodes of S_i; (2) the edges along the path d, c, e or the path $d, c, e, ..., a$, which led to an extension of P that contains all three edges of the triangle; (3) the nodes e and a only, or the node a only, which, in both cases, led to an extension of P; or (4) no node of S_i. In case (1), M' contains the path b, a, c, d. Since this path has three edges in M', no edge of S_i can be in a triangle of M'. In case (2), M' contains the path a, b, c, e. Since this path has three edges in M', no edge of S_i can be in a triangle of M'. In case (3), M' contains the path a, b, c, d. Since this path has three edges in M', no edge of S_i can be in a triangle of M'. In case (4), M' contains the path a, c, d. Since no node of this path is incident in G with an edge of P, and M was tri-free, no edge of S_i can be in a triangle of M'.

Suppose S_i has model M21 or M22: The path P contains no edge in S_i. Let x be a grey endnode of S_i. Then x is also in a substructure S_j with model M13 or M16. We have seen that the edges in S_j cannot occur in a triangle of M' and we also see, from our examination of the way P can pass through S_j, that M' has degree 1 at x in S_j. It follows that no edge of S_i can occur in a triangle of M'.

Suppose S_i has model M23: (Recall that M23 contains at least two edges.) No edge of S_i can be in P, hence, if S_i contains three or more edges, none of them can be in a triangle of M'. The case where S_i has two edges is examined in Stage 2.

Suppose S_i has model M24: If S_i is not the uv-substructure, then it cannot be in M'. The case where S_i is the uv-substructure is examined in Stage 2.

Stage 2

Here is a list the substructures of $\tilde{G}$, and situations with respect to P, where it remains to be shown that an edge from such a substructure cannot occur in a triangle of M'.

- S_i has model M2 (where ab is not in P).
- S_i has model M5 (where ab is not in P)
- S_i has model M6 (where ab is in P)
- S_i has model M8, M9, or M10.
- S_i has model M23 and contains precisely two edges in M.
- S_i has model M24 and is the uv-substructure.

In this stage we show that no triangle with all its edges in $\tilde{G}$ can occur in M'. Given our work in Stage 1, it suffices to consider triangles consisting of edges from substructures as listed above.

Consider Figure 18. Disregarding the dashed edges, graphs B1, B2, ..., B7 illustrate all the ways two substructures with models M2, M5, and M6 can share one node in $\tilde{G}$. The white nodes in each case can also be shrunk. We observe, in each of these cases, the only way to form a triangle in M' in $\tilde{G}$ containing both these edges and using a substructure from the above list is to use a substructure with model M24, where this is the *uv*-substructure (otherwise it could not enter M'). This possibility is illustrated by the dashed edges in graphs B1, ..., B7 in Figure 18; in each case this edge forms a triangle of edges in G. In Subroutine 2 we described transformations in the cases where at least one of the nodes in the triangle is white (not shrunk). In these cases the triangles are transformed into larger substructures, so there is no alternating path P to consider. So, let us assume in each case that all three nodes are shrunk. First, observe that this implies for B1 and B2 that *ac* and *bc*, respectively, are temporary substructures. Hence, they would already have been considered in Step 2 of the Main Algorithm, due to the selection criteria for S_t, so these cases are not possible. Consider graphs B3, .., B7: By considering the corresponding cases in the algorithm for each of the substructures, we see that these graphs would shrink, so again there is no alternating path P to consider.

Next, observe that a substructure S_i with model M23 can only form a triangle in M' with a *uv*-substructure with model M24 (see B8 in Figure 18). This possibility is addressed in Subroutine 2 where these two substructures are transformed into a single substructure with model M11. So there is no alternating path P to consider in this case.

We next consider how a triangle in M' in $\tilde{G}$ might be formed using a substructure with model M8, M9, or M10.

First, consider the ways a substructure with model M8, M9, or M10 can form a triangle with a substructure with model M2, M5, or M6. Three possibilities are illustrated in Figure 10. These three combinations are prohibited by Property S12. They can only be formed when the black nodes are first constructed in Subroutine 4, but when this occurs the combinations are transformed into single substructures with grey nodes. No other combination of these substructures results in a triangle in M' when we perform an exchange as indicated by the arrows.

We next consider the ways a substructure with model M8, M9, or M10 can form a triangle T with a substructure *uv* with model M24. By examining Figure 7 and the alternating paths indicated by the arrows, we see that there are two ways in which such a triangle could appear in M. First, suppose we have a substructure with model M9 (on nodes a, b, c, d as in Figure 7) and we consider a temporary substructure $uv = bd$ with model M24 in Subroutine 1*. In the course of defining the shrinking, gd is a pendant edge for bd and is assigned the alternating path $gd \cup db \cup P^b$. If P^b passes through node a, then performing an exchange on the path would create a triangle bcd in M. However, by substructure property S13, this cannot happen (and this is enforced when substructures with model M9 are created in Subroutine 4). A second way a triangle could appear in M is if we have a substructure with model M10 and we consider a temporary substructure $uv = ab$ with model M24. Performing an exchange on any alternating path through ab would create the triangle abc in M. However, in Subroutine 2 we consider this situation (see B8) and instead form a new substructure with model M11.

Finally, observe that no triangle with two or three edges in a substructure with model M24 can be in a triangle of M' since at most one of those edges can be in M' (i.e., the uv-substructure).

Stage 3

To this point we have presented a number of situations where an edge of $\tilde{G}$ cannot occur in a triangle of M'. In particular, we have shown that no triangle with all three edges in $\tilde{G}$ can occur in M'. So, the theorem follows if $\tilde{G}$ contains no shrunk nodes. For the remainder of the proof, we assume $\tilde{G}$ has at least one shrunk node and P contains at least one edge inside a shrunk node.

We next consider the ways M' could contain a triangle with one or three edges inside a shrunk node of $\tilde{G}$. (Precisely two edges inside a shrunk node is not possible.) To begin, note that it is not possible for an edge in a substructure in $\tilde{G}$ with model M8, M9, or M10 to be in a triangle with a shrunk edge since each such edge is incident with a black node that must be non-shrunk in $\tilde{G}$ and the three adjacent nodes (represented by a, b, and d in Figure 7) are distinct in $\tilde{G}$. Similarly, consider a substructure with model M23 that contains exactly two edges. Each node is non-shrunk, hence these edges cannot be in a triangle with a shrunk edge.

The following claim addresses this final issue. We remark, after the statement of the claim and before its proof, that establishing its validity concludes the proof of the theorem.

Claim 2. *Let w be a shrunk node in $\tilde{G}$ during the algorithm. Let* boundary(w) *be the edges incident with w in $\tilde{G}$ that are contained in substructures that have models M2, M3 (where node a plays the role of w), M5, M6, or M24. Let W denote the subgraph of G induced by the edges inside w together with the edges in boundary(w). Let $P(W)$ denote the restriction of alternating path P to W. Then performing an exchange on $P(W)$ results in a tri-free simple 2-matching in W.*

Before proving Claim 2 let us make two observations that show this claim concludes the proof of the theorem.

First, note that there may be edges of $\tilde{G}$ incident with w that are not included in W. For example, consider a substructure with model M11 where a plays the role of w. The edges ab and ac are not included in W. However, we have already argued that edges of this type cannot be in a triangle of M'.

Second, note that this claim allows us to conclude that using an extension of P creates no triangles in M'. To see this, choose a substructure S_i with model M14, M15, M16, or M20 and let b play the role of w. Note that performing an exchange on an extension of P is simply an exchange on a type 1 alternating path starting from the base node of b in S_i, and ending along edge ba, plus an exchange on the path a, c, b. We have already shown that no triangle in M' can contain an edge of the triangle abc and Claim 2 shows that no triangle in M' is created in W.

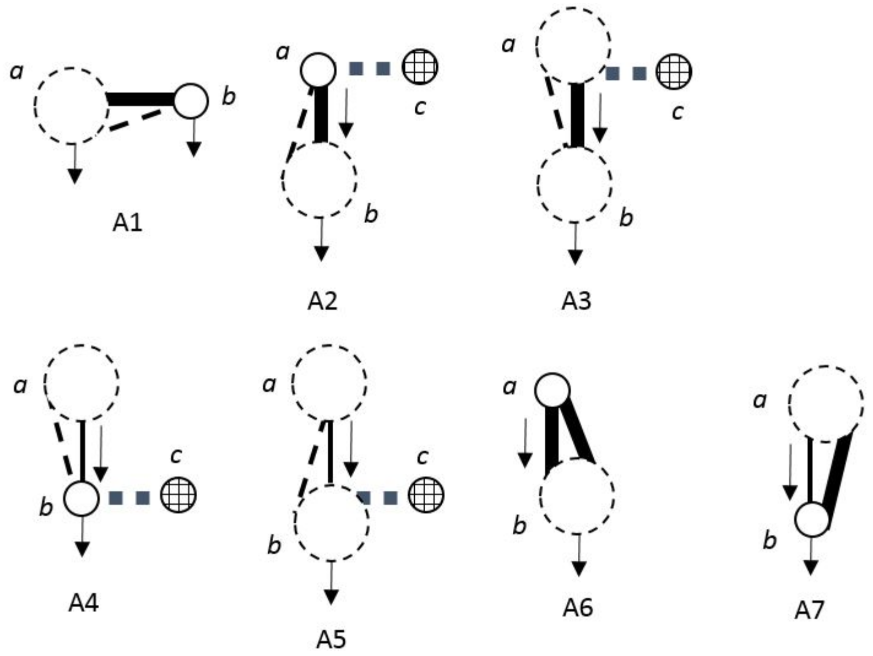


Figure 30: Triangles with two single-edge substructures and one edge inside a shrunk node of $\tilde{G}$. (Thin dashed edges represent substructures with model M24.)

We next prove Claim 2.

Proof. Let us assume $P(W)$ contains at least one edge of W; and let M'' denote the simple 2-matching on W obtained by performing an exchange on $P(W)$.

The proof has three stages. In Stage $1'$, we show that no triangle with two edges in *boundary*(w) and one edge inside w can occur in M''. Let $\bar{W}$ denote the graph obtained by undoing the final shrinking that yielded w and adding to this the edges in *boundary*(w). (Thus $\bar{W}$ may contain shrunk nodes.) In Stage $2'$, we show that no triangle with all its edges in $\bar{W}$ and all its edges inside w can occur in M''. In Stage $3'$, we show that no triangle with at least one edge shrunk in $\bar{W}$ and all its edges inside w can occur in M''.

Stage $1'$

Let us first consider the ways we can have a triangle T with two edges in *boundary*(w) and one edge inside w. We first consider the possibilities resulting from the pairs of substructures illustrated in Figure 30. In cases A1 to A5, the thin dashed edge represents a substructure with model M24. The remaining edges in the figure have substructures M2, M5, or M6. (Note that we have shown in Stage 1 above that no edge in a triangle formed by the edges in a substructure with model M3 can be in a triangle of a simple 2-matching that results from an exchange on P, so we need not consider any triangles T involving an edge in such a substructure.) Cases A1 to A7 illustrate all the ways substructures with models M2, M5, M6, or M24 can form a triangle with one edge inside w, with one exception: two substructures with model M24. We address cases A1 - A7 and then the exception.

Suppose we have the situation in graph A1 in Figure 30 and the node a plays the role of w. Let a' and a'' denote the endnodes inside a of the substructures with models M24 and M2, respectively. Let us assume there is an edge $a'a''$ inside a. In order for $a'b$ to be in M'', we have that $a'b$ is the uv-substructure. Hence, we have that $P(W)$ follows a type 3 alternating path through a that starts with ba'. Note that ba' plays the role of a "cross edge" in all corresponding cases in Subroutine 1^* in the process of shrinking a. In all such cases, the specified alternating path results in a triangle T containing two edges of M''. Hence no edge of T can occur in a triangle of M'' due to a type 3 alternating path starting with $a'b$.

Suppose we have the situation in graph A2 in Figure 30 and the node b plays the role of w. Let b' and b'' denote the endnodes inside b of the edges corresponding to the substructures with models M24 and M5, respectively. Let us assume there is an edge $b'b''$ inside b. In order for ab' to be in M'', we have that ab' is the uv-substructure. Hence, we have that $P(W)$ follows a type 3 alternating path through b that starts with ab'. Note that ab' plays the role of a "cross edge" in all corresponding cases in Subroutine 1^* in the process of shrinking b. In all such cases, the specified alternating path results in a triangle T containing two edges of M''. Hence no edge of T can occur in a triangle in M'' due to a type 3 alternating path starting with ab'.

Suppose we have the situation in graph A3 in Figure 30 and node a plays the role of w. Let a' and a'' denote the endnodes inside a of the substructures with models M24 and M5, respectively. An alternating path of type 3 for $a'b$ is not defined nor used in the algorithm. No alternating path of type 1 or 2 contains $a'b$, hence M'' cannot contain T in those cases. The algorithm simply shrinks nodes a and b together.

Suppose we have the situation in graph A4 in Figure 30 and the node a plays the role of w. Let a' and a'' denote the endnodes inside a of the edges corresponding to the substructures with models M24 and M6, respectively. Let us assume there is an edge $a'a''$ inside a. In order for $a'b$ to be in M'', we have that $a'b$ is the uv-substructure. Hence, we have that $P(W)$ follows a type 3 alternating path through a that starts with $a'b$. Note that $a'b$ plays the role of a "cross edge" in all corresponding cases in Subroutine 1^* in the process of shrinking a. In all such cases, the specified alternating path results in the triangle T containing two edges of M''. Hence no edge of T can occur in a triangle in M'' due to a type 3 alternating path starting with $a'b$.

Suppose we have the situation in graph A5 in Figure 30 and the node b plays the role of w. Let b' and b'' denote the endnodes inside b of the substructures with models M24 and M6, respectively. An alternating path of type 3 for ab' is not defined nor used in the algorithm. No alternating path of type 1 or 2 contains ab', hence M'' cannot contain T in those cases. The algorithm simply shrinks nodes a and b together.

Suppose we have the situation in graph A6 in Figure 30 and the node b plays the role of w. Let b' and b'' denote the endnodes inside b of the substructures with models M5 and M2, respectively. Let us assume there is an edge $b'b''$ inside b. Suppose the exchange on alternating path $P(W)$ through b creates a triangle in M'' with the edges ab' and ab''. So, the path $P(W)$ must contain the edge $b'b''$, which is not in M, and neither of the edges ab' and ab''. At some point while running the algorithm, we must first have shrunk the edge $b'b''$. If it were not in a substructure, then it could not be on the alternating path P. Let S_i be the substructure that contained $b'b''$ when it first shrunk. Consider, in Subroutine 1^*, all the edges that could play the role of $b'b''$, where b' and b'' shrink and there can exist edges ab', ab'' in M, where a is not shrinking. Among those cases, consider these when there exists an alternating path that contains $b'b''$ but neither ab' nor ab''. The only possibility is a substructure with model M24 where nodes c and d are identified. However, in this situation we would have created a substructure with model M11 in Subroutine 2. At no future point in the algorithm could the nodes a' and a'' then shrink without also shrinking b, hence the situation could not occur.

Suppose we have the situation in graph A7 in Figure 30 and the node a plays the role of w. Let a' and a'' denote the endnodes inside a of the edges corresponding to models M6 and M2, respectively. Let us assume there is an edge $a'a''$ inside a. Let us first assume $a'a''$ is in M. Because a is a type 1 shrunk node with base edge $a'b$, a substructure S_i with model M8 was previously identified in a call to Subroutine 1, Step 1 (which is the only way to initially form a substructure with model M8) and the nodes a, b, c (using the labeling for model M8 in Figure 11) were identified for shrinking. Consider the end of this call, before the shrinking occurs. The edge $a''b$ of M in A7 connected, say, nodes b and d in S_i. Let us consider the possible substructures S_j at this point in the algorithm that could contain edge $a''b$ of M. The edge of S_j could be the edge of M in a substructure with model M2; M3; M11 (either edge of M); M12 (the edge bc); M13 (the edge ab); M14; M15; M17 or M18 (the edge ab); or M19 (either edge ab or de). For S_j with model M2, we would have a forbidden pair of substructures. For S_j with model M3, we have established that $a''b$ cannot occur in a triangle of M after an exchange (and a substructure with model M3 cannot transform to any other substructure; it can only shrink). For S_j with model M11 (for edge ab), M12, M13, M17, M18, or M19, this cannot occur because in each case these two substructures imply a di-cycle in the associated digraph D. So, this can only occur for S_j with model M11 using edge bc or model M14 using edge ac (with two possible orientations of S_i). Observe that these edges of M in S_j cannot change to a substructure with M2 as in our subsequent situation A7. So, this situation cannot occur.

Let us next assume, for the A7 case, that the edge $a'a''$ is not in M. Note that a base edge of a type 1 shrunk node is created only when we shrink a substructure with model M8 in Subroutine 1, Step 1 (as discussed above). Furthermore, a base edge for a type 1 shrunk node remains such an edge, unless it shrinks into a larger shrunk node. Let us denote by x the first type 1 shrunk node for which $a'b$ was a base edge. Note that the only edges inside x and incident with the base node that can be in an alternating path are the two edges of M incident with it (since the base node was black in a substructure with model M8). So, a' is incident with two edges of M that are inside x and the edge $a'a''$ cannot be inside x, since it must be in an alternating path in order to end up in an updated M. Consider the point in the algorithm, call it time t, where Subroutine 1* has finished a run and we are about to shrink $a'a''$ for the first time in Step 2 of Subroutine 1. Since $a'a''$ is in an alternating path through shrunk node a, it must be in a substructure at time t. The possibilities are a substructure with model M6, M7, M8, M9, M17, M20, or M24. M6 and M7 are not possible because the base node a'' would be incident with two edges of M inside a shrunk node, hence there can be no edge $a''b$ in M. M8, M17, and M20 are not possible because each would imply a di-cycle in D. M9 is not possible because this substructure with $a'b$ is a prohibited pair. Thus $a'a''$ must be a substructure with model M24 and it must serve as the temporary substructure for this shrinking since it must be in an alternating path through the shrunk node a. We claim that, at time t in the algorithm, the edge $a''b$, which is in M, must be a substructure with model M2. Suppose this is not the case. By assumption, we know that $a''b$ is a substructure with model M2 at the later time in the algorithm being considered for this case A7. Hence, there must be some time, with the shrinking at time t or later, when $a''b$ switches to a substructure with model M2. However, an examination of the cases in the algorithm when an edge of M switches to a substructure with model M2 shows that no such edge has each endnode either white or shrunk, as is the case for $a''b$ at time t and later. This is a contradiction, so $a''b$ is a substructure with model M2 at time t. However, we have noted that at time t the edge $a'a''$ is a substructure with model M24. Thus, rather than shrinking, the algorithm would have identified (through Step 2, part 1 in the Main Algorithm) the triangle $a'a''b$ in Subroutine 2 as satisfying case B2, where the triangle would have become a substructure with model M14. This contradicts our assumption that $a'a''$ shrinks just after time t. So, there can be no edge $a'a''$ not in M that can occur in an exchange on an alternating path through shrunk node a.

Finally, let us consider the case of a triangle formed by two edges of the form aw' and aw'', where a is outside w; w' and w'' are inside w; both aw' and aw'' are substructures with model M24; and $w'w''$ is an edge inside w. Clearly, in this case at most one of the two edges aw' and aw'' can be in M''; this would be the edge that is the uv-substructure that created path P.

Stage $2'$

For the remainder of the proof, we let $\bar{W}$ denote the graph obtained by undoing the final shrinking that yielded w and adding to this the edges in $boundary(w)$. (Thus $\bar{W}$ may contain shrunk nodes.) Using our previous notation, let us say the shrinking that yielded w was due to a temporary substructure in $\bar{W}$ with endnodes u and v.

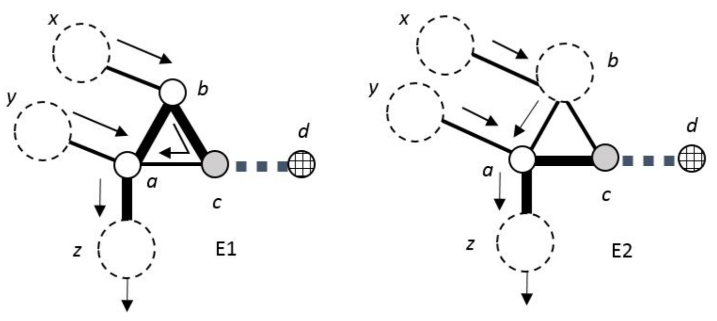


Figure 31: Origin of possible triangles in our updated simple 2-matching

In this stage, we address the possibility that M'' contains a triangle completely contained in w with all its edges in $\bar{W}$ (hence no edges are inside a shrunk node of $\bar{W}$).

Note that when new alternating paths are defined in Subroutine 1^*, they are constructed using previously defined paths with a *standard* form, say P^u, P^u_a, P^{uv}, and P^v, plus, in some cases, an initial segment of the path that has a *non-standard* form and involves a short path through a single substructure that is being shrunk (where we exclude any initial edges outside w). For example, consider a substructure with model M12 where a and b are the only nodes in $C(S_i) \cup Ext(S_i)$. In this case we shrink nodes a, b, and c. For pendant edge dc we define the type 2 alternating path $dc \cup ca \cup ab \cup P^u_b \cup P^{uv} \cup P^v$; for our purposes here we drop dc which is outside w to get our non-standard and standard subpaths. And for a substructure with model M13 where a and b are the only nodes in $C(S_i) \cup Ext(S_i)$, we define the type 1 alternating path $cb \cup P^u_b \cup P^{uv} \cup P^v$.

Consider a substructure obtained by unshrinking w. If the standard part of $P(W)$ passes through the substructure, then we can use the same arguments from Stages 1 and 2 from earlier in the proof of the theorem to conclude that a triangle containing an edge of the substructure cannot be in M''. Of course, all triangles that are inside w, have all edges in $\bar{W}$, and contain no edges of $P(W)$, are not in M''.

Let us consider the cases where the path $P(W)$ contains a non-standard path through a substructure. By examining Subroutine 1^*, we see that in many cases (except for a few special cases discussed below), the edges in these substructures cannot occur in a triangle of M'' using the same arguments as above. For example, again consider a substructure S_i with model M12. Suppose when shrinking w we have the case that the nodes of S_i corresponding to a and b are the only nodes of S_i in $C(S_i) \cup Ext(S_i)$. Then S_i shrinks and performing an exchange on the type 2 alternating path for dc results in two of the three edges of the triangle abc being in M'', hence, none of these edges can be in a triangle of M''.

The cases when our previous arguments do not work involve substructures with models M13, M16, M17, M18, M19, and M20. Let us focus on M13 and M16. For an example, again consider a substructure S_i with model M13. Suppose when shrinking w we have the case that the nodes of S_i corresponding to a and b are the only nodes of S_i in $C(S_i) \cup Ext(S_i)$. Then S_i shrinks and performing an exchange on the type 1 alternating path from node c results in the triangle having only the one edge ab in M''. Hence, it does not immediately follow that this edge is not in a triangle of M'' inside w, with all edges in $\bar{W}$. So, let us consider our list above of possible edges that can occur in a triangle of M''. By applying our rules for combining substructures, it follows that ab cannot occur in a triangle with an edge in substructures with models M2, M8, M9, M10, or M23. The possible ways for the nodes a and b in S_i to share a node with a substructure with models M5 or M6 are shown in graph E1 in Figure 31. Observe that no pair of the illustrated nodes x, y, and z can be identified to form a triangle without violating our combination rules. Thus, the final possibility to obtain a triangle in M'' containing ab is to use one substructure with model M5 or M6 (as illustrated in graph E1) together with one substructure with model M24. The possibilities are illustrated in graphs C1 and C3 in Figure 19, where the roles of xa and xb can be interchanged in C1. Note that these cases are identified and dealt with in Subroutine 2 by transforming C1 and C3 to the substructures in graphs C2 and C4, respectively. The logic for the case that S_i has model M16 is similar and the details are illustrated in Figure 23. The same logic applies to cases involving M17, M18, M19, and M20.

We now have shown that all triangles that are inside w, with all edges in $\bar{W}$, are not in M''.

Stage $3'$

Finally, we consider the possibility that there is a triangle in M'' with all edges inside w and at least one edge inside a shrunk node of $\bar{W}$. For this, we recursively apply the arguments from Stages 1, 2, and 3 of the proof of the theorem and Stages $1'$ and $2'$ from this proof of the claim to the shrunk nodes in $\bar{W}$.

This concludes the proof of the claim and the theorem.

□

□

Before proving the algorithm solves problem P_3, we show the algorithm has polynomial worst-case time complexity.

Proposition 10. *The algorithm can be implemented with polynomial worst-case time complexity.*

Proof. In the Main Algorithm we have the following:

At the uppermost level, the algorithm repeatedly executes Step 0. Between consecutive executions of Step 0, the algorithm repeatedly executes Step 1. Each execution of Step 1 is followed by an execution of Step 2, 3, or 4. Each of these executions ends with a return to Step 1, except Step 2, part 3, which is a return to Step 0. To prove the result, we calculate a bound on the number of times Step 0 is executed, and a bound on the number of times Step 1 is executed between consecutive executions of Step 0. We then calculate bounds on the work performed in Step 0 and Step 1 and between consecutive executions of Step 1.

We execute Step 0 at most $|V|+1$ times, since at the start of each execution, except the first, we have added one edge to M in Step 2, part 3. Consider the time between the starts of two consecutive executions of Step 0.

Observe that each execution of Step 0 requires $O(|V|)$ time, up front, to initialize the substructures. Just before returning to Step 0, we perform exchanges on alternating paths in Step 2, part 3, which requires $O(|V|)$ time.

Let us consider how many times Step 1 can be executed between consecutive executions of Step 0. To get at this, consider the time between two consecutive executions of Step 1. Steps 2 (except part 3), 3, and 4 all finish executions with a return to Step 1. Step 2, part 1 calls Subroutine 2 where the total number of substructures is reduced by at least one and then returns to Step 1. Step 2, part 2 creates a temporary substructure with model M24 and results in a shrinking in Step 2, part 4, followed by a return to Step 1. Step 2, part 4, of course, results in a shrinking. If we execute Step 3, part 1, then, in Subroutine 3, we transform a grey or striped node (node c) to a white node and return to Step 1. If we execute Step 3, part 2, then, in Subroutine 4, we transform a striped node (node v) into a black or grey node and return to Step 1. Otherwise, we perform a shrinking in Step 2, part 4 or in Step 4. So, in each execution of Step 1, we either (1) reduce the number of substructures; (2) transform a grey or striped node to a white node; (3) transform a striped node to a black or grey node; or (4) perform a shrinking. We can perform a reduction in the number of substructures at most $|E|$ times, consecutively. A node can be transformed at most twice (from striped to grey to white) for a total, over all nodes, of at most $2|V|$ transformations. And, we can perform a shrinking at most $|V|$ times. So a worst case scenario is to alternate between $|E|$ substructure reductions (1) and one node transformation (2) or (3) or one shrinking (4). This is at most $2|V||E|$ or $O(|E||V|)$ passes through Step 1.

We next find an upper bound on the time between the start of consecutive executions of Step 1 during a single execution of Step 0.

Observe that the work in Step 1 itself requires at most $O(|E|)$ operations to identify the proper set of edges and check the types of their endnodes. We then execute Step 2, followed by Step 2, part 1, part 2 (plus part 4), or part 4 alone; or we execute Step 3; or we execute Step 4. Let us consider time bounds for each of these executions.

Step 2: Selecting S_i takes constant time if we keep a list of the standard substructures and a list of the temporary substructures.

Step 2, part 1: Deciding if we have a situation described in Subroutine 2 is bounded by a constant. The update to the substructures in Subroutine 2 is $O(|V|)$, since a new alternating path may be created.

Step 2, part 2: The update to the substructures is $O(|V|)$, since a new alternating path is created. We then go to Step 2, part 4 and execute Subroutine

2, which, as determined below, requires $O(|E||V|)$ time.

Step 2, part 4: Subroutine 1: Determining the set C and initializing the sets $V(S_i)$, $C(S_i)$, $Unproc(S_i)$, $Proc(S_i)$, $Int(S_i)$, and $Ext(S_i)$ requires $O(|E|)$ time, since there are $O(|E|)$ substructures and each substructure is bounded in size by a constant. This is done only once. Each time a substructure is considered in Subroutine 1*, at least one node is changed from unprocessed to processed in that substructure, hence we perform $O(|E|)$ such updates. With each update, we may also update the alternating paths (types 1, 2, and 3) associated with the substructure; each alternating path update takes at most $O(|V|)$ time. Thus, each pass through Subroutine 1 takes at most $O(|E||V|)$ time (again noting that each substructure is bounded in size by a constant).

Step 3: Checking for such an edge uv requires at most $O(|E|)$ time. The time to update the substructures in Subroutines 3 and 4 is bounded by a constant. Updating the alternating paths is bounded by $O(|V|)$. So Step 3 is bounded by $O(|E|)$ time.

Step 4: Checking for a special substructure with model M5 or M6 requires $O(|E|)$ time. We then execute Subroutine 1, which, as determined above, requires $O(|E||V|)$ time.

We can now see that the time between consecutive executions of Step 1, during a single execution of Step 0, is bounded by $O(|E||V|)$.

We can now compute a complexity bound for the entire algorithm: $O(|V|(|V|+|E||V|(|E||V|)))$. Note that the first $|V|$ is the number of times Step 0 is executed. The following parentheses contain the time between consecutive executions of Step 0, where the $|V|$ is the time for Step 0 itself and the following $|E||V|$ is the number of times Step 1 is executed. The following parentheses contain the time between consecutive executions of Step 1, between consecutive executions of Step 0. This simplifies to $O(|E|^2|V|^3)$. The proposition follows.

□

Observation: The complexity of the algorithm described in the proof of Proposition 10 can be improved by better use of data structures to keep track of the alternating paths. Our purpose here is only to show the algorithm is polynomial.

Suppose the algorithm has stopped running. Using the final set of substructures, we identify a set of tri-blossom clusters of the input graph G. Consider the collection of connected components, call it $\mathcal{H}$, of the subgraph of $\tilde{G}$ induced by the shrunk nodes at the end of the algorithm. For each $H \in \mathcal{H}$, we construct a tri-blossom cluster of G called $F(H)$. To begin, select an $H \in \mathcal{H}$ and initialize $F(H)$ to be the subgraph of G corresponding to H, where the shrunk nodes in H define the centers of $F(H)$; i.e., the nodes inside each shrunk node of H are a center of $F(H)$.

Before defining the petals for $F(H)$, we note some properties of the substructures. Since there are no temporary substructures at the end of the algorithm and due to Proposition 5, we have that each substructure with model M2, M7, M11, M12, M13, M14, M15, or M16 has at least one white node. Also due to

Proposition 5, we have the following: For each substructure with model M13 or M17, at least one of nodes a and b is white; for each substructure with model M18 or M20, node a is white; and for each substructure with model M19, at least one of a and b is white and at least one of d and e is white. Due to Step 4 of the algorithm, we have that in each substructure with model M5, the node a is white, and in each substructure with model M6, the node b is white.

With these properties, we next define petals for $F(H)$. After that we show that our construction satisfies the properties T1-T8 of a tri-blossom cluster.

Every substructure with model M7, M11, M12, M14, or M15 that shares a shrunk node with H is added to $F(H)$ as a tri-petal. Every substructure with model M2, M5, or M6 that shares a shrunk node with H is added to $F(H)$ as an edge-petal. For every substructure with model M13, M16, M17, M18, M19, or M20, if node a or b is in H, then ab is added to $F(H)$ as an edge-petal. Similarly for node d or e in M19. Thus, each white node in these petals plays the role of a petal tip for $F(H)$.

In the following proposition, we show that $F(H)$, as defined above, satisfies the properties of the tri-blossom clusters. We also prove another useful property.

Proposition 11. *At the end of a run of the algorithm, let M be the final tri-free simple 2-matching and let $F(H)$ be a graph constructed from $\tilde{G}$, as described above. Then $F(H)$ is a tri-blossom cluster and is saturated by M.*

Proof. Let us denote the centers for $F(H)$ as $C_1 \dots C_k$ and let us construct the graphs $B_M(C_i)$ as we did in Section 3. Consider an arbitrary $B_M(C_i)$ arising from a shrunk node v. Let v_b denote the base node inside v. By examining the cases generated by our different types of petals defined for $F(H)$ together with different combinations of shrunk nodes for these petals, we see that every petal for $B_M(C_i)$ is one of the following:

1. An edge-petal in M. (Examples: For a substructure with model M5 with node b shrunk, the edge ab has this property for shrunk node b. Node b may or may not be a root. For a substructure with model M11 with a and b shrunk, the edge ab has this property for shrunk node a. Node a may or may not be a root.)

2. A 2-tip tri-petal that has two edges in M, where exactly one of the two edges of M is incident with a node inside v. (Example: For a substructure with model M11 with a the only shrunk node, the triangle has this property for shrunk node a.)

3. An edge-petal not in M that contains v_b, where v has type 1. (Example: For a substructure with model M6, the edge ab has this property for shrunk node a. For a 1-tip tri-petal arising from a substructure with model M7, edge ca has this property for shrunk node a and edge bc has this property for shrunk node b. For a 1-tip tri-petal arising from a substructure with model M14 where a is shrunk, the edge bc has this property for shrunk node

b. Note that, in this last example, bc is not the base edge, but essentially plays that role.)

4. A 2-tip tri-petal that contains v_b and has two edges in M that both contain v_b and one of which is the base edge for v. (Example: For a substructure with model M11 with b the only shrunk node, the triangle has this property for shrunk node b.)

5. A 2-tip tri-petal that contains v_b, contains exactly one edge in M, and the edge in M does not contain v_b, where v has type 1 and its base edge is contained in the tri-petal. (Example: For a substructure with model M14 with only the node b shrunk, the triangle has this property for shrunk node b.)

It follows from points 1 and 2 above and Properties A1, A2, and A3 (in Section 5.4) that each node v' inside v, different from v_b, is incident with two edges of M in $B_M(C_i)$, hence it is at its target value in $B_M(C_i)$. It also follows that v' is incident with at most one petal for $B_M(C_i)$ and that such a petal is at its target value in $B_M(C_i)$. If we have the situation in point 3, then we have v_b incident with two edges of M (both inside v), hence v_b is at its target value in $B_M(C_i)$; the edge-petal is not in M, hence it is one below its target value in $B_M(C_i)$. If we have the situation in point 4, then v_b is at its target value in $B_M(C_i)$, the petal is one below its target value in $B_M(C_i)$, and no other type of petal listed can also contain v_b. If we have the situation in point 5, then v_b is at its target value in $B_M(C_i)$, since v has type 1, the petal is one below its target value in $B_M(C_i)$, and no other petal can contain v_b since the petal contains the base edge for v.

Finally, suppose we have a petal from point 1 or 2 that is incident with v_b. If v_b is deficient in M, then it is deficient by 1 where v is a root node, and v_b is incident with just the one petal, which is at its target value. If v_b is not deficient, then v_b must have a base edge in M. Hence the base edge is in M8, M9, M13, M17, M18, or M19. None of these possibilities generates a new petal at v_b. Hence, v_b is one below its target value in $B_M(C_i)$ and incident with just the one petal, which is at its target value.

Property T5 follows immediately from this discussion. For Property T6, we can conclude that $B_M(C_i)$ has an odd number of petals from the following: the target for every node in C_i is 2; the target for every petal is odd; each node in C_i and each petal is at its target, except exactly one (it can be a node or petal) that is below its target by 1; and the sum of the degrees of the edges of M in $B_M(C_i)$ is even. The above discussion also demonstrates that $B_M(C_i)$ is saturated by M. It then follows from Observation 1 (see page 12) that $F(H)$ is saturated by M.

□

We next consider the following slight variation. Define $\mathcal{H}$ as above. For each $H \in \mathcal{H}$, we construct a tri-blossom cluster of G called $F'(H)$ to be the same as $F(H)$ except: For every substructure with model M13 or M16, if node a or b is in H and node c is not in a substructure with model M22 that contains exactly one edge, only then is abc added to $F'(H)$ as a tri-petal. The following corollary follows easily from Proposition 11.

Corollary 4.1. *At the end of a run of the algorithm, let $F'(H)$ be a graph constructed from $\tilde{G}$, as described above, and let M be the final tri-free simple 2-matching. Then $F'(H)$ is a tri-blossom cluster and is saturated by M.*

Observation 2: An examination of our substructures and their models shows that any edge of $\tilde{G}$ with both endnodes shrunk in H is contained in a 1-tip tri-petal we have defined for $F(H)$ and $F'(H)$. Furthermore, the only edges of $\tilde{G}$ with one endnode shrunk in H and one endnode white are contained in the petals for $F(H)$ and $F'(H)$. Consider an edge xy in $\tilde{G}$, where x is shrunk and xy is not in a substructure. Then y cannot be white in $F(H)$ (or $F(H')$); and y cannot be shrunk. If there were such an edge, it would have become a substructure with model M24, hence, in a substructure. Furthermore, y cannot be grey in a tri-petal of $F(H')$. If it were then it would have been considered in Step 3, part 1.

Theorem 5. *The algorithm outputs a maximum cardinality tri-free simple 2-matching.*

Proof. Let M denote the tri-free simple 2-matching output by the algorithm. We produce sets U, W, $\mathcal{T}$, $\mathcal{C}$, R, and R^* such that the sum of the degrees of the edges in M equals $\alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 - \alpha_5$.

At the end of the algorithm: Let U^* be the set of grey nodes contained in a substructure with model M22 that contains exactly one edge. Let R^* denote the remaining grey nodes. Let U be the set of black nodes plus U^*. Let W be the set of white nodes plus R^*. Let $\mathcal{T}$ denote the triangles with no node in U^* and contained in substructures with models M7, M11, M12, M13, M14, M15, and M16.

Let $\mathcal{C}$ denote the connected components of the subgraph of $\tilde{G}$ induced by the shrunk nodes. Note that the graphs in $\mathcal{C}$ are connected components of $G(V - U - W)$ since the nodes in $G(V - U - W)$ are all the shrunk and striped nodes, and there are no edges of G between a shrunk node and a striped node; otherwise, Step 3 part 2 in the Main Algorithm would have been executed. Observe that, by our definitions, $\mathcal{C}$ is the set $\mathcal{H}$. For each $C \in \mathcal{C}$, consider the subgraph that is the union of C, the triangles of $\mathcal{T}$ that share one or two nodes with C, and the remaining edges from C to W (with their endnodes). Let $\mathcal{C}^*$ denote the subgraphs from this collection that are tri-blossom clusters where the incident triangles of $\mathcal{T}$ are the tri-petals, and the edges from C to W, that are not in a triangle of $\mathcal{T}$, are the edge-petals. From our definition of $\mathcal{H}$, Corollary 4.1, and Observation 2, the graphs in $\mathcal{C}^*$ are the same as the graphs $F'(H)$, for all $H \in \mathcal{H}$, which are tri-blossom clusters. The nodes in $CN(\mathcal{C}^*)$ are the nodes of G inside the shrunk nodes, which are the center nodes of $\mathcal{C}^*$. Set $R = V - U - W - CN(\mathcal{C}^*)$, which equals the set of striped nodes.

Note that, for each grey node $r \in R^*$, the triangle $T_r \in \mathcal{T}$ that contains r satisfies the properties (1), (2), and (3) given in Section 4. (In particular, if there were an edge, not in T_r, from R^* to $V-(U\cup R\cup R^*)$, then Step 3, part 1 would have been executed in the Main Algorithm.)

From $\mathcal{C}^*$ construct the tri-blossom clusters $\mathcal{C}^{**}$. Observe that, by our choice of $\mathcal{C}$ and R^*, the tri-blossom clusters $\mathcal{C}^{**}$ are precisely the tri-blossom clusters $F(H)$, for $H\in\mathcal{H}$, as defined above (see Observation 2).

Observe that each node in U is incident with two edges of M and the other endnode of each such edge is either white or shrunk, hence not in U. Thus the edges in E_1 contribute the amount α_1 to the sum of the degrees of the edges in M.

As shown in Propostion 11, the edges in E_2 contribute the amount α_2 to the sum of the degrees of the edges in M.

Next consider $E_3=\gamma(R\cup R^*)\cup E[R\cup R^*, V-U-(R\cup R^*)]$. The nodes in R are the striped nodes, each of which is incident with two edges of M that are in E_3. Also, the nodes in R^* are grey nodes, hence each is incident with two edges of M. The other endnode of these edges is striped, grey in R^*, white, or shrunk, hence these edges are also in E_3, by our definitions. Every edge from a node in R to a node in $V-U-(R\cup R^*)$ is in M; otherwise, Step 3, part 2 in the Main Algorithm would have been executed. For each node r in R^*, there are two edges from r to nodes in $V-U-(R\cup R^*)$. These are the two edges, rr_a and rr_b, of the triangle T_r, defined in Section 4. One of these edges is in M and one is not. The edge r_ar_b can be an edge-petal of a tri-blossom cluster in $\mathcal{C}^{**}$ or an edge in M with endnodes in W. Hence, the edges in E_3 contribute the amount $\alpha_3-|R^*|$ to the sum of the degrees of the edges in M.

Next consider $E_4=\gamma(W-R^*)-E_2$. Since the nodes in $W-R^*$ are all white, the substructures with an edge in E_4 have models M2, M11 (which is a triangle in $\mathcal{T}_3^*$), M13 (just the edge ab), M17 (just the edge ab), M19 (just the edge ab and/or edge de), or M24. However, no substructure with model M24 is possible since we would have then executed Step 2 in the Main Algorithm. So, the edges in E_4 contribute the amount $\alpha_4-2|\mathcal{T}_3^*)|$ to the sum of the degrees of the edges in M.

Hence, the sum of the degrees of the edges in M is $\alpha_1+\alpha_2+\alpha_3+\alpha_4-\alpha_5$ and, with Proposition 1, the result follows.

□

Corollary 5.1. *Theorems 1 and 2 hold.*

Proof. These follow immediately from the proof of Theorem 5.

□

8 Acknowledgements

I'd like to thank Gérard Cornuéjols and Bill Pulleyblank for introducing me to the problem of finding tri-free 2-factors. This was back in the early 1980s when I was a Ph.D. student. The bulk of my thesis was devoted to presenting a polynomial-time algorithm for the problem (different from the one presented in this paper). Gérard served as my Ph.D. advisor at Carnegie Mellon University where we spent countless hours working together on this project. His generous help and expert guidance were invaluable. Bill was kind enough to take me on as a post-doc at the University of Waterloo where we had many helpful further discussions of this problem. The revamped version of the algorithm that appears in this paper was very much influenced by both Gérard and Bill. Furthermore, through recent discussions, they helped improve the presentation in this paper.

I would also like to thank the anonymous referees whose work greatly improved the paper.

References

[1] Maxim Babenko, Alexey Gusakov, and Ilya Razenshteyn. Triangle-free 2-matchings revisited. *Discrete Mathematics, Algorithms and Applications*, 2(04):643–654, 2010.

[2] Maxim A Babenko. Improved algorithms for even factors and square-free simple b-matchings. *Algorithmica*, 64(3):362–383, 2012.

[3] Hans-Boris Belck. Reguläre faktoren von graphen. *Journal für die reine und angewandte Mathematik*, 1950(188):228–252, 1950.

[4] Kristóf Bérczi. The triangle-free 2-matching polytope of subcubic graphs. Technical report, Technical Report TR-2012-2, Egerváry Research Group, 2012.

[5] Kristóf Bérczi and Yusuke Kobayashi. An algorithm for (n-3)-connectivity augmentation problem: Jump system approach. *Journal of Combinatorial Theory, Series B*, 102(3):565–587, 2012.

[6] Kristóf Bérczi and László A Végh. Restricted b-matchings in degree-bounded graphs. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 43–56. Springer, 2010.

[7] Claude Berge. Two theorems in graph theory. *Proceedings of the National Academy of Sciences of the United States of America*, 43(9):842, 1957.

[8] Claude Berge. Sur le couplage maximum d'un graphe. *Comptes Rendus Hebdomadaires des Séances de l'Académie des Sciences*, 247:258–259, 1958.

[9] Sylvia Boyd, Satoru Iwata, and Kenjiro Takazawa. Finding 2-factors closer to TSP tours in cubic graphs. *SIAM Journal on Discrete Mathematics*, 27(2):918–939, 2013.

[10] Sylvia C Boyd and William H Cunningham. Small travelling salesman polytopes. *Mathematics of Operations Research*, 16(2):259–271, 1991.

[11] William Cook and William R Pulleyblank. Linear systems for constrained matching problems. *Mathematics of Operations Research*, 12(1):97–120, 1987.

[12] William John Cook. On some aspects of totally dual integral systems. *Ph. D. Thesis, Department of Combinatorics and Optimization, University of Waterloo*, 1983.

[13] Gérard Cornuéjols and William R Pulleyblank. A matching problem with side conditions. *Discrete Mathematics*, 29(2):135–159, 1980.

[14] Gérard Cornuéjols and William R Pulleyblank. Perfect triangle-free 2-matchings. In *Combinatorial Optimization II*, pages 1–7. Springer, 1980.

[15] William H Cunningham. Matching, matroids, and extensions. *Mathematical programming*, 91(3):515–542, 2002.

[16] William H Cunningham and Yaoguang Wang. Restricted 2-factor polytopes. *Mathematical programming*, 87(1):87–111, 2000.

[17] Jack Edmonds. Maximum matching and a polyhedron with 0, 1-vertices. *Journal of research of the National Bureau of Standards B*, 69(125-130):55–56, 1965.

[18] Jack Edmonds. Paths, trees, and flowers. *Canadian Journal of mathematics*, 17:449–467, 1965.

[19] Jack Edmonds, Ellis L Johnson, and Scott C Lockhart. Blossom i: a computer code for the matching problem. *IBM TJ Watson Research Center, Yorktown Heights, New York*, 1969.

[20] Marshall L Fisher, George L Nemhauser, and Laurence A Wolsey. An analysis of approximations for finding a maximum weight hamiltonian circuit. *Operations Research*, 27(4):799–809, 1979.

[21] András Frank. Restricted t-matchings in bipartite graphs. *Discrete Applied Mathematics*, 131(2):337–346, 2003.

[22] Tibor Gallai. On factorisation of graphs. *Acta Mathematica Hungarica*, 1(1):133–153, 1950.

[23] Tibor Gallai. Maximum-minimum sätze über graphen. *Acta Mathematica Hungarica*, 9(3-4):395–434, 1958.

[24] Jim Geelen. Personal communication. 2000.

[25] Martin Grötschel and William R Pulleyblank. Clique tree inequalities and the symmetric travelling salesman problem. *Mathematics of Operations Research*, 11(4):537–569, 1986.

[26] David Hartvigsen. *Extensions of matching theory.* PhD thesis, Carnegie Mellon University, under the supervision of Gérard Cornuéjols, 1984.

[27] David Hartvigsen. Finding maximum square-free 2-matchings in bipartite graphs. *Journal of Combinatorial Theory, Series B*, 96(5):693–705, 2006.

[28] David Hartvigsen and Yanjun Li. Maximum cardinality simple 2-matchings in subcubic graphs. *SIAM Journal on Optimization*, 21(3):1027–1045, 2011.

[29] David Hartvigsen and Yanjun Li. Polyhedron of triangle-free simple 2-matchings in subcubic graphs. *Mathematical Programming*, 138(1-2):43–82, 2013.

[30] Pavol Hell, David Kirkpatrick, Jan Kratochvíl, and Igor Kříž. On restricted two-factors. *SIAM Journal on Discrete Mathematics*, 1(4):472–484, 1988.

[31] Zoltán Király. C_4-free 2-factors in bipartite graphs. Technical Report TR-2001-13, Egerváry Research Group, Budapest, 2001. www.cs.elte.hu/egres.

[32] Yusuke Kobayashi. A simple algorithm for finding a maximum triangle-free 2-matching in subcubic graphs. *Discrete Optimization*, 7(4):197–202, 2010.

[33] Yusuke Kobayashi. Triangle-free 2-matchings and M-concave functions on jump systems. *Discrete Applied Mathematics*, 175:35–42, 2014.

[34] Yusuke Kobayashi. Weighted triangle-free 2-matching problem with edge-disjoint forbidden triangles. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 280–293. Springer, 2020.

[35] Yusuke Kobayashi, Jácint Szabó, and Kenjiro Takazawa. A proof of cunninghams conjecture on restricted subgraphs and jump systems. *Journal of Combinatorial Theory, Series B*, 102(4):948–966, 2012.

[36] Marton Makai. On maximum cost $K_{t,t}$-free t-matchings of bipartite graphs. *SIAM Journal on Discrete Mathematics*, 21(2):349–360, 2007.

[37] Yunsun Nam. *Matching theory: Subgraphs with degree constraints and other properties.* PhD thesis, University of British Columbia, under the supervision of Richard Anstee, 1994.

[38] George L Nemhauser and Leslie Earl Trotter. Vertex packings: structural properties and algorithms. *Mathematical Programming*, 8(1):232–248, 1975.

[39] Gyula Pap. Combinatorial algorithms for matchings, even factors and square-free 2-factors. *Mathematical programming*, 110(1):57–69, 2007.

[40] Julius Petersen. Die theorie der regulären graphs. *Acta Mathematica*, 15(1):193, 1891.

[41] WR Pulleyblank. *Faces of Matching Polyhedra, Univ. of Waterloo, Dept. Combinatorics and Optimization.* PhD thesis, Ph. D. Thesis, 1973.

[42] M Russell. Restricted 2-factors. Master's thesis, University of Waterloo, under the supervision of William Cunningham, 2001.

[43] Alexander Schrijver. *Combinatorial optimization: polyhedra and efficiency*, volume 24. Springer Science & Business Media, 2003.

[44] Kenjiro Takazawa. A weighted $K_{t,t}$-free t-factor algorithm for bipartite graphs. *Mathematics of Operations Research*, 34(2):351–362, 2009.

[45] Kenjiro Takazawa. Decomposition theorems for square-free 2-matchings in bipartite graphs. In *International Workshop on Graph-Theoretic Concepts in Computer Science*, pages 373–387. Springer, 2015.

[46] Kenjiro Takazawa. Excluded t-factors in bipartite graphs: A unified framework for nonbipartite matchings and restricted 2-matchings. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 430–441. Springer, 2017.

[47] Kenjiro Takazawa. Finding a maximum 2-matching excluding prescribed cycles in bipartite graphs. *Discrete Optimization*, 26:26–40, 2017.

[48] William T Tutte. The factorization of linear graphs. *Journal of the London Mathematical Society*, 1(2):107–111, 1947.

[49] William T Tutte. The factors of graphs. *Canadian Journal of Mathematics*, 4:314–328, 1952.

[50] William T Tutte. A short proof of the factor theorem for finite graphs. *Canadian Journal of Mathematics*, 6:347–352, 1954.

[51] Oliver Vornberger. Easy and hard cycle covers. *Preprint, Universität Paderborn*, 1980.