SIAM J. COMPUT.
Vol. 40, No. 3, pp. 678–708

TERMINAL BACKUP, 3D MATCHING, AND COVERING CUBIC GRAPHS[*]

ELLIOT ANSHELEVICH[†] AND ADRIANA KARAGIOZOVA[‡]

Abstract. We define a problem called *Simplex Matching* and show that it is solvable in polynomial time. While Simplex Matching is interesting in its own right as a nontrivial extension of nonbipartite min-cost matching, its main value lies in many (seemingly very different) problems that can be solved using our algorithm. For example, suppose that we are given a graph with terminal nodes, nonterminal nodes, and edge costs. Then, the Terminal Backup problem, which consists of finding the cheapest forest connecting every terminal to *at least one* other terminal, is reducible to Simplex Matching. Simplex Matching is also useful for various tasks that involve forming groups of at least two members, such as project assignment and variants of facility location. In an instance of Simplex Matching, we are given a hypergraph H with edge costs and edge size at most 3. We show how to find the min-cost perfect matching of H efficiently if the edge costs obey a simple and realistic inequality that we call the *Simplex Condition*. The algorithm we provide is relatively simple to understand and implement but difficult to prove correct. In the process of this proof we show some powerful new results about covering cubic graphs with simple combinatorial objects.

Key words. graph packing, Simplex Matching, cycle cover, network design

AMS subject classifications. 05C70, 68Q25, 68R10

DOI. 10.1137/090752699

1. Introduction.

Matching theory, as well as its extensions, is both extremely important and well studied. Perhaps surprisingly, there still remain basic matching problems that can be solved efficiently and yet are not solvable using existing matching algorithms and techniques [12, 15, 21, 22]. In this paper, we address one such problem that we call *Simplex Matching* and show how to solve it in polynomial time using an elegant covering argument. While Simplex Matching is interesting in its own right as a nontrivial extension of nonbipartite min-cost matching, its main value lies in many (seemingly very different) problems that can be solved using our algorithm. After defining Simplex Matching, we give several representative examples of such problems.

Simplex Matching. The main focus of this paper lies in providing a polynomial-time algorithm for Simplex Matching, which is a generalization of min-cost nonbipartite matching. It is also a generalization of $\{K_2, K_3\}$-packing (see, e.g., [13, 22]). A lot of work has been devoted to packing of graphs with various subgraphs [6, 11, 13, 14, 21, 28] (for surveys, see [12, 15, 22]). In this context, the standard matching can be thought of as a packing with edges, i.e., a $\{K_2\}$-packing. The study of packing has a lot in common with our work as it deals with nontrivial extensions of matching, often using totally different methods. In [23], Hell and Kirkpatrick gave an

[*]Received by the editors March 16, 2009; accepted for publication (in revised form) March 31, 2011; published electronically June 2, 2011. A preliminary version of this paper appeared in the *Proceedings of the 39th ACM Symposium on Theory of Computing* (STOC), 2007.
http://www.siam.org/journals/sicomp/40-3/75269.html

[†]Department of Computer Science, Rensselaer Polytechnic Institute, Troy, NY 12180 (eanshel@cs.rpi.edu). This author's research was supported by an NSF Postdoctoral Fellowship in Mathematical Sciences.

[‡]Department of Computer Science, Princeton University, Princeton, NJ 08540 (karagioz@cs.princeton.edu). This author's research was supported by NSF ITR grant CCR-0205594, NSF CAREER award CCR-0237113, MSPA-MCS award 0528414, and Moses Charikar's Alfred P. Sloan Fellowship.

elegant algorithm (reminiscent of Edmonds's blossom algorithm) to find the perfect $\{K_2, K_3\}$-packing, [38] greatly improved its running time, and [29] classified some types of packings that can be found efficiently. Their results held only for unweighted graphs, however, where the cost of every edge is 1. Among other things, our algorithm for Simplex Matching gives a simple and intuitive way of finding the best (min-cost) perfect $\{K_2, K_3\}$-packing, even in the weighted case. The algorithm we provide is relatively simple to understand and implement but difficult to prove correct. In the process of this proof we show some powerful new results about covering cubic graphs with simple combinatorial objects.

We now define the Simplex Matching problem. In an instance of Simplex Matching, we are given a hypergraph H containing edges of sizes 2 and 3 with edge costs $c(e)$. Our goal is to find a perfect matching of H with minimum cost. Since H is a hypergraph, a perfect matching here (often referred to as a packing) is simply a collection S of edges in H such that every node of H appears in exactly one edge of S.

This problem is NP-hard without additional constraints on the costs $c(e)$. To see this, notice that if H contains only edges of size 3, we have exactly three-dimensional (3D) Matching. However, in the applications that the authors are interested in, H satisfies the following extra condition, which we call the *Simplex Condition.* It states that for every 3D edge (u_1, u_2, u_3), the corresponding two-dimensional (2D) edges also exist (see Figure 1.1(a)), with the cost relation of

$$c(u_1, u_2) + c(u_1, u_3) + c(u_2, u_3) \leq 2 \cdot c(u_1, u_2, u_3).$$

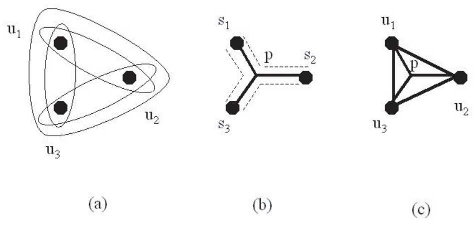

FIG. 1.1. *The Simplex Condition in Simplex Matching, Project Assignment, and Terminal Backup.*

It is worth pointing out that by replacing 2 with $2 + \epsilon$ in the Simplex Condition, this problem becomes NP-complete.

PROPOSITION 1.1. *If the Simplex Condition above is relaxed by replacing* 2 *with* $2+\epsilon$, *then finding a min-cost perfect matching in a hypergraph H obeying this condition is NP-complete.*

Proof. The reduction from 3D-Matching is straightforward: add the 2D edges required by the Simplex Condition, set their costs to 1, and set the costs of the 3D edges to $3/(2 + \epsilon)$. In the resulting instance of Simplex Matching, a perfect matching of cost $\leq n/(2 + \epsilon)$ (for n being the number of nodes) would use only 3D edges and so would exist exactly when a 3D perfect matching exists in the 3D-Matching instance. $\square$

Problems obeying the Simplex Condition. To understand why the Simplex Condition is natural, consider the special case where the costs $c(u, v)$ correspond to distances between u and v that obey the triangle inequality, and $c(u, v, w)$ corresponds to the cheapest way of connecting u, v, and w, which is a three-pointed star, as shown in Figure 1.1(b). In this special case, by the triangle equality, $c(u, v)$ must be smaller than the distance along the star, and so the Simplex Condition is obeyed. In general, this occurs whenever the costs $c(u, v)$ are obtained from an application where some version of triangle inequality holds. Below we give two representative examples of such problems, called Terminal Backup and Project Assignment. For more examples of problems that can be shown to obey the Simplex Condition, as well as the reductions from Project Assignment and Terminal Backup to Simplex Matching, see section 6.

Terminal Backup. Consider the following network design scenario. As in the Steiner tree problem, we are given a graph consisting of terminal nodes, nonterminal nodes, and edges with costs c_e. The terminal nodes represent facilities that need to be connected for backup purposes. To do this, we must construct a network of edges so that every facility is connected to at least one other facility. In other words, we need to find a forest of minimum cost such that every connected component of this forest contains at least two terminals. The facilities connected together can back up their data, and if any one facility failed, there would be at least one other that contains its data. In [41], Xu, Anshelevich, and Chiang discuss applications of this problem beyond simply backing up data and show how to solve it using Simplex Matching.

Project Assignment. Now consider a teacher with a list of projects for the students, who wants to break the students into groups of at least 2 (and at most k) and assign each group a project (several groups may do the same project as long as they do not work together). Also suppose that there is a function $u(s, p)$ that shows how much a student s likes project p. How should the teacher break the students into groups so that the sum of the students' utilities is maximized? This question is a special case of facility location with lower bounds and can be reduced to Simplex Matching. Notice that if the groups were allowed to be of size 1, the optimum would simply assign each student to the project she likes best. If the groups had to be of size exactly 2, this is reducible to nonbipartite matching (notice that not all projects need to be assigned; otherwise this would be easily solvable by a flow argument). And if the group size had to be at least 3, this problem would immediately become NP-hard. The variant with group size at least 2 and arbitrary k, however, is reducible to Simplex Matching and so has nontrivial structure that can be exploited to form an efficient algorithm. We discuss these two problems further in section 6.

Our results. Our main contribution consists of providing a polynomial-time algorithm for Simplex Matching, which can be used to solve a variety of related problems. The algorithm is very simple conceptually.

DEFINITION 1.2. *A* 2-factor *is a subgraph with every node in this subgraph having exactly two incident edges. For a matching M, an M-*alternating 2-factor *is a 2-factor in which every node has exactly one incident edge in M.*

Our algorithm simply starts with a perfect matching (packing) M and at every step finds an M-alternating 2-factor such that augmenting M by this 2-factor creates a significantly cheaper perfect matching. It is not surprising that such an algorithm exists, since the min-cost perfect matching can be obtained from any perfect matching if we just augment it by the correct 2-factor. What is surprising here is that a desirable 2-factor can be found efficiently. Most of the paper is devoted to proving this.

Consider how a similar algorithm would behave if we wanted to find the min-cost perfect matching without any edges of size 3. Then any 2-factor is simply a collection of cycles and could find an alternating cycle that decreases the matching cost sufficiently. In the case of Simplex Matching, however, the 2-factors can have very complex structures (see Figure 1.2), and finding a good M-alternating 2-factor

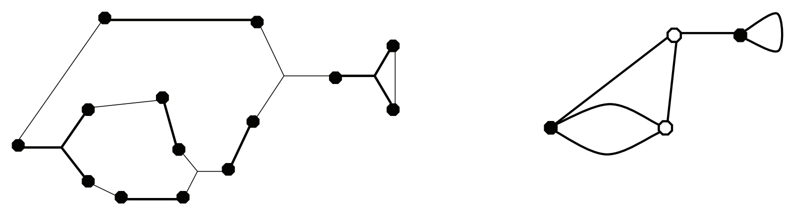

FIG. 1.2. *(Left) An M-alternating 2-factor. The bold edges are edges in M. 3D edges are drawn as a star with 3 leaves (i.e., the nodes in the middle of these stars are not real nodes). (Right) The dual of this 2-factor (see section 3). Filled circles are nodes in M.*

may seem difficult.

To get around this problem, we show that there is no need to consider arbitrary 2-factors like that in Figure 1.2, as there always exist good 2-factors with simple structures (containing at most two 3D edges), even in the weighted case. The proof of this is complex and relies on our theorem about covering arbitrary cubic graphs with simple combinatorial objects we call *dual augmentors*. For a discussion on the relationship between our results and other covering results, especially cycle covers [24, 25, 37, 42], see section 4.

Related work. Terminal Backup is similar to many Steiner-tree variations [10, 17, 20]. However, all such variations are required to either connect particular pairs of terminals, connect terminals from a particular set, or connect at least k terminals in total. The problem of finding the cheapest forest with at least k terminals in *each* connected component has not been addressed before. In addition, all of the above variations are NP-hard, while Terminal Backup is solvable in polynomial time for $k = 2$. For $k > 2$ it becomes NP-hard, although there is a 2-approximation algorithm shown in [4] using techniques from [18].

Simplex Matching and especially Project Assignment are also very similar to variants of facility location. In fact, we can use Simplex Matching to solve instances of facility location where all open facilities have lower bounds of 2 and the facility costs obey the Simplex Condition (e.g., the costs are all 0, or the cost of serving three clients is at least 3/2 times the cost of serving two clients). Although this is a very special case of facility location, it is the first result (to our knowledge) of a nontrivial facility location problem with lower bounds [2, 19, 27] that can be exactly solved efficiently.

Matching theory is a very large field (see, e.g., [30]), and there are many algorithms for weighted nonbipartite matching. A lot of work has also been done on exact *packings*, which are exact covers of a graph using more complicated combinatorial structures than just edges. For some results on packings, see, e.g., [6, 11, 13, 14, 21, 28, 32], and for surveys see [12, 15, 22]. Especially relevant to our work is packing by edges and triangles ($\{K_2, K_3\}$ packing), since choosing a 3D edge in Simplex Matching is similar to choosing a triangle for a packing. Hell and Kirkpatrick's algorithm for finding the perfect $\{K_2, K_3\}$ packing in unweighted graphs (in [23]) and the more efficient algorithm given in [38] can easily be extended to solve the unweighted version of Simplex Matching [26]. At the core of the algorithms for unweighted Simplex Matching lies the ability to efficiently find an improving M-alternating 2-factor that can be used to augment the matching. This can be done because in unweighted graphs, it is not difficult to show the existence of an improving 2-factor with simple structure (e.g., having at most two 3D edges). Showing the same result for weighted graphs, however, is significantly more complicated and requires very different methods (in fact, a large part of this paper is devoted to proving exactly this). Given that we can find an improving M-alternating 2-factor efficiently, the algorithm for weighted Simplex Matching is similar to the algorithms for unweighted graphs: just keep augmenting the matching by such a 2-factor until we reach the matching of minimum cost. Since Simplex Matching is a generalization of $\{K_2, K_3\}$ packing, our algorithm can also be used to find the min-cost $\{K_2, K_3\}$ perfect packing. Another relevant line of research is packing by cycles of length *at least* three [7, 8].

Much of the literature on packing concerns itself with matching polyhedra. Unlike the standard perfect matching, which has a nice characterization as a linear program, many similar results for packing can be extremely complicated. While the weighted perfect matching problem lends itself to a primal-dual algorithm, this is not true for Simplex Matching, for which there is no nice linear program characterizing the solutions. See [12, 15, 22, 29] for polytope characterizations of other packing problems, although most of these are either for unweighted versions or for packing problems very different from ours. An exception is Pap, who produces some results similar to ours in [33], although he uses completely different techniques and looks at this problem from quite a different perspective.

Finally, [41] is a companion paper to this one. In it the reader will find detailed applications of Simplex Matching, much discussion of the Terminal Backup problem, and an implementation of the Simplex Matching algorithm. Additionally, [4] looks at game-theoretic versions of Terminal Backup.

Paper organization. In section 2 we give an efficient algorithm to solve the weighted Simplex Matching problem. In section 3 we argue its correctness. To do this, we define the concept of *dual augmentors*, as well as a *valid augmentor sum*, and show that our algorithm is correct if Theorem 3.5 holds, which states that any cubic multigraph has a valid augmentor sum. In section 4 we prove this theorem, which is the most technical part of the paper. In section 5 we discuss the running time of our algorithm and show that it is polynomial. Finally, in section 6 we give several examples of problems that can be solved efficiently using Simplex Matching.

2. Algorithm for weighted Simplex Matching.

For the standard 2D matching we know that if we take a perfect matching M that is not the minimum-weight matching, then there exists an alternating cycle that could be used to improve the current matching. We now show a similar condition for Simplex Matching.

Definition 2.1. *To* augment *a perfect matching M by a set of edges S means to replace all edges in $M \cap S$ with the edges in $S - M$. In other words, augmenting M by S results in $M \triangle S$ (the symmetric difference).*

Let M be a perfect matching of cost $\sum_{e\in M} c(e)$. For any set of edges S, define a potential function $\phi_M(S) = \sum_{e\in M\cap S} c(e) - \sum_{e\in S-M} c(e)$. If we augment M by S, then the cost of the new set decreases by $\phi_M(S)$. Moreover, if S is an M-alternating 2-factor, then this is still a perfect matching. Recall from Definition 1.2 that an M-alternating 2-factor is a set S such that every node in S has exactly two edges incident to it, exactly one of which is in M. Let M^* be a min-cost perfect matching. The connected components of the symmetric difference $M \triangle M^*$ are M-alternating 2-factors that augment M to M^*. Therefore, there always exists an M-alternating 2-factor S with $\phi_M(S) = cost(M) - cost(M^*)$.

If we could find the M-alternating 2-factor with maximum ϕ_M, we could simply augment by it and get the min-cost perfect matching. Instead, our algorithm will proceed by finding an M-alternating 2-factor S with high $\phi_M(S)$ at each step and augmenting by it. Finding a 2-factor S with a high potential $\phi_M(S)$ seems difficult, since 2-factors for Simplex Matching can have a complex structure, as in Figure 1.2. We will show, however, that there is no need to consider arbitrary 2-factors like the one in Figure 1.2, because there always exists a good 2-factor with simple structure: it should contain at most two 3D edges. We call such 2-factors *augmentors*.

DEFINITION 2.2. *An* augmentor *is an M-alternating 2-factor that contains at most two 3D edges.*

More specifically, augmentors can be of the following types (see Figure 2.1):

Type-0. A Type-0 augmentor is an M-alternating cycle of 2D edges. This is the same as a 2D matching augmenting cycle.

Type-1. A Type-1 augmentor consists of two 3D edges (a_1, a_2, a_3) and (b_1, b_2, b_3) together with M-alternating paths of 2D edges connecting a_1 to b_1, a_2 to a_3, and b_2 to b_3. These paths must be node-disjoint, and the entire augmentor must be M-alternating (so the 3D edges may or may not be in M).

Type-2. This is the same as Type-1, but the paths connect a_1 to b_1, a_2 to b_2, and a_3 to b_3.

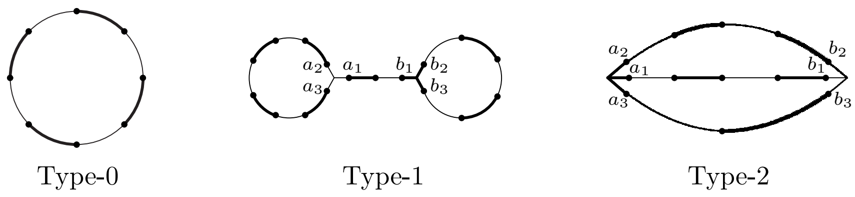

FIG. 2.1. *Simplex Matching augmentors.*

The bulk of this paper is devoted to proving that an augmentor with high potential always exists. The following lemma allows us to claim that if this is true, then we can improve the current perfect matching in polynomial time.

LEMMA 2.3. *Let A be an augmentor with maximum potential with respect to some perfect matching M in a hypergraph H with edges of sizes 2 and 3 and edge costs $c(e)$. Given H and M, we can find an M-alternating 2-factor S with $\phi_M(S) \geq \phi_M(A)$ in polynomial time (which may or may not be A itself).*

Proof. Suppose A is of Type-0. Delete all 3D edges from H as well as all nodes that are matched in M using 3D edges, forming a graph H' with no 3D edges. By the notation $M \mid_{H'}$, we will mean the subset of edges in M restricted to edges of H', i.e., only those edges of M that are also in the graph H'. Notice that $M \mid_{H'}$ is a perfect matching of H', since all nodes of H adjacent to 3D edges of M were deleted. Now find a min-cost perfect matching M^* of H'. Then, $S = M^* \triangle M \mid_{H'}$ gives us an M-alternating 2-factor to augment M by. Since A is of Type-0, it cannot include any nodes incident to 3D edges of M; thus A is contained in H'. Augmenting by S results in the best possible matching in H'; therefore, $\phi_M(S) \geq \phi_M(A)$.

Now suppose A is of Type-1 or Type-2. Fix the two 3D edges e_1 and e_2 that it contains (there are only $|E|^2$ possibilities). We will find the best M-alternating 2-factor with only e_1 and e_2 as the 3D edges. Form a new graph H' as above, except do not delete edges e_1 or e_2. If this forces some nodes incident to e_1 or e_2 to be deleted, we do not need to consider the (e_1, e_2) pair, since this can occur only if $e_1 \notin M$ shares a node with a 3D edge $e \in M$ with $e \neq e_2$, in which case for A to be an M-alternating 2-factor, it must also contain e, giving us a contradiction. Therefore, 3D edges e_1 and e_2 that A contains must be such that H' still has all the nodes incident to e_1 and e_2.

As before, we know that $M \mid_{H'}$ is a perfect matching of H', since for every edge of M that was removed, so were all the adjacent nodes. Let M^* be the min-cost perfect matching of H'. Since A is entirely contained in H', we once again know that M^* is a cheaper matching than $A \triangle M \mid_{H'}$. Therefore, if $S = M^* \triangle M \mid_{H'}$, then $\phi_M(S) \geq \phi_M(A)$. To find M^*, notice that H' is a graph with exactly two 3D edges. Thus, there are only four cases for whether those edges are in the matching, and we can try each case, computing a 2D min-cost perfect matching on the rest of H' if one exists. In fact, since we are trying only to find a matching better than $A \triangle M \mid_{H'}$, we can just assume that e_i for $i = 1, 2$ is in the matching if and only if $e_i \notin M$, and find the best 2D perfect matching on the rest of H' with this constraint. We can do this because augmenting M by A is guaranteed to add e_i to the matching exactly if it were not in it before.

We now choose the best one of the resulting $|E|^2 + 1$ M-alternating 2-factors. For analysis of the running time, as well as ways to make the above algorithm run faster, see section 5. □

In the next few sections we will show that if M is not the min-cost perfect matching, then there exists an augmentor with positive potential. Using the above lemma, we can state our algorithm for finding the min-cost weighted Simplex Matching. The initial perfect matching in this algorithm can be computed using an algorithm for unweighted Simplex Matching [23, 26, 38].

```
Start with any perfect matching (possibly containing 3D edges).
Repeat until done
  Find an $M$-alternating 2-factor better than any augmentor, and
augment by it.
```

3. Dual augmentors. We now focus on the algorithm's correctness and termination. To prove that this algorithm finds the min-cost perfect matching, we need to show that for any perfect matching M that is not of minimum cost there exists an augmentor A with $\phi_M(A) > 0$. We will accomplish this by showing that every M-alternating 2-factor of positive potential contains an augmentor of positive potential. This will be sufficient since we know that for any perfect matching M that is not minimum-cost there exists an M-alternating 2-factor S with $\phi_M(S) > 0$ (in fact, with $\phi_M(S) = cost(M) - cost(M^*)$).

For any M-alternating 2-factor S we form a dual graph S^* that is easier to deal with than S (see Figures 1.2 and 3.1). The nodes of S^* are the 3D edges of S; we will denote by v_e the node of S^* corresponding to the 3D edge e of S. For every path of 2D edges in S connecting one of the nodes of e with one of the nodes of e', there is an edge in S^* between nodes v_e and $v_{e'}$. Note that this may result in parallel edges as well as self-loops (e.g., if both (u, v, w) and (v, w) were in S). The resulting graph S^* is a cubic (3-regular) graph. We will say that a node $v \in S^*$ is in M if its corresponding 3D edge of S is in M.

DEFINITION 3.1. *A* dual augmentor *with respect to M is a connected subset of edges of S^* satisfying the following conditions:*

1. *Degree 2 everywhere except at most two nodes.*
2. *All degree 1 nodes are in M.*
3. *No degree 2 node is in M.*

Let S_{extra} be the multiset of 2D edges (u, v) such that a 3D edge (u, v, w) is in S, but $(u, v) \notin S$, as in Figure 3.1. Let $S_{aug} = S \cup S_{extra}$. As we prove below, dual augmentors are exactly the subgraphs in the dual graph S^* corresponding to augmentors in S_{aug}, as shown in Figure 3.1.

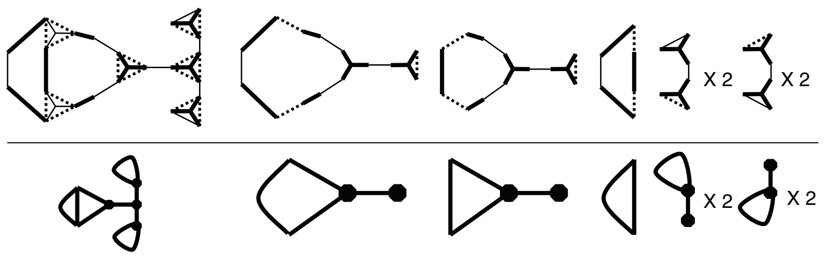

FIG. 3.1. *(Top) An M-alternating 2-factor S with edges of S_{extra} shown as dashed lines. To the right of it are some augmentors. (Bottom) The dual cubic graph S^* and the corresponding dual augmentors. The nodes of M are shown as black circles.*

Type-0. A Type-0 augmentor A of S_{aug} is an M-alternating cycle of 2D edges, some of which are in S_{extra}, like the third augmentor shown in Figure 3.1. In the dual graph S^*, every path of 2D edges in S corresponds to a single edge. Thus in the dual graph S^*, A corresponds to a cycle of nodes v_e where each e is the 3D edge that produced one of the above S_{extra} edges. Let v_e be one of these nodes, with $e = (u, w, z)$ and the edge (u, w) being the edge of S_{extra} in A. Since A is M-alternating, u and w must each belong to some edge of M in A, and neither of these edges is the edge e, since then A would contain a 3D edge and would not be an augmentor of Type-0. Therefore, this implies that $v_e \notin M$ since M is a matching, and so A corresponds to a dual augmentor that is simply a cycle with no nodes in M, to which we refer as a dual augmentor of Type-0.

Conversely, every Type-0 dual augmentor of S^* corresponds to a Type-0 augmentor of S_{aug}. Let A^* be a dual augmentor of S^* that has degree 2 at all nodes, with all nodes not in M. Let $(v_e, v_{e'})$ be an edge of A^*. Every edge in A^* corresponds to an M-alternating path of 2D edges in S; denote the path corresponding to $(v_e, v_{e'})$ by $P(v_e, v_{e'})$. Since neither v_e nor $v_{e'}$ is in M, then neither e nor e' is in M. The path $P(v_e, v_{e'})$ is a path from a node of e to a node of e', and since the original 2-factor S is M-alternating, then this path $P(v_e, v_{e'})$ must begin and end with an edge in M. We can create a Type-0 augmentor of S_{aug} by simply taking the paths of 2D edges corresponding to the edges of A^*, together with an edge of S_{extra} for every node $v_e \in A^*$. Specifically, if A^* contains two edges (v_{e_1}, v_{e_2}) and (v_{e_2}, v_{e_3}), then we include the paths $P(v_{e_1}, v_{e_2})$ and $P(v_{e_2}, v_{e_3})$ in our augmentor, as well as the edge (u, w) of S_{extra}, where u and w are nodes of e_2 that are endpoints of the paths $P(v_{e_1}, v_{e_2})$ and $P(v_{e_2}, v_{e_3})$. Note that $u \neq w$ since otherwise u would have degree 3 in S, which is a 2-factor. The above process creates a cycle in S_{aug}, and this cycle is M-alternating, since all edges of S_{extra} are not in M, and all the M-alternating paths of 2D edges that we include begin and end with an edge of M.

Type-1. A Type-1 augmentor A will produce a dual augmentor of one of three kinds, as shown in Figure 3.2. The M-alternating paths connecting a_1 to b_1, a_2 to a_3, and b_2 to b_3 behave exactly as the cycle augmentor in the previous case; namely, they correspond to paths that do not contain vertices of M. This gives us a dual augmentor of Type-1c in Figure 3.2 that is a path together with the two cycles attached to it, with only nodes $v_{(a_1,a_2,a_3)}$ and $v_{(b_1,b_2,b_3)}$ possibly being in M.

Notice that if both (a_1, a_2, a_3) and (a_2, a_3) are in A, then the "cycle" incident to $v_{(a_1,a_2,a_3)}$ in the dual augmentor is just a self-loop. Consider the special case, however, when $(a_1, a_2, a_3) \in M$ and $(a_2, a_3) \in S_{extra}$, as in many augmentors of Figure 3.1. We cannot form a self-loop in the dual augmentor, because we only formed self-loops in

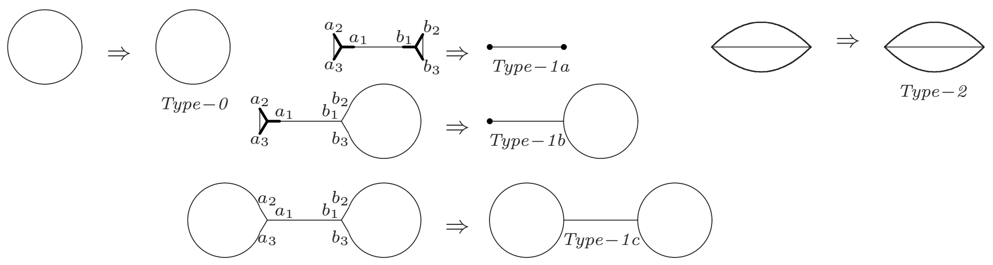

FIG. 3.2. *Transformations from augmentors to dual augmentors.*

S^* for edges of S, not S_{extra}. Because of this, we simply have no loop at all, and we associate to A a dual augmentor of Type-1a or Type-1b (depending on whether this special case occurs at both (a_1, a_2, a_3) and (b_1, b_2, b_3) or just one of them). Notice that only nodes that are in M can have degree 1 in such a set of edges, so these are indeed dual augmentors as defined in Definition 3.1.

Conversely, a dual augmentor A^* of Type 1a, 1b, or 1c as shown in Figure 3.2 corresponds to an augmentor of Type-1 in S_{aug}. By using the same arguments as above for Type-0 augmentors, we know that A^* corresponds to an M-alternating 2-factor in S_{aug}, except that now this 2-factor will have two 3D edges: one for each node of degree 1 or 3 in A^*. If v_e is a node of A^* with degree 3, then edge e is in the corresponding augmentor. If v_e has degree 1 in A^*, then both $e = (u, w, z)$ and an edge (u, w) of S_{extra} (where z is the node of e that is incident on the path of S_{aug} that corresponds to an edge of A^*) are in the corresponding augmentor. Notice that in the latter case, this still results in an M-alternating 2-factor, since $e \in M$ because v_e has degree 1 in A^*, and so both u and w have exactly one edge of M incident to them (the edge e) and exactly one edge not in M incident to them (the edge $(u, w) \in S_{extra}$).

Type-2. By similar reasoning, the set of edges in S^* corresponding to a Type-2 augmentor of S_{aug} is a Type-2 dual augmentor shown in Figure 3.2, with only the degree 3 nodes possibly being in M. Conversely, a dual augmentor of Type-2 with two nodes of degree 3 corresponds to a Type-2 augmentor of S_{aug} with two 3D edges.

The above discussion establishes that every augmentor of S_{aug} corresponds to a dual augmentor of S^*. The converse also holds, since the structures in Figure 3.2 are the only possible structures of dual augmentors as defined by Definition 3.1, and each such structure corresponds to an augmentor of S_{aug}. Therefore, the following lemma holds.

LEMMA 3.2. *There is a one-to-one correspondence between augmentors in S_{aug} and dual augmentors in the cubic graph S^*.*

In other words, dual augmentors are the structures in Figure 3.2, with the nodes in M being only of degree 1 or 3. The reason for considering the dual graph S^* instead of S is that we now have a cubic (i.e., 3-regular) graph and, as the lemma below will show, our goal now will be to cover this cubic graph with dual augmentors. While the same results can be proven directly for S_{aug} instead of S^*, their statements become a lot more messy and complicated.

We now proceed to argue that there always exists an augmentor with high potential for which we need the concept of *augmentor sum*. Let $\mathcal{A}^*$ be the set of all possible dual augmentors with respect to M contained in S^*.

DEFINITION 3.3. *We call a function $\alpha : \mathcal{A}^* \to \mathbb{N}$ a* valid augmentor sum *of S^* with respect to M iff $\exists x > 0$ such that for all edges e of S^*, we have that*

$\sum_{A \in \mathcal{A}^*, A \ni e} \alpha(A) = x$.

In other words, a valid augmentor sum is a cover of S^* with dual augmentors so that every edge is contained in exactly the same number of elements (which we call the *cover number*). Given that there is a one-to-one correspondence between augmentors in S_{aug} and dual augmentors in S^*, we can also view α as a weight assignment on the augmentors in S_{aug}. Figure 3.1 shows a set of dual augmentors that form a valid augmentor sum by covering every edge of S^* twice. It also shows the augmentors of S_{aug} they correspond to. The lemma below shows that if $\phi_M(S) > 0$, then the same must be true for at least one of the augmentors in that list. The idea behind it is that if all augmentors corresponding to the dual augmentors in α "add up" to S and S is improving, then so is some augmentor in the sum. By $|S|$, we will mean the number of nodes incident to edges of S.

LEMMA 3.4. *Given a perfect matching M and an M-alternating 2-factor S with $\phi_M(S) > 0$, there exists an augmentor A that is a subset of S_{aug} with $\phi_M(A) \geq \frac{\phi_M(S)}{|S|}$ if there exists a valid augmentor sum α of S^*.*

Proof. If we had a cover of S by augmentors, such that every edge of S is contained in the same number of augmentors, then we immediately know that some augmentor must have positive potential. This follows because the total potential of the augmentors must equal a multiple of $\phi_M(S)$. Unfortunately, we have such a cover of S^*, but not S. As shown in Figure 3.1, dual augmentors of S^* can correspond to augmentors that include edges in S_{extra}, but not S. In fact, there are some 3D edges of S in Figure 3.1 that are not contained in any augmentors from the list, even though this list forms a valid augmentor sum of S^* (with cover number of 2). Notice, however, that the edges of S_{extra} corresponding to these 3D edges *are* included in the list of augmentors, which we are able to relate to the cost of the 3D edges using the Simplex Condition.

Let x be the cover number of $\alpha : \mathcal{A}^* \rightarrow \mathbb{N}$, and let $\mathcal{A}$ be the set of augmentors in S_{aug}. Since there is a one-to-one correspondence between $\mathcal{A}^*$ and $\mathcal{A}$, we will consider α as an integer weight assignment to augmentors in S_{aug}. First we will compute how many times α covers an edge in $S \subseteq S_{aug}$.

1. If $e = (u, v) \in S$, then $\sum_{A \in \mathcal{A}, A \ni e} \alpha(A) = x$.
   Proof. A 2D edge e of S is part of some path P of 2D edges in S that connects two 3D edges. In S^*, this path P corresponds to a single edge, which we denote by f. This edge f of S^* is covered exactly x times, which means that every edge is P is covered exactly x times, since the dual augmentors that contain f correspond exactly to augmentors containing P.
2. If $e = (u, v, w) \in S$ such that some $(u, v) \in S$, then $\sum_{A \in \mathcal{A}, A \ni e} \alpha(A) = x$.
   Proof. This implies that the node v_e corresponding to e in S^* has a self-loop. By the definition of dual augmentors, all dual augmentors in a valid augmentor sum that contain v_e must also contain the self-loop, which is covered x times, so e is covered exactly x times. The proof of this is as follows.
   Let f be the self-loop incident to v_e, and f' be the other edge incident to v_e. If a dual augmentor contains v_e but not the self-loop incident to v_e, then node v_e must have degree 1 in this dual augmentor, and thus $e \in M$. Since all edges of S^* are covered x times, and there is a dual augmentor that contains f' but not f, then there must be a dual augmentor containing f but not f'. In this dual augmentor, however, v_e would have degree 2, which is a contradiction since the definition of dual augmentors states that all nodes of degree 2 are not in M. Therefore, every dual augmentor containing v_e also contains the self-loop f.

3. If $e = (u, v, w) \in M \cap S$ such that $e_1 = (u, v)$, $e_2 = (v, w)$, $e_3 = (u, w) \notin S$, then for every $i = 1, 2, 3$, we have that $\sum_{A \in \mathcal{A}, A \ni e} \alpha(A) - 2 \sum_{A \in \mathcal{A}, A \ni e_i} \alpha(A) = x$.
   Proof. Consider the middle rightmost 3D edge of Figure 3.1. It appears in several augmentors, with different corresponding edges from S_{extra}. What the above statement implies is that each of these edges in S_{extra} is covered the same number of times y, and that e is covered exactly $x + 2y$ times.
   Let v_e be the node in S^* corresponding to e. Each edge incident to v_e in S^* appears in dual augmentors exactly x times, and each dual augmentor that includes v_e either contains all three edges incident to v_e or exactly one of them. This is because by Definition 3.1, the degree of a node of M in a dual augmentor is 1 or 3. Let a be the number of dual augmentors in α containing all 3 edges incident to v_e. Then all remaining dual augmentors that contain v_e must contain only one of the edges incident to v_e, since v_e must have degree 1 in these dual augmentors. Therefore, there must be $x - a$ dual augmentors containing each edge incident to v_e. Every dual augmentor like this corresponds to an augmentor of S_{aug} that contains the edge e and also contains exactly one of e_1, e_2, or e_3 in S_{extra}. Therefore, for $i = 1, 2, 3$, we know that e_i is contained in exactly $x - a$ augmentors. By the above argument, the node v_e is contained in exactly $a + 3(x - a)$ dual augmentors. Each of the augmentors corresponding to these contains e in S_{aug}, and so edge e is covered $a + 3(x - a)$ times, and each e_i is covered $x - a$ times. This gives us the desired result, since $a + 3(x - a) - 2(x - a) = x$.
4. If $e = (u, v, w) \in S - M$ such that $e_1 = (u, v)$, $e_2 = (v, w)$, $e_3 = (u, w) \notin S$, then for every $i = 1, 2, 3$, we have that $\sum_{A \in \mathcal{A}, A \ni e} \alpha(A) + 2 \sum_{A \in \mathcal{A}, A \ni e_i} \alpha(A) = x$.
   Proof. Let v_e be the node in S^* corresponding to e. Each edge incident to v_e in S^* appears in dual augmentors exactly x times, and each dual augmentor that includes v_e contains either all three edges incident to v_e or exactly two of them. This is because by Definition 3.1, the degree of a node not in M in a dual augmentor can only be 2 or 3. Let a be the number of dual augmentors in α containing all 3 edges incident to v_e. These correspond exactly to the augmentors containing e in S_{aug}. The rest of the dual augmentors that cover edges adjacent to v_e must cover each edge exactly $x - a$ times. Since there are 3 edges incident to v_e, and v_e has degree 2 in each of these dual augmentors, then we know that the set of these dual augmentors can be partitioned into $(x - a)/2$ triples such that each dual augmentor in the triple contains two different edges incident to v_e, so that the entire triple covers each edge twice. A corresponding triple of augmentors in S_{aug} contains each of e_1, e_2, and e_3 once. Therefore, edge e is contained in a augmentors, and each edge e_i is contained in $(x - a)/2$, producing the result.

Using these covering results we now bound $x\ \phi_M(S) = x \sum_{e \in S} \phi_M(e)$.

If $e = (u, v)$, or $e = (u, v, w) \in S$ with some $(u, v) \in S$, we have $x\ \phi_M(e) = \sum_{A \in \mathcal{A}, A \ni e} \alpha(A)\phi_M(e)$. If $e = (u, v, w) \in M \cap S$ such that $e_1 = (u, v)$, $e_2 = (v, w)$, $e_3 = (u, w) \notin S$, then $\phi_M(e) = c(e)$ and $\phi_M(e_i) = -c(e_i)$, so

$$x\ \phi_M(e) = \sum_{A \in \mathcal{A}, A \ni e} \alpha(A)\phi_M(e) - 2 \sum_{A \in \mathcal{A}, A \ni e_i} \alpha(A)\phi_M(e) \tag{3.1}$$

$$\leq \sum_{A \in \mathcal{A}, A \ni e} \alpha(A)\phi_M(e) - \sum_{A \in \mathcal{A}, A \ni e_i} \alpha(A)(c(e_1) + c(e_2) + c(e_3)) \tag{3.2}$$

$$= \sum_{A\in\mathcal{A},A\ni e} \alpha(A)\phi_M(e) + \sum_{A\in\mathcal{A},A\ni e_i} \alpha(A)(\phi_M(e_1)+\phi_M(e_2)+\phi_M(e_3)) \tag{3.3}$$

$$= \sum_{A\in\mathcal{A},A\ni e} \alpha(A)\phi_M(e) + \sum_{A\in\mathcal{A},A\ni e_1} \alpha(A)\phi_M(e_1) + \sum_{A\in\mathcal{A},A\ni e_2} \alpha(A)\phi_M(e_2) + \sum_{A\in\mathcal{A},A\ni e_3} \alpha(A)\phi_M(e_3). \tag{3.4}$$

The inequality holds because of the Simplex Condition on e and the last part holds because $\sum_{A\in\mathcal{A},A\ni e_i}\alpha(A)$ is the same for all $i=1,2,3$.

Similarly, if $e=(u,v,w)\in S-M$ such that $e_1=(u,v)$, $e_2=(v,w)$, $e_3=(u,w)\notin S$, then $\phi_M(e)=-c(e)$ and $\phi_M(e_i)=-c(e_i)$, and so

$$x\ \phi_M(e) = \sum_{A\in\mathcal{A},A\ni e} \alpha(A)\phi_M(e) + 2\sum_{A\in\mathcal{A},A\ni e_i} \alpha(A)\phi_M(e) \tag{3.5}$$

$$\leq \sum_{A\in\mathcal{A},A\ni e} \alpha(A)\phi_M(e) - \sum_{A\in\mathcal{A},A\ni e_i} \alpha(A)(c(e_1)+c(e_2)+c(e_3)) \tag{3.6}$$

$$= \sum_{A\in\mathcal{A},A\ni e} \alpha(A)\phi_M(e) + \sum_{A\in\mathcal{A},A\ni e_1} \alpha(A)\phi_M(e_1) + \sum_{A\in\mathcal{A},A\ni e_2} \alpha(A)\phi_M(e_2) + \sum_{A\in\mathcal{A},A\ni e_3} \alpha(A)\phi_M(e_3). \tag{3.7}$$

Therefore, we have that

$$x\ \phi_M(S) \leq \sum_{A\in\mathcal{A}}\sum_{e\in A}\alpha(A)\phi_M(e) = \sum_{A\in\mathcal{A}}\alpha(A)\phi_M(A).$$

Recall that our goal is to show that if S has high potential, then so does some augmentor. If for all $A\in\mathcal{A}$, $\phi_M(A) < \frac{\phi_M(S)}{|S|}$, then by the above inequality $x\ \phi_M(S) < \frac{\phi_M(S)}{|S|}\sum_{A\in\mathcal{A}}\alpha(A)$. Now consider the dual augmentors corresponding to these augmentors A. Every edge in S^* is covered exactly x times, and every dual augmentor must contain at least one edge; hence $\sum_{A\in\mathcal{A}}\alpha(A)\leq x|E(S^*)|$. By definition of S^* we know that $|E(S^*)|\leq|S|$. Overall, this implies that $x\ \phi_M(S) < \frac{\phi_M(S)}{|S|}x|S| = x\ \phi_M(S)$, giving us a contradiction. This means that there must exist at least one augmentor A with $\phi_M(A)\geq\frac{\phi_M(S)}{|S|}$, as desired. □

Given a perfect matching M that is not min-cost, we know there exists an M-alternating 2-factor S with $\phi_M(S)>0$, since the symmetric difference of M with the min-cost perfect matching is such a 2-factor. By Lemma 3.4, it is enough to show that S^* has a valid augmentor sum in order to prove that our algorithm finds a min-cost perfect matching. The following theorem completes the correctness proof.

THEOREM 3.5. *Any cubic multigraph S^* (possibly with self-loops) has a valid augmentor sum α with respect to any set of nodes M.*

4. Valid augmentor sums (proof of Theorem 3.5).

In this section we set aside our algorithm and Simplex Matching and concentrate on proving Theorem 3.5. We assume that we are given an arbitrary cubic multigraph S^* that may contain self-loops, and some set M of nodes in S^*. We show that there always exists a valid augmentor sum of S^* with respect to M. In other words, we show that we can always cover any cubic graph with dual augmentors so that each edge appears in the same number of these objects.

To understand when such covers may exist, consider the special case when $M = \emptyset$. In [37], Seymour states that we can cover any cubic 2-edge-connected graph with cycles so that every edge is in the same number of cycles. Since $M = \emptyset$, all cycles are dual augmentors (they satisfy the conditions of Definition 3.1), and so we always have a valid augmentor sum. There has been much work in finding cycle covers with small cover numbers [24, 42], and it is unknown whether there always exists a cycle cover of a cubic 2-edge-connected graph with cover number 2 (this is the Cycle Double Cover Conjecture). Since we are proving only an existence result, however, for our purposes the cover number does not need to be small.

The fact that M may not be empty complicates things. For example, while forming a cycle cover of a planar graph is easy, consider forming an augmentor sum of Figure 4.5, or of the complete graph with 4 nodes and $|M| = 1$. Definition 3.1 puts degree constraints on nodes of M, which makes augmentor sums much more difficult to deal with than cycle covers. Our hope is that these results will lead to more covering results where nodes have general degree constraints.

The main idea of the proof of Theorem 3.5 involves constructing a valid augmentor sum for S^* from valid augmentor sums on smaller subgraphs. We will prove Theorem 3.5 with a series of lemmas that show the existence of a valid augmentor sum of S^* in a different way depending on the structure of S^*. Specifically, the proof of Theorem 3.5 is as follows.

Proof of Theorem 3.5. We prove this by induction on the number of nodes in S^*. For the base case, the smallest cubic multigraph consists of two nodes u, v, in which case the entire graph is a dual augmentor of either Type-1 or Type-2.

Now assume that all cubic multigraphs smaller than S^* have a valid augmentor sum. We have the following cases:

- If S^* is not connected, then we can consider each connected component separately and use the inductive hypothesis.
- If S^* has a cut of size 1, i.e., contains a bridge (an edge whose removal disconnects S^*), then by Lemma 4.5, S^* has a valid augmentor sum.
- If S^* does not contain a bridge, but has a cut of size 2, then by Lemma 4.7, S^* has a valid augmentor sum.
- If S^* is 3-edge-connected and $|M| \neq 1$, then by Lemma 4.3, S^* has a valid augmentor sum.
- If S^* is 3-edge-connected and $|M| = 1$, then by Lemma 4.4, S^* has a valid augmentor sum.

Since these are all the possible cases, this finishes the proof of the theorem. $\square$

4.1. Composition lemmas. To prove Theorem 3.5, we now need to show a series of composition lemmas that allow us to construct valid augmentor sums from augmentor sums of smaller graphs. We begin with the following easy lemma about augmentor sums.

LEMMA 4.1. *Let $\mathcal{B}$ be a collection of edge subsets of S^*, and let $\beta : \mathcal{B} \to \mathbb{N}$ and x_β be such that for all edges e in S^*, $\sum_{B \in \mathcal{B}, B \ni e} \beta(B) = x_\beta$. If every $B \in \mathcal{B}$ has a valid augmentor sum, then S^* has a valid augmentor sum.*

Proof. Notice that by Definition 3.1, any dual augmentor of $B \subseteq S^*$ is also a dual augmentor of S^*. Let α_B be the augmentor sum for $B \in \mathcal{B}$, and let x_B be the cover number of α_B. Let x be the least common multiple of all x_B's and $\alpha'_B = \frac{x}{x_B}\alpha_B$, so α'_B is a valid augmentor sum of B with cover number x. α'_B is defined only for dual augmentors in B, but we can extend it to the set of all dual augmentors $\mathcal{A}^*$ in S^* by setting $\alpha'_B(A) = 0$ for any dual augmentor A of S^* not contained in B.

Define $\alpha(A) = \sum_{B \in \mathcal{B}} \beta(B)\alpha'_B(A)$. Then for all edges e in S^*

$$\sum_{A \in \mathcal{A}, A \ni e} \alpha(A) = \sum_{A \in \mathcal{A}, A \ni e} \sum_{B \in \mathcal{B}} \beta(B)\alpha'_B(A) \tag{4.1}$$

$$= \sum_{B \in \mathcal{B}} \beta(B) \sum_{A \in \mathcal{A}, A \ni e} \alpha'_B(A) \tag{4.2}$$

$$= \sum_{B \in \mathcal{B}, B \ni e} \beta(B)\, x = x_\beta x. \tag{4.3}$$

Since every edge of S^* appears in exactly $x_\beta x$ dual augmentors, α is a valid augmentor sum of S^*. $\square$

We will also make use of the following cycle cover theorem due to Seymour [37]. It gives a sufficient condition for the existence of a valid cycle sum, which we define in the same way as a valid augmentor sum.

THEOREM 4.2 (see [37]). *We are given a graph G with capacities $cap(e)$. Let $\mathcal{C}(G)$ be the collection of all cycles in G and a* valid circuit sum *of G be a function $\beta : \mathcal{C}(G) \to \mathbb{Q}^+$ such that for every edge e of G, $\sum_{C \in \mathcal{C}(G), C \ni e} \beta(C) = cap(e)$. Then, a valid circuit sum exists if for every cut K, we have that for all $e \in K$, $cap(e) \leq \sum_{e' \in K-e} cap(e')$.*

First, we address a special case.

LEMMA 4.3. *If S^* is 3-edge-connected and $|M| \neq 1$, then there exists a valid augmentor sum of S^*. Moreover, if $|M| = 0$, then the only dual augmentors in this sum are of Type-0 (i.e., cycles).*

Proof. Construct a new graph G by adding an extra node s to S^*, together with an edge (s, v) for all $v \in M$ (see Figure 4.1). Associate a capacity $cap(e)$ with all edges e of G. If e is one of the new edges (s, v), set $cap(e) = 3$; otherwise set $cap(e) = 1$. By Theorem 4.2, we know that there exists a valid circuit sum of G if for all cuts K of G and all edges $e \in K$:

$$cap(e) \leq \sum_{e' \in K-e} cap(e'). \tag{4.4}$$

We now show that this holds true for G.

Consider an arbitrary cut K of G. If K is the cut $(s, G - s)$, then inequality (4.4) is satisfied since all edges in the cut have the same capacity (there cannot be exactly one edge in the cut, since $|M| \neq 1$). Any other cut K of G contains at least three edges since S^* is 3-edge-connected. If all edges $e \in K$ are such that $cap(e) = 1$, then the above inequality is trivially satisfied (this also finishes the case when $|M| = 0$). Otherwise, if $e = (s, v)$ and K is not $(s, G - s)$, then $K - e$ is also a cut in S^*, and it contains at least three edges of capacity 1 since S^* is 3-edge-connected. Inequality (4.4) holds since $cap(e) = 3$ and there are at least three edges in K with capacity 1. By Theorem 4.2, we can therefore form a circuit sum of the graph G.

To form dual augmentors from these cycles, we need to break up all the cycles containing a node of M, since nodes of M cannot have degree 2 in a dual augmentor. We can do this since all such cycles contain at least two nodes of M. Specifically, we will partition each cycle C with $\beta(C) > 0$ into dual augmentors as follows. Every cycle that does not contain any nodes in M is contained in S^* and is trivially a Type-0 dual augmentor. No cycle passes through two edges with capacity 1 incident to a node $v \in M$, since all other cycles passing through v would not be able to fill (s, v) to its capacity. Therefore, every cycle entering a node in M must proceed to s. Removing

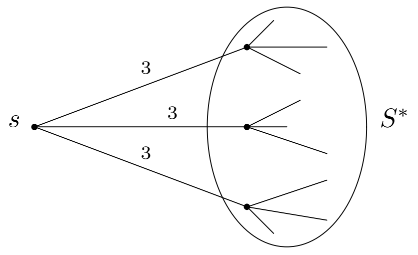

FIG. 4.1. *Capacitated graph G in Lemma 4.3.*

all (s, v) edges from these cycles gives us a collection of Type-1a dual augmentors in S^*. The valid circuit sum β can be viewed as a fractional weight assignment on those Type-1a and Type-0 dual augmentors such that every edge in S^* is covered exactly once. We can now multiply β by a large enough constant to form a valid augmentor sum. $\square$

Section 4.2 addresses the special (and tricky) case when only a single node of S^* is in M and proves the following lemma.

LEMMA 4.4. *If S^* is 3-edge-connected, and $|M| = 1$, then there exists a valid augmentor sum of S^*. Moreover, every dual augmentor in this sum that contains the node in M also contains all edges incident to this node.*

Hence, all that is left to show are Lemmas 4.5 and 4.7, which provide us with augmentor sums for the cases when S^* has a bridge or is 2-edge-connected.

LEMMA 4.5. *Assume that all cubic multigraphs smaller than S^* have valid augmentor sums and that S^* contains a bridge.*[1] *Then S^* has a valid augmentor sum.*

Proof. Let $e = (u_1, v_1)$ be a bridge in S^*. Let the other two edges incident to u_1 be (u_1, u_2) and (u_1, u_3), and let the other two edges incident to v_1 be (v_1, v_2) and (v_1, v_3). Form two smaller cubic multigraphs S_1 and S_2 by removing nodes u_1 and v_1 together with their incident edges and adding two new edges (u_2, u_3) and (v_2, v_3), as in Figure 4.2. In the case where $(u_1, u_2) = (u_1, u_3)$ (u_1 has a self-loop), we delete u_1 and the edge e but keep the loop, resulting in a loop without any nodes, which is a dual augmentor of Type-0 and corresponds to a cycle with no 3D edges in S.

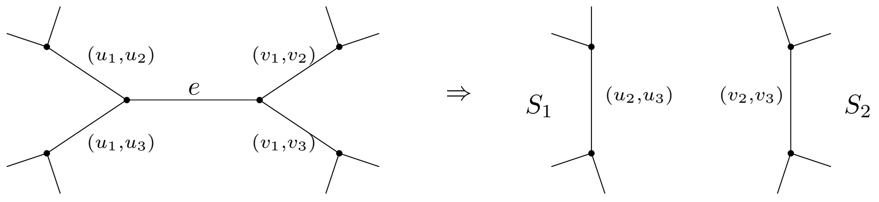

FIG. 4.2. *Breaking S^* with a bridge into two smaller cubic multigraphs.*

By the assumption made in the statement of the lemma, there exist valid augmentor sums α_1 and α_2 of S_1 and S_2, with corresponding cover numbers x_1 and x_2. If x is the least common multiple of x_1 and x_2, then $\frac{x}{x_1}\alpha_1$ and $\frac{x}{x_2}\alpha_2$ are valid augmentor sums of S_1 and S_2 with cover number x. This gives us a multiset of size x of dual augmentors in S_1 that contain (u_2, u_3) and another multiset of size x of

[1] Recall that a bridge is an edge whose removal disconnects the graph.

dual augmentors in S_2 that contain (v_2, v_3). Form a matching of the dual augmentors of the S_1 multiset with the dual augmentors of the S_2 multiset, and let (A_1, A_2) be one such pair in the matching. Consider the multigraph $C(A_1, A_2)$ constructed from $A_1 \cup A_2$ by adding nodes u_1 and v_1, replacing the edges (u_2, u_3) and (v_2, v_3) in A_1 and A_2 with edges (u_1, u_2), (u_1, u_3), (v_1, v_2), and (v_1, v_3), and then adding the edge $e = (u_1, v_1)$. As we will prove below in Lemma 4.6, $C(A_1, A_2)$ has a valid augmentor sum for all pairs (A_1, A_2). This is enough to prove our lemma, since we can then use Lemma 4.1 to deduce that the entire graph S^* has a valid augmentor sum. Specifically, we define a collection $\mathcal{B}$ of edge subsets of S^* as the set of dual augmentors in the valid augmentor sums of S_1 and S_2 with cover number x, except that for every pair (A_1, A_2) as above, we replace A_1 and A_2 with $C(A_1, A_2)$. This creates exactly x sets containing each edge $e = (u_1, v_1)$, (u_1, u_2), (u_1, u_3), (v_1, v_2), and (v_1, v_3). Since every edge e' of S^* is contained in exactly x sets of $\mathcal{B}$, we can apply Lemma 4.1 to prove that S^* has a valid augmentor sum if all sets $C(A_1, A_2)$ have valid augmentor sums. ▯

LEMMA 4.6. *Let A_1 and A_2 be node-disjoint dual augmentors, and let $C = C(A_1, A_2)$ be a multigraph created from A_1 and A_2 as described in the proof of Lemma 4.5. Then C has a valid augmentor sum.*

Proof. Rather than providing a valid augmentor sum individually for all possible graphs C resulting from the pairing of the different types of dual augmentors, we give three simple rules that reduce most such possible graphs to trivial cases. Each of these rules decomposes C into smaller graphs, such that if all these smaller graphs have valid augmentor sums, then by Lemma 4.1 C also has a valid augmentor sum. We will call a structure *reducible* if at least one of these three rules can be applied to it, and *irreducible* otherwise. To prove that C has a valid augmentor sum, we will recursively apply these three rules to C until we obtain irreducible structures and then prove that these irreducible structures have valid augmentor sums, which implies that C must also have a valid augmentor sum by Lemma 4.1. Figure 4.3 illustrates these rules (for a detailed description see below).

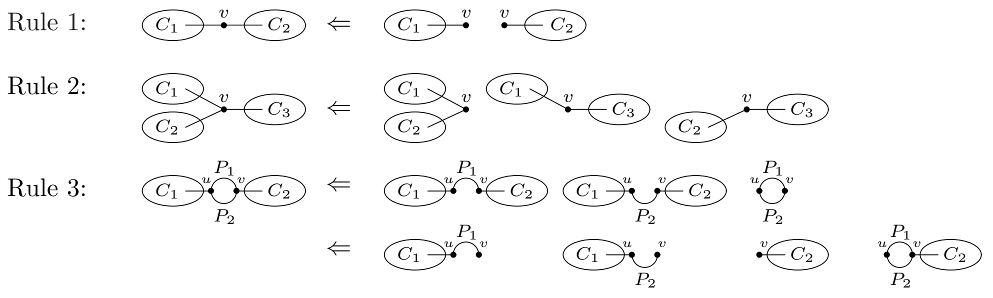

FIG. 4.3. *Rules for decomposing graphs into easier cases to prove the existence of augmentor sums.*

Rule 1. Suppose C contains a vertex $v \in M$ of degree 2, the removal of which disconnects C into components C_1 and C_2. If $C - C_1$ and $C - C_2$ have valid augmentor sums, then by Lemma 4.1 C also has a valid augmentor sum, as the edge sets of $C - C_1$ and $C - C_2$ cover every edge of C exactly once. We can also apply a similar rule if v is of degree 3, and removing it disconnects C into 3 connected components.

Rule 2. Suppose C contains a vertex $v \notin M$ of degree 3, the removal of which disconnects C into components C_1, C_2, and C_3. As above, if $C - C_1$, $C - C_2$, and $C - C_3$ have valid augmentor sums, then so does C, as these sets cover every edge of

C exactly twice.

Rule 3. Suppose C contains two vertices u and v of degree 3, the removal of which partitions C into components C_1, C_2, P_1, and P_2 as in Figure 4.3, where P_1 and P_2 are paths. As above, if $C - P_1$, $C - P_2$, and $C - C_1 - C_2$ have valid augmentor sums, then so does C, since those graphs cover every edge of C twice. Similarly, if $C - P_1 - C_2$, $C - P_2 - C_2$, $C - C_1 - P_1 - P_2$, and $C - C_1$ have valid augmentor sums, then so does C. We apply the first variant of Rule 3 when either $u, v \in M$ or $u, v \notin M$, since it is then that the cycle $C - C_1 - C_2$ has a valid augmentor sum (if $u, v \in M$, then it is a sum of two dual augmentors, with u and v having degree 1 in each). We apply the second variant when $u \notin M$ and $v \in M$, since then v always has degree 1 or 3, as needed by Definition 3.1.

To illustrate the use of these rules, consider the following example. Let A_1 be a dual augmentor of Type-1c, A_2 be a dual augmentor of Type-2, and $u \notin M$ lie on one of the cycles of A_1. Moreover, suppose that the node of degree 3 on the same cycle of A_1 is in M. Then we would apply the second version of Rule 3, as shown in Figure 4.4. The first two resulting structures are dual augmentors (of Type-1b and Type-1c), and so they each have a valid augmentor sum. If we can show that the last two resulting structures also have valid augmentor sums, then we know that C also has a valid augmentor sum, since Rule 3 is such that the resulting structures contain each edge of C exactly the same number of times. At this point, we can apply any of the three rules to the resulting structures in order to decompose them further.

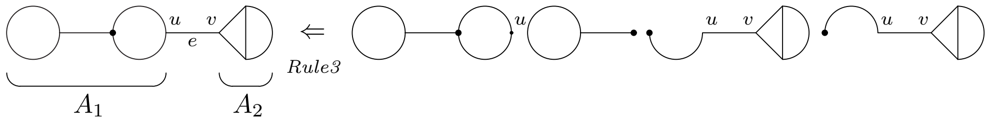

FIG. 4.4. *Sample reduction for Lemma* 4.5.

Graphs like the last one in Figure 4.4, however, and graphs of similar structure are irreducible and yet are not dual augmentors. For these types of graphs we show a valid augmentor sum directly in Figure 4.5. Notice that if the leftmost node of M is substituted by a cycle, the same augmentor sum is valid.

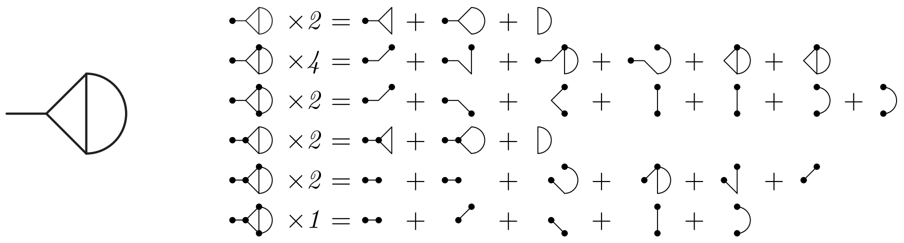

FIG. 4.5. *Proof that the last remaining case of Lemma* 4.5 *has a valid augmentor sum. Small circles indicate nodes of M, so this shows all possible cases. Notice that structures on the right are dual augmentors because they are objects from Figure* 3.2 *and nodes of M have degree* 1 *or* 3.

It is easy to check that no matter what two structures A_1 and A_2 from Figure 3.2 are patched together to form $C = C(A_1, A_2)$, we can always decompose them via repeated application of the above three rules into irreducible structures which are either dual augmentors (and so trivially have a valid augmentor sum) or a structure in Figure 4.5, for which we explicitly gave a valid augmentor sum. For completeness, the full description of the types of structures resulting from patching together dual augmentors, and the order of applying the three rules to those structures, can be found in Table A.1 in the appendix. These irreducible structures all have valid augmentor sums, and since we obtained them through repeated applications of our three rules, we know that we can use these structures to cover every edge of C exactly the same number of times. By Lemma 3.4, this implies that C has a valid augmentor sum. ∎

The only case left now is if S^* is bridgeless, but not 3-edge-connected, which is addressed below.

LEMMA 4.7. *Assume that all cubic multigraphs smaller than S^* have a valid augmentor sum and that S^* is bridgeless but contains two edges e_1 and e_2, the removal of which disconnects it. Then S^* has a valid augmentor sum.*

Proof. The proof of this lemma is similar to Lemma 4.5: we decompose S^* into two smaller cubic graphs S_1 and S_2, and form augmentor sums of each. Then we show that after patching together a dual augmentor of S_1 with a dual augmentor of S_2, we get an object that itself has a valid augmentor sum. Note that the "patching" process here is different from Lemma 4.5.

We assumed that S^* is bridgeless. Let $e_1 = (u_1, v_1)$ and $e_2 = (u_2, v_2)$, so that u_1, u_2 are in the same connected component of $S^* - e_1 - e_2$. We proceed as in the proof of Lemma 4.5 to form two smaller cubic multigraphs S_1 and S_2 by removing e_1 and e_2, and forming two new edges (u_1, u_2) and (v_1, v_2), as in Figure 4.6. Notice that the four nodes u_1, u_2, v_1, and v_2 must be distinct, since if $u_1 = u_2$, then the third edge incident to u_1 would be a bridge.

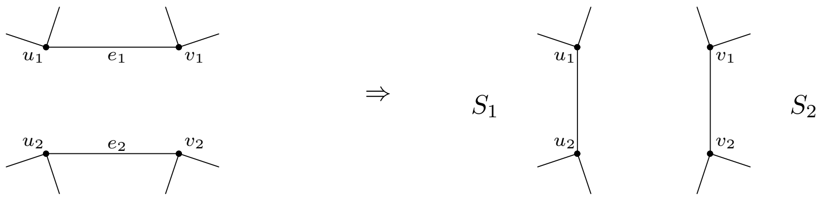

FIG. 4.6. *Breaking a 2-edge-connected bridgeless S^* into two smaller cubic multigraphs.*

Similar to Lemma 4.5, there exist valid augmentor sums for S_1 and S_2 with cover number x. This means that there exists a multiset of size x of dual augmentors in S_1 that contain (u_1, u_2) and another multiset of size x of dual augmentors in S_2 that contain (v_1, v_2). Pair up the dual augmentors of the S_1 multiset with the dual augmentors in the S_2 multiset, and let (A_1, A_2) be one such pair. Consider the cubic multigraph C resulting from removing $(u_1, u_2), (v_1, v_2)$ and adding e_1 and e_2 to A_1 and A_2. We will now show that C has a valid augmentor sum, regardless of the types of A_1 and A_2, which finishes the proof using Lemma 4.1.

Figure 4.7 shows what each dual augmentor might look like once the edge (u_1, u_2) (similarly (v_1, v_2)) is removed. The graph C is simply the joining of two of these objects along the two "connector" vertices (marked with empty nodes).

It is easy to see that all possible ways to join these objects to form C have already been proven to have a valid augmentor sum in Lemma 4.6, with the exception of the graph formed when the two dual augmentors A_1 and A_2 are of Type-2. Figure 4.8 shows the valid augmentor sum for this graph, which completes the proof of the lemma. For completeness, see Table A.2 for all possible ways to join together two objects from Figure 4.7. All of the resulting objects have valid augmentor sums, as

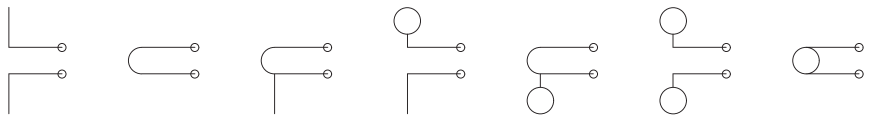

FIG. 4.7. *Dual augmentors with an edge removed. From left to right these are Type-*1a*, Type-*0*, two versions of Type-*1b *(depending which edge is removed), two versions of Type-*1c*, and Type-*2.

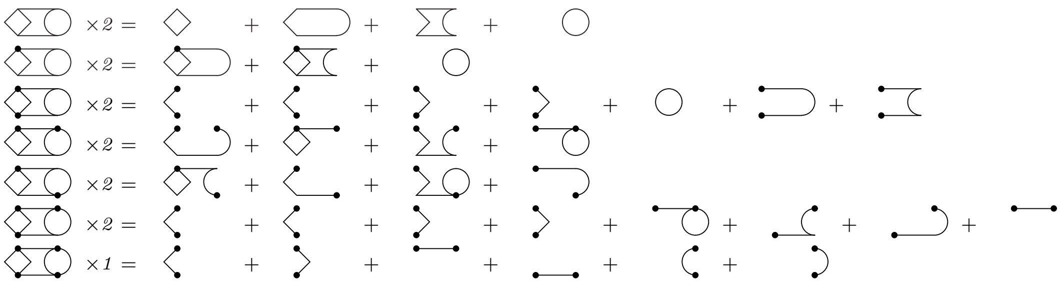

FIG. 4.8. *Proof that the last remaining case of Lemma* 4.7 *has a valid augmentor sum. Small circles indicate nodes of M, so this shows all possible cases. The structures on the right are dual augmentors because they are objects from Figure* 3.2 *and nodes of M have degree* 1 *or* 3.

desired. ▯

4.2. Special case: $|M| = 1$. This section is devoted to the special case where $|M| = 1$ and S^* is 3-edge-connected.

LEMMA 4.4. *If S^* is* 3*-edge-connected, and $|M| = 1$, then there exists a valid augmentor sum of S^*. Moreover, every dual augmentor in this sum that contains the node in M also contain all edges incident to this node.*

Proof. We proceed by induction. The smallest cubic graph with $|M| = 1$ is a graph on two nodes. This graph is either a Type-2 dual augmentor or a Type-1 dual augmentor, and so the lemma holds for this base case.

We now assume that the statement of the lemma holds for all cubic multigraphs smaller than S^*. If there does not exist a cut of size 3 with more than a single node on each side, then the lemma holds for S^* by Lemma 4.8, which is proved below. Therefore, we assume that there exists such a cut of size 3, and prove the inductive step for this case.

This proof is similar to the proofs of Lemmas 4.5 and 4.7. Let the cut of size 3 be $\{e_1, e_2, e_3\}$, and let $M = \{r\}$. Form two new 3-edge-connected cubic multigraphs S_1, S_2 by contracting each side of the cut into a single node, with $r \in S_1$. S_1 is a smaller graph with a single node of M, so by the inductive hypothesis, there exists a valid augmentor sum α_1 of S_1. Let v be the node of S_2 representing the contracted side of the cut, which included the node r. Here we have a choice: do we say whether v is in M or not? Denote the first graph with $v \in M$ as S_2 and the second graph with $v \notin M$ as S_2'. In the first case, S_2 is a graph with a single node of M, which must have some valid augmentor sum α_2, with cover number y, by the inductive hypothesis. In the second case, S_2' would have no nodes in M, so by Lemma 4.3, we also have a valid augmentor sum α_2', with cover number y'.

We can say something more specific about α_2 and α_2'. Since S_2' is a 3-edge-connected graph with no nodes in M, by Lemma 4.3 we can assume that α_2' is a cycle cover (i.e., the only dual augmentors appearing in it with positive weight are cycles). In particular, all dual augmentors containing v in α_2' contain exactly two edges incident to v. As for α_2, we know by the inductive hypothesis that all dual augmentors containing v in α_2 contain all three edges incident to it. Similarly, all dual augmentors containing r in α_1 must contain all edges incident to it.

Now let w be the node of S_1 representing the contracted side of the cut. Let a be the number of times w appears in a dual augmentor of α_1 containing all 3 edges of w, and let b be the number of times w appears in a dual augmentor of α_1 containing exactly two edges of w. Since $w \notin M$, these are the only options, and so the cover number of α_1 is $a + 2b/3$.

The idea is that we are going to attach α_2 to the a dual augmentors above, and α_2' to the other b dual augmentors. To do this, let x be the least common multiple of y and $3y'/2$, and form new augmentor sums $x\alpha_1$, $\frac{xa}{y}\alpha_2$, and $\frac{2xb}{3y'}\alpha_2'$. This means that we now have xb dual augmentors containing exactly two edges of w in S_1, and xb dual augmentors containing exactly two edges of v in S_2', the latter coming from $\frac{2xb}{3y'}\alpha_2'$. Just as in Lemmas 4.5 and 4.7, we can place these dual augmentors into pairs (A_1, A_2). Furthermore, since dual augmentors covering exactly two edges of a node must appear in triples (so that all edges are covered the same number of times), we can make sure that in every pair (A_1, A_2), both A_1 and A_2 use the same two edges from the set $\{e_1, e_2, e_3\}$. We can then form a multigraph C in S^* by patching A_1 and A_2 together; i.e., C consists of edges in S^* corresponding to either A_1 or A_2. Since all dual augmentors in α_2' are cycles, C must be a dual augmentor, since patching a dual augmentor in this manner together with a cycle results in a dual augmentor again.

We now consider the xa dual augmentors containing exactly three edges of w in S_1, and xa dual augmentors containing exactly three edges of v in S_2, the latter coming from $\frac{xa}{y}\alpha_2$. We pair them up in the same way and patch them together to form subgraphs $C = A_1 \cup A_2$ for each pair (A_1, A_2). All structures C that can be formed in this way are dual augmentors.

By taking the above subgraphs C, together with the dual augmentors of α_1, α_2, and α_2' that do not intersect v or w, we form a valid augmentor sum for S^*, as desired. Moreover, notice that $C = A_1 \cup A_2$ contains edges incident to r exactly when A_1 contains them, and so if C contains r, it must also contain all edges incident to r. ☐

The most difficult subcase of Theorem 3.5 is proven in the following lemma, which requires different techniques from most of our other proofs. We prove the cases when S^* is planar and nonplanar separately. For the nonplanar case, we show that there must be some subdivision of $K_{3,3}$ containing the node of M, and from this we form a valid augmentor sum using Theorem 4.2. In the planar case, we use powerful edge-coloring results (see Figure 4.10). The following lemma finishes our proof that every cubic graph has a valid augmentor sum.

LEMMA 4.8. *Assume that S^* does not have a cut of size* 3 *with more than a single node on each side. If S^* is* 3*-edge-connected, and $|M| = 1$, then there exists a valid augmentor sum of S^*. Moreover, every dual augmentor in this sum that contains the node in M also contain all edges incident to this node.*

Proof. Let $M = \{r\}$, the three nodes adjacent to r be v_1, v_2, v_3, and denote the edge (r, v_i) by e_i. First, we address the nonplanar case.

Nonplanar. Assume that S^* is not planar. A *subdivision* of $K_{3,3}$ is a graph with nodes s_1, s_2, s_3 and t_1, t_2, t_3 with a node-disjoint path between every s_i and t_j. We first want to show that there exists a subdivision of $K_{3,3}$ as a subgraph of S^* with r as one of the degree-3 nodes in this $K_{3,3}$, which we do in a separate technical Lemma 4.9.

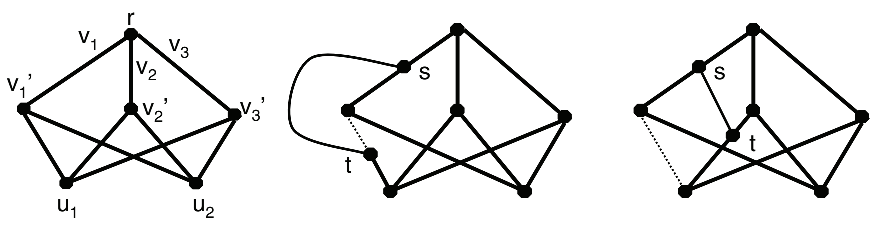

FIG. 4.9. *Proof of Lemma 4.8. (Left) A subdivision of $K_{3,3}$. (Middle) An s-t path with $t \in P(u_1, v'_1)$. (Right) An s-t path with $t \in P(u_1, v'_2)$.*

Let K be this $K_{3,3}$ instance, and let $\sigma(K)$ be the subdivision of it found above. That is, $\sigma(K)$ is a subgraph of S^* with nodes r, u_1, u_2 and v'_1, v'_2, v'_3 and a node-disjoint path between every u_i and v'_j, as well as between r and every v'_j. As pictured in Figure 4.9, we let v'_1, v'_2, v'_3 be the nodes adjacent to r in K, with v'_i appearing at the end of a path in $\sigma(K)$ starting at v_i, and we let u_1, u_2 be the nodes in K that are at distance 2 from r. Define $P(r, v'_j)$ and $P(u_i, v'_j)$ to be the node-disjoint paths described above for $j = 1, 2, 3$ and $i = 1, 2$, and let $\mathcal{P}$ be the collection of these 9 paths. In general, for any u, v contained in the same path of $\mathcal{P}$, define $P(u, v)$ to be the unique path in $\sigma(K)$ that does not go through any degree 3 node except at its endpoints.

Next we want to show that there exists such a $\sigma(K)$ with the length of $P(r, v'_j)$ equal to 1 for all j. Take $\sigma(K)$ as above so that the lengths of $P(r, v'_j)$ are minimal. Suppose that there exists some path P with endpoints $s, t \in \sigma(K)$, disjoint from $\sigma(K)$ except at endpoints, with $s \in P(r, v'_1)$ and $s \neq v'_1$. If $t \in P(v'_1, u_1)$, then we can form a new $K_{3,3}$ subdivision by replacing $P(t, v'_1)$ with P, as in Figure 4.9(Middle). This shortens the length of $P(r, v'_1)$, since s becomes the "new" v'_1, giving a contradiction. If $t \in P(u_1, v'_2)$, we can similarly redesign K by replacing $P(u_1, v'_1)$ with P and making t the "new" u_1, as in Figure 4.9(Right). This also shortens $P(r, v'_1)$, and all other cases with $s \in P(r, v'_j)$ and $t \in P(u_i, v'_k)$ can be reduced to these for any i, j, k. Therefore, we can assume that all paths from nodes in $P(r, v'_j)$ must pass through v'_1, v'_2, v'_3 to reach nodes outside of $\cup_j P(r, v'_j)$. Let e_j be the edge closest to v'_j on the path $P(r, v'_j)$. The above reasoning implies that $\{e_1, e_2, e_3\}$ is a cut in the graph S^*. In the statement of the lemma we assumed that this is impossible unless one side of the cut is a single node. Since distinct nodes u_1 and u_2 are on the same side of this cut, this implies that r is the only node on one side of this cut, and so $|P(r, v'_j)| = 1$ for all j. Thus, from this point on we can assume that $v_j = v'_j$.

Take this $\sigma(K)$, which is really the union (although not a valid sum) of two Type-2 dual augmentors containing r: one with degree 3 at u_1, and one with degree 3 at u_2; both with degree 3 at r. Set the α value of each of these two dual augmentors to 1. To form a valid augmentor sum, it is enough to cover all edges not in $\sigma(K)$ by two dual augmentors, and all edges in $\sigma(K)$ not adjacent to r by one dual augmentor. We do not need to cover the edges adjacent to r anymore, since we already covered them with two dual augmentors, and so we may as well remove them. Call the graph formed by removing r and its incident edges S'. All nodes in this graph have degree 3, except for v_1, v_2, and v_3, which have degree 2.

We now apply Theorem 4.2 to S', with the capacity of an edge being 1 if this edge was in $\sigma(K)$, and 2 otherwise. We know that we can cover this graph with cycles in the desired manner if for every cut C, we have that for all $e \in C$, $cap(e) \leq \sum_{e' \in C-e} cap(e')$. Consider an arbitrary cut C. If all nodes v_1, v_2, and v_3 are on the same side of the cut, this C is also a cut in S^*. Since S^* is 3-edge-connected, C must consist of at least three edges, and since the edge capacities are either 1 or 2, we know that the desired inequality holds. Therefore, we can assume that not all nodes v_j are on the same side of the cut. Without loss of generality, assume that C disconnects v_1 from nodes v_2 and v_3. If the cut C is simply $(v_1, S' - v_1)$, i.e., if v_1 is the only node on one side of the cut, then C consists of two edges that are incident to v_1, both with capacity 1 (since they are both in $\sigma(K)$). Therefore, we once again have that for all $e \in C$, $cap(e) \leq \sum_{e' \in C-e} cap(e')$. The final case is if there is another node $w \neq v_1$ that is on the same side of the cut as v_1. We will show that C must contain at least 3 edges, and thus that the desired inequality holds. Suppose to the contrary that C consists of only two edges. Now consider the cut $C \cup \{(r, v_1)\}$ in S^*. This cut disconnects r from w, since if it did not, then there would be a path from v_2 or v_3 to w in S' that does not use edges of C. This gives us a set of three edges that disconnects the graph S^*. Both sides of this cut contain more than a single node (one side contains at least r, v_2, and v_3, and the other contains at least v_1 and w). By our assumption in the statement of the lemma, this is not possible, giving us a contradiction. Therefore, there exists a circuit sum of this graph, giving us a valid augmentor sum together with the the augmentors in $\sigma(K)$. As in the proof of Lemma 4.3, this is a fractional augmentor sum, since each dual augmentor is assigned a fractional weight, such that the sum of the weights for dual augmentors containing any edge equals 2. To form a valid augmentor sum where each augmentor is used an integral number of times, we can simply multiply this fractional weight assignment by a large enough integer x to form a valid augmentor sum with cover number $2x$. Notice in addition that all dual augmentors in this sum containing r also contain all 3 edges incident to r, as desired.

Planar. We now address the planar case. Tait [40] showed that the Four Color Theorem [5, 35] is equivalent to the following statement: "Every 2-edge-connected cubic planar graph is edge-3-colorable." Take such a coloring of S^*, where each color just forms a perfect matching. Call these matchings M_1, M_2, M_3 and form symmetric differences $M_1 \triangle M_2$, $M_2 \triangle M_3$, and $M_3 \triangle M_1$. Each of these is a set of node-disjoint cycles, with every edge being in exactly two of these. Consider what $M_1 \triangle M_2$ looks like with respect to r, and let C_1 be the cycle of it containing r, and C_2 be the cycle of it containing the node adjacent to r attached by an edge e of M_3. Form a dual augmentor by taking $C_1 \cup C_2 \cup \{e\}$ (note that C_1 may equal C_2). Figure 4.10 shows what this object can look like. To check that this is a dual augmentor, notice that the only nodes of degree 3 are the endpoints of e, since C_1 and C_2 are node-disjoint. The only node of M is r, which has degree 3, as desired.

Now, take all the cycles of $M_1 \triangle M_2, M_2 \triangle M_3, M_3 \triangle M_1$, but replace C_1, C_2 by $C_1 \cup C_2 \cup \{e\}$ (and similarly for $M_2 \triangle M_3$ and $M_3 \triangle M_1$). If we take all of these dual augmentors and cycles, we get a dual augmentor cover that covers the edges next to r three times and all the other edges twice. Remove r and its adjacent edges, forming

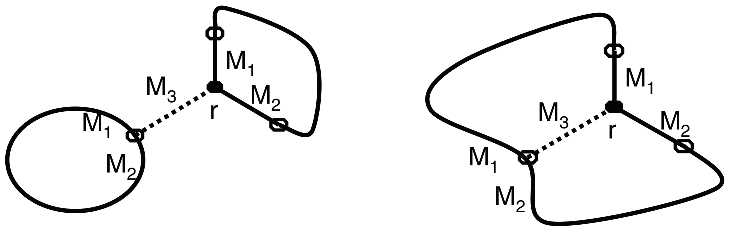

FIG. 4.10. *The planar case in Lemma* 4.8. *(Left)* $C_1 \neq C_2$. *(Right)* $C_1 = C_2$.

a new graph S'. As in the nonplanar case, we can use Theorem 4.2 to show that we can cover the resulting graph with cycles so that every edge appears in exactly the same number x of cycles. Together this gives a valid augmentor sum, since we can multiply the cover above by x, combine it with the cycle cover, and end up with a valid augmentor sum with cover number $3x$. As in the nonplanar case, all dual augmentors containing r also contain the three edges incident to r. ▯

LEMMA 4.9. *If the conditions of Lemma* 4.8 *are met, and* S^* *is nonplanar, then there exists a subdivision of* $K_{3,3}$ *as a subgraph of* S^* *with the node of* M *as one of the degree* 3 *nodes in this* $K_{3,3}$.

Proof. The proof of this lemma is due largely to Paul Seymour. Let $M = \{r\}$, the three nodes adjacent to r be v_1, v_2, v_3, and denote the edge (r, v_i) by e_i. A *subdivision* of $K_{3,3}$ is a graph with nodes s_1, s_2, s_3 and t_1, t_2, t_3 with a node-disjoint path between every s_i and t_j. We want to show that there exists a subdivision of $K_{3,3}$ as a subgraph of S^* with r as one of the degree 3 nodes in this $K_{3,3}$.

To derive that this must hold, we use Theorem 2.4 in [34] by Robertson and Seymour. For completeness, we restate this theorem here, with all the relevant notation from [34]. A *society* is a pair (G, Ω) where G is a graph and Ω is a cyclic permutation of a subset of nodes Ω^* in G. A *separation* of a society (G, Ω) is a pair of subgraphs (G_1, G_2) where $G_1 \cup G_2 = G$, $E(G_1 \cap G_2) = \emptyset$, and $\Omega^* \subseteq V(G_2)$. By $V(G)$ and $E(G)$ we denote the nodes and edges of a graph G, respectively. We say that a separation (G_1, G_2) is a *k-separation* if $|V(G_1 \cap G_2)| \leq k$, and $V(G_2) \neq V(G)$. We say that (G, Ω) is *k-connected* if it has no k'-separation for $k' < k$. Additionally, we define *cross*, *tripod*, and *rural*.

DEFINITION 4.10. *Let* u_1 *and* u_2 *be distinct vertices of* G. *A* tripod *is a set of nodes* $s_1, s_2, s_3 \in V(G)$ *and* $t_1, t_2, t_3 \in \Omega^*$ *together with mutually node-disjoint paths (except at their endpoints) from every* u_i *to* s_j $(i = 1, 2,\ j = 1, 2, 3)$, *and from each* s_j *to* t_j $(j = 1, 2, 3)$. *Thus, a tripod consists of* 9 *disjoint paths.*

DEFINITION 4.11. *A society* (G, Ω) *is* rural *if* G *has a drawing* Γ *in the plane without crossings (so* G *is planar), and there is a closed disc* Δ *in the plane, such that*

(i) *the drawing* Γ *uses no points outside the disc* Δ, *and*
(ii) *for* $v \in V(G)$, *the point of* Γ *representing* v *lies on the boundary of* Δ *if and only if* $v \in \Omega^*$. *Moreover, the points on the boundary appear clockwise in the order* Ω.

We will not define precisely the term *cross* (see [34] for the exact definition), since for our purposes it suffices to point out that a cross cannot exist unless $|\Omega^*| \geq 4$. We are now ready to state Theorem 2.4 from [34].

THEOREM 4.12 (see [34]). *Let (G, Ω) be a 3-connected society with no cross or tripod. Then (G, Ω) is rural.*

We now apply Theorem 4.12 to our graph S^* with a single node r of M. Specifically, delete r and its adjacent edges from S^* to form a new graph S'. Set $\Omega^* = \{v_1, v_2, v_3\}$, and let a permutation Ω of the elements of Ω^* be (v_1, v_2, v_3). To apply Theorem 4.12, we must show that the society (S', Ω) is 3-edge-connected. Suppose to the contrary that there exists a 2-separation of (S', Ω). That is, by definition of 2-separation, we assume that there exist two subgraphs G_1 and G_2 such that $G_1 \cup G_2 = S'$, $E(G_1 \cap G_2) = \emptyset$, $\Omega^* \subseteq V(G_2)$, $|V(G_1 \cap G_2)| \leq 2$, and $V(G_2) \neq V(S')$. Let u_1 and u_2 be the two (possibly not distinct) nodes in $V(G_1 \cap G_2)$, and let w be the node that must exist in $V(S') \backslash V(G_2)$. The above definition of 2-separation simply implies that by removing nodes u_1 and u_2 from S', we disconnect all the remaining nodes of Ω^* from the node w. Consider what this statement means for the original graph S^*. Since r connects exactly to the nodes of Ω^*, it is equivalent to saying that removing nodes u_1 and u_2 from S^* disconnects r from w. Since S^* is cubic, this implies that there is an edge-cut of size 2 that disconnects r from w, which is a contradiction with S^* being 3-edge-connected. Therefore, we have shown that S' is 3-edge-connected as well.

We now apply Theorem 4.12 to the society (S', Ω). We know that (S', Ω) cannot contain a cross, since $|\Omega^*| = 3$. Our goal is to show that (S', Ω) contains a tripod, so to do this we must prove that (S', Ω) is not rural. Suppose to the contrary that (S', Ω) is rural. In this case, we can draw S' in the plane without edge crossings, with the nodes v_1, v_2, v_3 on the outer face. Using this drawing, we could show that S^* is planar, since we could add r to the outer face of the drawing and attach it to nodes v_1, v_2, v_3 without adding edge crossings. Since S^* is not planar, we know that S' is not rural. Therefore, by Theorem 4.12, S' must have a tripod. A tripod combined with r and e_1, e_2, e_3 gives us exactly a subdivision of $K_{3,3}$. ☐

5. Running time.

5. Running time. Here we argue that our algorithm given in section 2 runs in polynomial time. We can find the initial perfect matching using the unweighted version of the Simplex Matching algorithm from [38]. If we are applying this to Terminal Backup or similar problems, then there always exists a perfect matching without 3D edges, so we can find it using traditional matching algorithms.

THEOREM 5.1. *Our algorithm solves* Simplex Matching *with integer costs in polynomial time.*

Proof. Let OPT be the cost of the minimum-cost perfect matching M^*, and let M be some perfect matching. As mentioned before, $M^* \triangle M$ is an M-alternating 2-factor. By Lemma 3.4 and Theorem 3.5 we know that there exists an augmentor A with $\phi_M(A) \geq \frac{\phi_M(M^* \triangle M)}{n}$, where n is the number of nodes in G. By Lemma 2.3 we can efficiently find an M-alternating 2-factor S such that $\phi_M(S) \geq \phi_M(A)$. Since $\phi_M(M^* \triangle M) = cost(M) - OPT$, then every time we augment in our algorithm, we decrease the cost by at least $(cost(M) - OPT)/n$.

Therefore, in the above algorithm, if we start with a matching that is D more expensive than OPT, then we will decrease D by at least a factor of $\frac{n-1}{n}$ at every step. So at the kth step we have a solution of cost at most

$$D \cdot \left(\frac{n-1}{n}\right)^k + OPT.$$

Therefore, it will take

$$\frac{\log D}{\log(n/n-1)}$$

steps until we find a perfect matching with $cost(M) - OPT < 1$. Since we have integer weights, this matching must be optimal:

$$\log \frac{n}{n-1} = \log n - \log(n-1) > 1/n,$$

so we require at most $n \log D$ steps. In general, D can be as large as $n \cdot c_{\max}$, where $c_{\max}$ is the maximum cost of any edge (or the ratio between the maximum and minimum costs), so the total necessary number of invocations of Lemma 2.3 is $O(n \log n + n \log c_{\max})$. For most of our applications, however, we can find an initial matching which is a close approximation to OPT, and so $D < OPT$.

Done in a naive manner, each invocation of Lemma 2.3 consists of running a min-cost weighted matching algorithm for every pair of 3D edges. This could take as long as $O(n^3m^2)$, where m is the number of 3D edges. However, there are some simple ways to make this step run faster. We could take advantage of the fact that the min-cost matchings that we are calculating are closely related. If we use an "augmenting path" algorithm for calculating min-cost matchings [16], then each calculation takes only $O(n^2)$ time, giving us a running time of $O(n^3 + n^2m^2)$ for each invocation of Lemma 2.3. We can reduce this running time further in the geometric setting. Finally, notice that not all pairs of 3D edges need to be considered. A lot of the pairs can be eliminated in advance, significantly reducing the running time. This is especially true when applying this algorithm to the Terminal Backup problem, or to any problem involving covering instead of exact matching. Even in general settings, we do not need to consider pairs of 3D edges (e_1, e_2) if $e_1 \notin M$ is adjacent to a 3D edge of M that is not e_2. For more details on optimizing the running time, see Xu, Ansheleuch, and Chiang [41].

Recall that in the first step of the algorithm we find an unweighted perfect matching, which can be done in time $O(m^2)$ using [38]. In the worst case, we can assume that m is $\Omega(n)$. Therefore, the total bound on the running time of our algorithm (without any optimizations) is $O(n^3m^2 \log n + n^3m^2 \log c_{\max})$. Since the running time depends on $\log c_{\max}$, it is polynomial-time, but not strongly polynomial-time. $\square$

In the case where the edge costs are not integer, the running time will depend on how these costs are represented. By running the augmentation algorithm until the improvement is at most ε, we can obtain an algorithm that finds a solution that costs at most $OPT + \varepsilon$ in time polynomial in $\log D$, $\log(1/\varepsilon)$, and n. At this time, it is still an open question whether a strongly polynomial-time algorithm exists to find a min-cost Simplex Matching.

6. Conclusion: Solving problems using Simplex Matching.

In this section we show how several different problems can be solved using our algorithm for Simplex Matching, which we presented in section 2.

Project Assignment. Consider the Project Assignment problem described in the introduction. It easily reduces to Simplex Matching as follows. First notice that if the optimal solution forms groups larger than 3, then there is an equivalent solution with groups of only 2 and 3 students. This is because we can always break up a group of size more than 3 into groups of size 2 and 3 that are working on the same project. Define the cost of assigning a student s to a project p to be $L - u(s,p)$ for some large constant L that is greater than all utilities $u(s,p)$. Now, form a complete graph H with the students as nodes of H, 2D edges (s_1, s_2) representing the smallest cost of assigning students s_1 and s_2 to a project together, and 3D edges representing the same cost for triples of students. In other words, $c(s_1, s_2) = \min_p\{L - u(s_1, p) + L - u(s_2, p)\}$, and similarly, $c(s_1, s_2, s_3) = \min_p\{3L - u(s_1, p) - u(s_2, p) - u(s_3, p)\}$. The Simplex Condition will hold in any such H created from an instance of Project Assignment. All possible 2D edges (s_1, s_2) exist and if an edge (s_1, s_2, s_3) corresponds to a project p, then $c(s_i, s_j) \leq 2L - u(s_i, p) - u(s_j, p)$ for any pair of students (s_i, s_j) (Figure 1.1(b)), by definition of $c(s_i, s_j)$. Therefore, $c(s_1, s_2) + c(s_2, s_3) + c(s_1, s_3) \leq 6L - 2u(s_1, p) + 2u(s_2, p) + 2u(s_3, p)$. This gives us the inequality portion of the Simplex Condition.

By finding the min-cost perfect matching in H, we also find the best solution to the Project Assignment problem. This is simply because a perfect matching in H will have a cost of Ln minus the cost of a project assignment, and for every project assignment with groups of at most 3 people there must exist a corresponding perfect matching in H.

Terminal Backup. The reduction from Terminal Backup to Simplex Matching is more complicated (see [41] for details). First, we can assume that all connected components of an optimal solution to Terminal Backup are either a path between two terminals or a star with three terminals as the leaves and a Steiner node at the center. (Technically, a connected component can be a star with more than 3 leaves, but then we can think of it as several edge-disjoint stars with at most 3 leaves.) With this assumption we can form a complete graph H with terminals as the nodes. We set the cost $c(u, v)$ of a 2D edge in H to be the cost of the cheapest path connecting u and v, and the cost $c(u, v, w)$ of a 3D edge to be the cost of the cheapest star connecting all three terminals u, v, and w. As with Project Assignment, any such graph H will satisfy the Simplex Condition. Consider an instance of Terminal Backup shown in Figure 1(c). There are three terminals u_1, u_2, and u_3 and a single Steiner node p. Let $c(u_i, u_j)$ be the cost of the cheapest path connecting terminal u_i to terminal u_j and $c(u_1, u_2, u_3)$ be the cost to connect all three terminals through p. Notice that if we wanted only to connect u_i to u_j, we could connect them through p, so we obtain that $c(u_1, u_2) + c(u_1, u_3) + c(u_2, u_3) \leq 2\,c(u_1, u_2, u_3)$.

The min-cost Simplex Matching in H will give us the connected components for the optimum solution of the Terminal Backup instance. To see this, consider the optimum solution OPT of the Terminal Backup instance. By the argument in [41], it consists of a union of edge-disjoint cheapest paths between pairs of terminals and edge-disjoint cheapest stars joining triples of terminals. This cost is exactly the cost of some matching in H. Conversely, every perfect matching M in H with cost α can be used to form a solution to Terminal Backup with cost at most α, just by taking the stars and paths corresponding to the edges of M, and taking their union.

While the main result of this paper gives a polynomial-time algorithm for finding the minimum-weight Simplex Matching, we are also concerned with approximation algorithms. Specifically, there is a simple 4/3-approximation algorithm for Terminal Backup which is more efficient than the exact algorithm. This approximation algorithm could be useful when we do not care about finding the best solution but instead are more concerned with the running time. In this algorithm, we form a graph H', which is the same as the graph H above but contains only 2D edges. We then proceed to find the min-cost *edge cover* C of H'. We claim that C has cost at most 4/3 times the cost of M^* (the cheapest perfect matching of H), and so the solution consisting of paths corresponding to edges of C is a 4/3 approximation to Terminal Backup.

To prove the above claim, consider an edge cover C^* of H' formed by taking all the edges of M^* and replacing every 3D edge (u, v, w) of M^* with the cheapest pair of corresponding 2D edges, i.e., with $\{(u, v), (v, w)\}$, $\{(u, v), (u, w)\}$, or $\{(u, w), (v, w)\}$. This is an edge cover of H' since M is a perfect matching of H, and so $cost(C) \leq cost(C^*)$. Now consider how much the cost can increase by our replacing a 3D edge (u, v, w) with two 2D edges as described above. Assume without loss of generality that these two edges are (u, v) and (v, w). By our choice of the two edges, we know that $c(u, v) + c(v, w) \leq c(u, v) + c(u, w)$ and $c(u, v) + c(v, w) \leq c(u, w) + c(v, w)$. Therefore, $3(c(u, v) + c(v, w)) \leq 2c(u, v) + 2c(v, w) + 2(u, w) \leq 4\, c(u, v, w)$, with the last inequality being true because of the Simplex Condition. As desired, this tells us that $cost(C) \leq cost(C^*) \leq \frac{4}{3}cost(M^*)$.

This approximation algorithm works since the Terminal Backup problem required only that every terminal be connected to *at least* one other, and so is essentially a covering problem, instead of a matching problem. The same algorithm will work for other applications where we are concerned with only the edge-cover version of the Simplex Matching problem.

Problems obeying the Simplex Condition. Simplex Matching seems well suited to solving problems where a set of elements needs to be partitioned into groups of size 2 or 3, and the problem can be made to satisfy the Simplex Condition. Terminal Backup and Project Assignment are two seemingly different problems of this kind, since in both the costs satisfy a version of the triangle inequality.

Recently Blocki, Blum, and Williams [9] used the weighted Simplex Matching algorithm to show that the 2-anonymity optimization problem from data privacy is efficiently solvable (for a definition of k-anonymity, see [39]; for a discussion of its complexity, see [31]; and for approximation algorithms, see [3]).

Kidney Exchange (see, e.g., [1, 36]) is another important problem that is relevant to Simplex Matching. It involves pairing potential donors with potential kidney recipients. Every node is a pair of people, one of whom is willing to donate a kidney, and the other of whom needs a kidney. For example, such a pair might consist of a husband and wife, with the husband needing a kidney and the wife willing to donate a kidney to him, but with the kidneys being incompatible. Therefore, every such pair (node) needs another pair with appropriate compatibilities, so they could trade kidneys with the other pair [1, 36]. This is more general than a 2D matching problem because cycles of size 3 are also acceptable (where kidneys are traded along the cycle), and this is a weighted problem because there are different levels of compatibility between donors and recipients. Unfortunately, this is not directly reducible to Simplex Matching, since the nature of compatibilities makes the underlying graph directed (i.e., if pair (u_1, u_2) can give pair (v_1, v_2) a kidney, this does not mean that (v_1, v_2) can give (u_1, u_2) a kidney), and so the Simplex Condition does not hold. Extending the ideas from Simplex Matching to design better approximation algorithms for Kidney Exchange remains an interesting research direction.

Appendix. Detailed case analysis of graph decompositions in Lemmas 4.5 and 4.7.

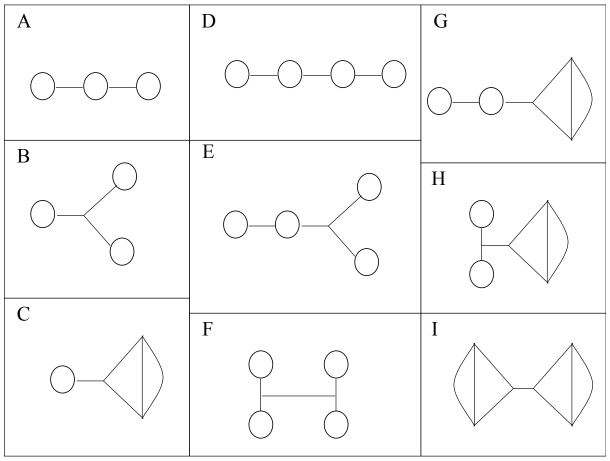

FIG. A.1. *All possible structures $C(A_1, A_2)$ that can be created by patching together disjoint dual augmentors A_1 and A_2 as described in the proof of Lemma* 4.5. *See Table* A.1 *for their descriptions. Note that we are not including the structures formed from dual augmentors of Type* 1a *or* 1b *(only* 1c*), but the arguments and descriptions for $C(A_1, A_2)$ involving those structures are completely analogous: just replace one or two loops with a node in M.*

TABLE A.1
A full list of structures $C(A_1, A_2)$ (see Lemmas 4.5 *and* 4.6*) and how to decompose them using one of the three rules given in the proof of Lemma* 4.6. *The first column gives one of the structures in Figure* A.1*, the second column says which types of dual augmentors could create such a structure by being patched together, and the third and fourth columns give a rule that can be used to decompose this structure.*

Structure type	Created from patching	Decomposed into	Using rule
A	Type 0 and Type 1c	Dual augmentors	Rule 3
B	Type 0 and Type 1c	Dual augmentors	Rule 1 or 2
C	Type 0 and Type 2	Dual augmentors	Figure 4.5
D	Type 1c and Type 1c	Dual augmentors and Structure A	Rule 3
E	Type 1c and Type 1c	Dual augmentors and Structure B	Rule 3
F	Type 1c and Type 1c	Dual augmentors and Structure B	Rule 1 or 2
G	Type 1c and Type 2	Dual augmentors and Structure C	Rule 3
H	Type 1c and Type 2	Dual augmentors and Structure C	Rule 1 or 2
I	Type 2 and Type 2	Dual augmentors and Structure C	Figure 4.5

Table A.2

A full list of structures formed by combining two node-disjoint dual augmentors A_1 and A_2 into a multigraph C using the patching process described in the proof of Lemma 4.7. *Removing an edge from a dual augmentor results in one of the objects in Figure* 4.7. *Number those objects from left to right (Types* 1*–*7*). C is a result of joining any two of these objects along the two connector nodes. The first column lists all possible pairs that could be joined together to form C. The second column describes what the result looks like. "Equivalent to" means that some of the loops in Figure* A.1 *are replaced with nodes in M, which does not change any of the arguments about those sets having valid augmentor sums.*

Types of objects we join together	Structure of resulting C
Type 1 + Type 1	Two dual augmentors of Type 1a
Type 1 + Type 2	Dual augmentor of Type 1a
Type 1 + Type 3	Equivalent to Structure B in Figure A.1
Type 1 + Type 4	Two dual augmentors of Type 1a and 1b
Type 1 + Type 5	Equivalent to Structure B in Figure A.1
Type 1 + Type 6	Two dual augmentors of Type 1b
Type 1 + Type 7	Equivalent to Structure A in Figure A.1
Type 2 + Type 2	Dual augmentor of Type 0
Type 2 + Type 3	Dual augmentor of Type 1b
Type 2 + Type 4	Dual augmentor of Type 1b
Type 2 + Type 5	Dual augmentor of Type 1c
Type 2 + Type 6	Dual augmentor of Type 1c
Type 2 + Type 7	Dual augmentor of Type 2
Type 3 + Type 3	Equivalent to Structure A in Figure A.1
Type 3 + Type 4	Equivalent to Structure B in Figure A.1
Type 3 + Type 5	Equivalent to Structure A in Figure A.1
Type 3 + Type 6	Equivalent to Structure B in Figure A.1
Type 3 + Type 7	Equivalent to Structure C in Figure A.1
Type 4 + Type 4	Two dual augmentors of Type 1a and 1c (or 1b and 1b)
Type 4 + Type 5	Equivalent to Structure B in Figure A.1
Type 4 + Type 6	Two dual augmentors of Type 1b and 1c
Type 4 + Type 7	Equivalent to Structure A in Figure A.1
Type 5 + Type 5	Equivalent to Structure A in Figure A.1
Type 5 + Type 6	Equivalent to Structure B in Figure A.1
Type 5 + Type 7	Equivalent to Structure C in Figure A.1
Type 6 + Type 6	Two dual augmentors of Type 1c
Type 6 + Type 7	Equivalent to Structure A in Figure A.1
Type 7 + Type 7	See Figure 4.8

Acknowledgments. The authors would like to thank Moses Charikar, Sanjeev Khanna, and Avi Wigderson for interesting and productive discussions. We especially thank Paul Seymour for his great contribution to this paper, and Dahai Xu and Mung Chiang for formulating the Terminal Backup problem.

REFERENCES

[1] D. Abraham, A. Blum, and T. Sandholm, *Clearing algorithms for barter exchange markets: Enabling nationwide kidney exchanges*, in Proceedings of the 8th ACM Conference on Electronic Commerce, ACM, New York, 2007, pp. 295–304.

[2] Z. Abrams, A. Meyerson, K. Munagala, and S. Plotkin, *On the Integrality Gap of Capacitated Facility Location*, Technical report CMU-CS-02-199, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, 2002.

[3] G. Aggarwal, T. Feder, K. Kenthapadi, R. Motwani, R. Panigrahy, D. Thomas, and A. Zhu, *Approximation algorithms for k-Anonymity*, J. Privacy Technology, 2005 (2005), 20051120001.

[4] E. Anshelevich and B. Caskurlu, *Exact and approximate equilibria for optimal group network formation*, in Proceedings of the 17th Annual European Symposium on Algorithms, Copenhagen, Denmark, 2009.

[5] K. Appel and W. Haken, *Every Planar Map Is Four-Colorable*, Contemp. Math. 98, AMS, Providence, RI, 1989.

[6] E. Arkin, R. Hassin, S. Rubinstein, and M. Sviridenko, *Approximations for maximum transportation problem with permutable supply vector and others capacitated star packing problems*, Algorithmica, 39 (2004), pp. 175–187.

[7] M. Bläser and B. Manthey, *Two approximation algorithms for 3-cycle covers*, in Proceedings of the 5th International Workshop on Approximation Algorithms for Combinatorial Optimization, Springer, Berlin, 2002, pp. 40–50.

[8] M. Bläser, L. Ram, and M. Sviridenko, *Improved approximation algorithms for metric maximum ATSP and maximum 3-cycle cover problems*, in Algorithms and Data Structures, Springer, Berlin, 2005, pp. 350–359.

[9] J. Blocki and R. Williams, *Resolving the complexity of some data privacy problems*, in Proceedings of the 37th International Colloquium on Automata, Languages and Programming, Springer, Berlin, 2010, pp. 393–404.

[10] G. Calinescu and A. Zelikovsky, *The polymatroid Steiner problems*, J. Combin. Optim., 9 (2005), pp. 281–294.

[11] J. Chen, S. Lu, S. Sze, and F. Zhang, *Improved algorithms for path, matching, and packing problems*, in Proceedings of the 18th ACM–SIAM Symposium on Discrete Algorithms, ACM, New York, SIAM, Philadelphia, 2007, pp. 298–307.

[12] G. Cornuéjols, *Combinatorial Optimization: Packing and Covering*, CBMS-NSF Regional Conf. Ser. in Appl. Math. 74, SIAM, Philadelphia, 2001.

[13] G. Cornuéjols and D. Hartvigsen, *An extension of matching theory*, J. Combin. Theory Ser. B, 40 (1986), pp. 285–296.

[14] G. Cornuéjols, D. Hartvigsen, and W. Pulleyblank, *Packing subgraphs in a graph*, OR Letters, 1 (1982), pp. 139–143.

[15] W. Cunningham, *Matching, matroids, and extensions*, Math. Program. Ser. B, 91 (2002), pp. 515–542.

[16] U. Derigs, *A shortest augmenting path method for solving minimal perfect matching problems*, Networks, 11 (1981), pp. 379–390.

[17] G. Even, G. Kortsarz, and W. Slany, *On network design problems: Fixed cost flows and the covering Steiner problem*, ACM Trans. Algorithms, 1 (2005), pp. 74–101.

[18] M. X. Goemans and D. P. Williamson, *A general approximation technique for constrained forest problems*, SIAM J. Comput., 24 (1995), pp. 296–317.

[19] S. Guha, A. Meyerson, and K. Munagala, *Hierarchical placement and network design problems*, in Proceedings of FOCS, IEEE, Washington, DC, 2000, pp. 603–612.

[20] A. Gupta and A. Srinivasan, *On the covering Steiner problem*, Theory Comput., 2 (2006), pp. 53–64.

[21] D. Hartvigsen, P. Hell, and J. Szabó, *The k-piece packing problem*, J. Graph Theory, 52 (2006), pp. 267–293.

[22] P. Hell, *Graph packings*, in 6th International Conference on Graph Theory, Electron. Notes Discrete Math. 5, Elsevier, Amsterdam, 2000.

[23] P. Hell and D. Kirkpatrick, *Packings by cliques and by finite families of graphs*, Discrete Math., 49 (1984), pp. 45–49.

[24] U. Jamshy and M. Tarsi, *Short cycle covers and the cycle double cover conjecture*, J. Combin. Theory Ser. B, 56 (1992), pp. 197–204.

[25] H. Kaplan, M. Lewenstein, N. Shafrir, and M. Sviridenko, *Approximation algorithms for asymmetric TSP by decomposing directed regular multigraphs*, J. ACM, 52 (2005), pp. 602–626.

[26] A. Karagiozova, *Aspects of Network Design*, Ph.D. thesis, Princeton University, Princeton, NJ, 2007.

[27] D. Karger and M. Minkoff, *Building Steiner trees with incomplete global knowledge*, in Proceedings of FOCS, IEEE, Washington, DC, 2000, pp. 613–623.

[28] A. Kelmans, *Optimal packing of induced stars in a graph*, Discrete Math., 173 (1997), pp. 97–127.

[29] M. Loebl and S. Poljak, *Efficient subgraph packing*, J. Combin. Theory Ser. B, 59 (1993), pp. 106–121.

[30] L. Lovasz and M. Plummer, *Matching Theory*, Elsevier Science, Amsterdam, 1986.

[31] A. Meyerson and R. Williams, *General k-Anonymization Is Hard*, CMU Technical report CMU-CS-03-113, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, 2003.

[32] G. Pap, *Hypo-matchings in directed graphs*, in Graph Theory 2004, Birkhäuser Verlag, Basel, 2006, pp. 325–335.

[33] G. Pap, *A TDI description of restricted* 2*-matching polytopes*, in IPCO 2004, New York, NY, pp. 139–151.
[34] N. Robertson and P. D. Seymour, *Graph minors.* IX. *Disjoint crossed paths*, J. Combin. Theory Ser. B, 49 (1990), pp. 40–77.
[35] N. Robertson, D. P. Sanders, P. D. Seymour, and R. Thomas, *The four-colour theorem*, J. Combin. Theory Ser. B, 70 (1997), pp. 2–44.
[36] S. Saidman, A. Roth, T. Sönmez, M. U. Ünver, and F. Delmonico, *Increasing the opportunity of live kidney donation by matching for two and three way exchanges*, Transplantation, 81 (2006), pp. 773–782.
[37] P. D. Seymour, *Sums of circuits*, in Graph Theory and Related Topics, J. A. Bondy and U. R. S. Murty, eds., Academic Press, New York, 1979, pp. 341–355.
[38] A. Shalita and U. Zwick, *Efficient algorithms for the* 2*-gathering problem*, in Proceedings of the 19th Annual ACM-SIAM Symposium on Discrete Algorithms, ACM, New York, SIAM, Philadelphia, 2009, pp. 96–105.
[39] L. Sweeney, *k-anonymity: A model for protecting privacy*, Internat. J. Uncertain. Fuzziness Knowledge-Based Systems, 10 (2002), pp. 557–570.
[40] P. G. Tait, *Note on a theorem in geometry of position*, Trans. Roy. Soc. Edinburgh, 29 (1880), pp. 657–660.
[41] D. Xu, E. Anshelevich, and M. Chiang, *On survivable access network design: Complexity and algorithms*, in Proceedings of INFOCOM, IEEE, Washington, DC, 2008.
[42] C. Q. Zhang, *Integer Flows and Cycle Covers of Graphs*, Marcel Dekker, New York, 1997.