FRACTIONAL PACKING OF T-JOINS

FRANCISCO BARAHONA

Abstract. Given a graph with nonnegative capacities on its edges, it is well known that the capacity of a minimum T-cut is equal to the value of a maximum fractional packing of T-joins. Padberg-Rao's algorithm finds a minimum capacity T-cut but it does not produce a T-join packing, we present a polynomial combinatorial algorithm for finding an optimal T-join packing.

1. Introduction

We present a polynomial combinatorial algorithm for packing T-joins in a capacitated graph. Given a graph $G = (V, E)$ and $S \subseteq V$, the set of all edges with exactly one endnode in S is called a *cut* and is denoted by $\delta_G(S)$. We say that S *defines* the cut $\delta_G(S)$. If the graph G is clear from the context we use $\delta(S)$. Given a set $T \subseteq V$ of even cardinality, we say that a cut $\delta(S)$ is a T-cut if $|S \cap T|$ is odd. A set of edges J is called a T*-join* if in the subgraph $G' = (V, J)$ the nodes in T have odd degree and the nodes in $V \setminus T$ have even degree. T-joins appear in the solution of the Chinese postman problem by Edmonds & Johnson [5]. Here the nodes in T are the nodes of odd degree and a T-join is a set of edges that have to be duplicated to obtain an Eulerian graph.

Edmonds & Johnson [5] proved that if A is a matrix whose rows are the incidence vectors of all T-cuts, then for any nonnegative objective function w the linear program below has an optimal integer solution that is the incidence vector of a T-join.

$$\min wx \tag{1}$$
$$Ax \geq 1 \tag{2}$$
$$x \geq 0. \tag{3}$$

Edmonds & Johnson gave a combinatorial polynomial algorithm to solve the linear program above and its dual

$$\max y1 \tag{4}$$
$$yA \leq w \tag{5}$$
$$y \geq 0. \tag{6}$$

This gives a packing of T-cuts. Seymour [15] proved that if the coefficients of w are integer, and their sum over every cycle is an even number, then (4)-(6) has an optimal integer solution. The algorithm of Edmonds & Johnson can be modified to produce this integer dual optimal solution, see [2].

Date: November 13, 2003.

It follows from the theory of Blocking Polyhedra [6] that if B is a matrix whose rows are all incidence vectors of T-joins then for any nonnegative objective function c the linear program below has also an optimal integer solution that is the incidence vector of a T-cut.

$$\min cx \tag{7}$$
$$Bx \geq 1 \tag{8}$$
$$x \geq 0. \tag{9}$$

The dual problem is

$$\max y1 \tag{10}$$
$$yB \leq c \tag{11}$$
$$y \geq 0. \tag{12}$$

A solution of (10)-(12) is a maximum packing of T-joins. So from linear programming duality we have that the value of a maximum packing of T-joins is equal to the value of a minimum T-cut. Padberg & Rao [13] gave a polynomial combinatorial algorithm that finds a minimum T-cut. However this algorithm does not give a maximum packing of T-joins, and this has remained unsolved. Due to the equivalence between separation and optimization, one could solve this in polynomial time with the ellipsoid method, see [10]. The purpose of this paper is to give a polynomial combinatorial algorithm for finding a maximum (fractional) packing. To the best of our knowledge the only case that is well solved is when $|T| = 2$, this is the well known maximum flow problem. Our algorithm has many similarities with an algorithm for packing arborescences given by Gabow and Manu [8].

There are several conjectures and questions related to the case when the linear program (10)-(12) has an integer solution. We discuss them below.

A graph is called *r-regular* if all its vertices have degree r. A graph is called an *r-graph* if it is r-regular and every V-cut has cardinality greater than or equal to r. A *perfect matching* is a set of nonadjacent edges that covers every vertex of the graph. Fulkerson made the following conjecture.

Conjecture 1. *Every 3-graph has six perfect matchings that include each edge at most twice.*

Notice that for a 3-graph, when $T = V$ every vertex defines a minimum T-cut. Also every T-join with positive weight in a maximum packing should intersect a minimum T-cut in exactly one edge, so the T-join should be a perfect matching. Thus in our terminology the conjecture above is equivalent to say that for a 3-graph when $T = V$ and c is a vector of all twos, then (10)-(12) has an optimal solution that is integer.

Seymour [14] generalized Fulkerson's conjecture as below.

Conjecture 2. *Every r-graph has $2r$ perfect matchings that include each edge at most twice.*

Seymour [14] also made the following two conjectures and proved that they are implied by Conjecture 2. A family of T-joins is called *k-disjoint* if every edge is included in at most k of them.

Conjecture 3. *If every vertex has an even degree then the size of a maximum 2-disjoint family of T-joins equals the double of the size of a minimum T-cut.*

Conjecture 4. *The size of a 4-disjoint family of T-joins equals four times the size of a minimum T-cut.*

Cohen & Lucchesi [3] made the conjecture below and proved that it is equivalent to Conjecture 2.

Conjecture 5. *If all T-cuts have the same parity then the size of a maximum 2-disjoint family of T-joins equals the double of the size of a minimum T-cut.*

They also proved the following.

Theorem 6. *If $|T| \leq 8$ and every T-cut has the same parity then the size of a maximum disjoint family of T-joins equals the size of a minimum T-cut.*

Conforti & Johnson [4] made the following conjecture. They proved their conjecture for graphs without a 4-wheel minor.

Conjecture 7. *If T is the set of nodes of odd degree, and the graph is not contractible to the Petersen graph, then the size of a maximum disjoint family of T-joins equals the size of a minimum T-cut.*

Holyer [11] proved that deciding whether a 3-regular simple graph has 3 disjoint perfect matchings is NP-complete. So finding an optimal integer solution of (10)-(12) is NP-hard. Tait [16] proved that the Four Color Theorem is equivalent to the statement that every 2-connected planar 3-regular graph has 3 disjoint perfect matchings. This is equivalent to say that for every 2-connected planar 3-regular graph, when $T = V$ and c is the vector of all ones, the linear program (10)-(12) has an optimal solution that is integer.

Now we give some extra notation and definitions. Let $n = |V|$ and $m = |E|$. We assume that every edge e has a nonnegative *capacity* $c(e)$. If $c(e)$ is zero then the edge e is removed from the graph. For $S \subseteq V$ we use $\theta(S)$ to denote

$$\theta(S) = \sum\{c(e) \,:\, e \in \delta(S)\},$$

this the *capacity* of the cut $\delta(S)$. Given $A, B \subseteq V$, we say that they *cross* if the sets $A \setminus B$, $B \setminus A$, and $A \cap B$ are nonempty. A family of sets such that no two of them cross is called *laminar*. A laminar family of subsets of V can have at most $2n - 1$ nonempty sets. It is well known that θ is a *submodular* function, i. e., for any two sets $A, B \subseteq V$

$$\theta(A \cup B) + \theta(A \cap B) + 2\beta(A, B) = \theta(A) + \theta(B),$$

where $\beta(A, B)$ is the sum of the capacities of the edges with one endnode in $A \setminus B$ and the other in $B \setminus A$. We use $\lambda(G)$ to denote the capacity of a minimum T-cut in G, i. e.,

$$\lambda(G) = \min\{\theta(S) \,:\, S \subset V,\ |S \cap T| \text{ is odd}\}.$$

For $U \subseteq E$ we use $\mu(U)$ to denote

$$\mu(U) = \min\{c(e) \,:\, e \in U\},$$

this is called the *bottleneck capacity* of U. If J is a T-join and $\delta(S)$ is a cut, then $|J \cap \delta(S)|$ is odd if and only if $\delta(S)$ is a T-cut. If $U \subseteq E$ and $0 \leq \alpha \leq \mu(U)$, we denote by $G - \alpha U$ the graph obtained by replacing the capacity $c(e)$ of every edge $e \in U$, by $c(e) - \alpha$. If $U \subseteq E$ the *incidence vector* of U, denoted by x^U, is defined as $x^U(e) = 1$ if $e \in U$,

$x^U(e) = 0$ otherwise. A minimum cut separating nodes s and t is called a *minimum st-cut*. The nodes in the set T are called *T-nodes*.

This paper is organized as follows. In Section 2 we give a short description of Padberg-Rao's algorithm for finding a minimum T-cut. In Section 3 we present an initial description of the algorithm for packing T-joins. Sections 4 and 5 are devoted to more technical aspects required to complete the description of our algorithm. Section 6 contains a final analysis of our algorithm.

2. Padberg-Rao's algorithm

For the sake of completeness we give a short description of Padberg-Rao's algorithm for finding a minimum T-cut. It is based on the following lemma.

Lemma 1. *Let S define a minimum cut separating at least two nodes in T. If $|S \cap T|$ is odd then S defines a minimum T-cut. Otherwise, there is a set $S' \subseteq S$ or $S' \subseteq V \setminus S$ that defines a minimum T-cut.*

Proof. Assume that $|S \cap T|$ is even and consider a set A that defines a minimum T-cut. Suppose that A and S cross.

Case 1: $|A \cap S \cap T|$ is odd. If $A \cup S$ separates at least two nodes in T, we have

$$\theta(A \cap S) + \theta(A \cup S) \leq \theta(A) + \theta(S).$$

Therefore $\theta(A \cap S) = \theta(A)$ and $\theta(A \cup S) = \theta(S)$. Thus $A \cap S$ defines a minimum T-cut.

If $T \subseteq A \cup S$, let $\bar{A} = V \setminus A$, then

$$\theta(\bar{A} \cap S) + \theta(\bar{A} \cup S) \leq \theta(\bar{A}) + \theta(S).$$

Thus $\theta(\bar{A} \cap S) = \theta(\bar{A})$, $\theta(\bar{A} \cup S) = \theta(S)$, and $\bar{A} \cap S$ defines a minimum T-cut.

Case 2: $|A \cap S \cap T|$ is even. Let $\bar{S} = V \setminus S$. Then $|A \cap \bar{S} \cap T|$ is odd and this reduces to Case 1. □

This lemma suggests a very simple algorithm, namely if S defines a minimum cut separating at least two nodes in T, then either S defines a minimum T-cut or one should continue working recursively with the graph G_1 obtained by contracting S and with the graph G_2 obtained by contracting $V \setminus S$.

Padberg & Rao also pointed out that one should first compute a Gomory-Hu (GH) tree [9], and then carry out the algorithm above on the GH-tree. This is because any minimum *st*-cut in the graph is given by a minimum *st*-cut in the GH-tree. Because of the tree structure, the algorithm becomes extremely simple: among all edges in the tree that are a T-cut, we should pick one of minimum capacity.

Thus the complexity of this procedure is the complexity of computing a GH-tree, i. e., computing $(n-1)$ minimum *st*-cuts.

3. The Algorithm

We start this section with an initial description of the algorithm. Clearly the capacity of any T-cut is an upper bound for the value of a T-join packing. As mentioned in the Introduction there is a fractional packing of T-joins whose value is equal to the capacity of a minimum T-cut. For this bound to be tight, any T-join with a positive weight in an optimal packing must intersect any minimum T-cut in exactly one edge. Also, given an optimal packing, every edge e in a minimum T-cut, with $c(e) > 0$, must appear in a T-join with positive weight. The algorithm works based on this.

Using $\lambda(G)$ as the target value, the problem is solved recursively in a *greedy* way as follows. For a T-join U, let α_U be the largest value of α such that $\lambda(G-\alpha U) = \lambda(G) - \alpha$ and $0 \leq \alpha \leq \mu(U)$. Then the weight α_U is assigned to U. If $\lambda(G - \alpha_U U) > 0$ one should continue working recursively with $G - \alpha_U U$. In the remainder of this paper we show that a refinement of this algorithm runs in polynomial time. We need first a simple lemma.

Lemma 1. *If U is a T-join and $\alpha_U = 0$ then there is a minimum T-cut $\delta(S)$ such that $|\delta(S) \cap U| > 1$.*

Proof. First notice that $\lambda(G - \alpha U) \leq \lambda(G) - \alpha$, for $0 \leq \alpha \leq \mu(U)$. This is because in $G - \alpha U$ the capacity of a T-cut $\delta(S)$ is $\theta(S) - k\alpha$, where $k = |\delta_G(S) \cap U|$.

So if $|\delta(S) \cap U| = 1$ for every minimum T-cut $\delta(S)$ then there is a small value of $\alpha > 0$, such that $\lambda(G - \alpha U) = \lambda(G) - \alpha$ and $\alpha \leq \mu(U)$. □

From the lemma above we can see that one should concentrate on T-joins that intersect *every* minimum T-cut in exactly *one* edge. When we impose this condition for a minimum T-cut $\delta(S)$, we say that it is *tight*, we also say that S is a *tight set*. The two lemmas below show that we only need to impose this for a laminar family of tight sets.

Lemma 2. *Assume that A and B define minimum T-cuts, they cross, and $|A \cap B \cap T|$ is odd. Then the tightness of $A \cap B$ and $A \cup B$ imply the tightness of A and B.*

Proof. We have that

$$\theta(A \cap B) + \theta(A \cup B) \leq \theta(A) + \theta(B).$$

Since A and B define minimum T-cuts, then $A \cap B$ and $A \cup B$ also define minimum T-cuts. Therefore this inequality must hold as equation. This implies that there is no edge between $A \setminus B$ and $B \setminus A$. Moreover for a T-join U and any cut $\delta(S)$ the cardinality of $\delta(S) \cap U$ is odd if S defines a T-cut and even otherwise. Then by a counting argument it is easy to see that any T-join that has exactly one edge entering $A \cap B$ and exactly one edge entering $A \cup B$ must have exactly one edge entering A and exactly one edge entering B. Figure 1 displays all possible configurations. □

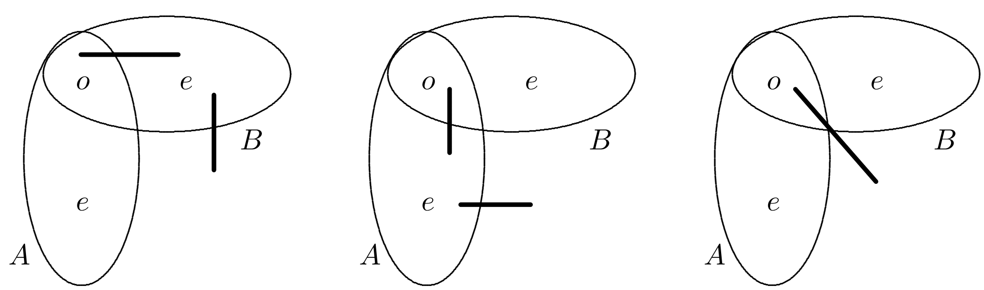

FIGURE 1. The labels e (even) and o (odd) refer to the parity of $|(A \setminus B) \cap T|$, $|A \cap B \cap T|$ and $|(B \setminus A) \cap T|$.

Lemma 3. *Assume that A and B define minimum T-cuts, they cross, and $|A \cap B \cap T|$ is even. Then the tightness of $A \setminus B$ and $B \setminus A$ imply the tightness of A and B.*

Proof. Apply Lemma 2 to A and $\bar{B} = V \setminus B$. □

So when we keep a family of tight sets, we can apply the last two lemmas to convert it into a laminar family. Denote by Φ this family, it can contain at most $2n-1$ tight sets. We are going to find a T-join that intersects every T-cut given by Φ in exactly one edge. Let U be this T-join. There are two possible cases:

1. If $\alpha_U = \mu(U)$ then the number of edges in $G - \alpha_U U$ is at least one less than the number of edges in G.

2. If $\alpha_U < \mu(U)$ then in $G - \alpha_U U$ there is a minimum T-cut $\delta(S)$, $S \notin \Phi$, such that $|U \cap \delta(S)| > 1$. In this case we should add S to Φ and uncross it using Lemmas 2 and 3 as in the procedure below.

Uncross (Φ, S, U)

Input: The family Φ, a set $S \notin \Phi$, a T-join U
While there is a set $A \in \Phi$ such that A and S cross
do
  if $|A \cap S \cap T|$ is odd
    if $|\delta(A \cup S) \cap U| > 1$ set $S \leftarrow A \cup S$
    if $|\delta(A \cup S) \cap U| = 1$ set $S \leftarrow A \cap S$
  if $|A \cap S \cap T|$ is even
    if $|\delta(A \setminus S) \cap U| > 1$ set $S \leftarrow A \setminus S$
    if $|\delta(A \setminus S) \cap U| = 1$ set $S \leftarrow S \setminus A$
end
Add S to Φ

It is easy to see that at each uncrossing step the number of crossing pairs decreases by at least one. Also at the end of this procedure the cardinality of Φ increases by one.

Now we can give a formal description of the algorithm.

Pack T-joins

Step 0. Set $\Phi \leftarrow \emptyset$.
Step 1. Find a T-join U such that $|U \cap \delta(S)| = 1$, for all $S \in \Phi$.
Step 2. Compute α_U as the maximum of α such that
$\lambda(G - \alpha U) = \lambda(G) - \alpha$, and $0 \le \alpha \le \mu(U)$.
Step 3. If $\alpha_U < \mu(U)$, a new tight T-cut $\delta(S)$ has been found.
Apply Uncross(Φ, S, U).
Step 4. Set $G \leftarrow G - \alpha_U U$. If $\lambda(G) = 0$ stop, otherwise go to Step 1.

Since at each iteration either the cardinality of Φ increases or one edge is deleted, the total number of iterations is at most $2n - 1 + m$. It remains to describe how to perform Steps 1 and 2. This is the subject of the next two sections.

4. Finding a T-join in Step 1

As it was said in the Introduction, it follows from linear programming duality that there is a fractional packing of T-joins whose value is $\lambda(G)$. The purpose of this section is to find one T-join that is a candidate for having positive weight in the optimal packing. We start with some properties of these T-joins.

Lemma 1. *Let $\delta(S)$ be a minimum T-cut, then every T-join with positive weight in an optimal packing intersects $\delta(S)$ in exactly one edge.* □

Lemma 2. *Let $\delta(S)$ be a minimum T-cut and $e \in \delta(S)$ with $c(e) > 0$. Let $\{U_i\}$ be the set of T-joins in an optimal packing with weights $y(U_i) > 0$ for all i. Then there is at least one T-join U_i such that $U_i \cap \delta(S) = \{e\}$. Moreover*

$$c(e) = \sum_{U_i \,:\, e \in U_i} y(U_i)$$

□

Lemma 3. *Let $\delta(S)$ be a minimum T-cut. Let G' be the graph obtained by shrinking S to a single node and giving it the label T. Let G'' be the graph obtained by shrinking $V \setminus S$ to a single node and giving it the label T. An optimal packing of T-joins in G can be obtained by combining the elements of an optimal packing in G' with the elements of an optimal packing in G''.*

Proof. Clearly $\lambda(G') = \lambda(G'') = \lambda(G)$. Let $\{U'_i\}$ be the family in an optimal packing in G' with weights $y'(U'_i) > 0$ for all i. Let $\{U''_j\}$ be the family in an optimal packing in G'' with weights $y''(U''_j) > 0$ for all j. Consider $e \in \delta(S)$ with $c(e) > 0$. We have

$$c(e) = \sum_{U'_i \,:\, e \in U'_i} y'(U'_i) = \sum_{U''_j \,:\, e \in U''_j} y''(U''_j).$$

Thus if U'_i and U''_j contain the edge e then their union gives a T-join U. We give it the weight $y(U) = \min\{y'(U'_i), y''(U''_j)\}$, then the value $y(U)$ is subtracted from $y'(U'_i)$ and $y''(U''_j)$ and if any of these weights becomes 0, the corresponding T-join is removed. This is repeated for any pair with positive weights containing the edge e.

Then this procedure is applied for every edge $e \in \delta(S)$. □

Given the family Φ of tight sets we need to find a T-join U such that $|U \cap \delta(S)| = 1$, for all $S \in \Phi$. Lemma 3 suggests that the graph should be decomposed using minimum T-cuts and it shows how to combine T-joins from the pieces. The procedure is described below.

The first time we start with $S = V$ and define G_S as the subgraph induced by S, with every maximal set of Φ that is properly contained in S contracted, labeled as a T-node and marked as *tight*. Let T_S be the set of T-nodes in G_S. We define an auxiliary graph whose node set is T_S, this is a complete graph. For any two nodes in T_S we find a path in G_S between them of minimum cardinality. Tight nodes can be the beginning or the end of a path, but not an intermediate node. This is to ensure that the resulting T-join intersects exactly once every tight T-cut. The cardinality of this path becomes the weight of the corresponding edge in the auxiliary graph. We give infinite weight if the path does not exist. We find a minimum weight perfect matching in the auxiliary graph. This is to ensure that the resulting T-join is minimal. In G_S we take the union of all paths whose corresponding edges are in the matching. This gives a T-join U' in G_S. Every tight node has exactly one edge of U' incident to it. Lemmas 1 and 2 show that a matching of finite weight exists, any T-join with positive weight in an optimal packing produces a matching of finite weight in the auxiliary graph.

Then we have to deal with each set W that has been contracted. In the T-join above, there is exactly one edge $e = \{i, j\}$, with $j \in W$. This time G_S is the subgraph induced by W plus the edge e, and the node i labeled as a T-node. The idea is to find a T-join U'' in this new graph and combine U' and U'' as in Lemma 3. Again every maximal set of Φ that is properly contained in $S = W \cup \{i\}$ is contracted and we proceed as above.

The algorithm continues recursively. This produces a T-join U that is a candidate to appear in an optimal packing, its weight α_U is obtained as in the next section.

The complexity of finding a minimum weight perfect matching in a complete graph with t nodes is $O(t^3)$, see [7, 12]. Also the complexity of finding all shortest paths in G_S is $O(t^3)$. Therefore the complexity of Step 1 is $O(n^3)$.

5. Finding α_U in Step 2

Given a T-join U we are going to compute the maximum value of α such that

$$\lambda(G - \alpha U) = \lambda(G) - \alpha, \quad \text{and} \quad 0 \le \alpha \le \mu(U).$$

Let us define $f(\alpha) = \lambda(G - \alpha U)$. The function f is the minimum of a set of affine linear functions, so it is concave and piecewise linear. We have to find its first breakpoint. For this we start with a tentative value $\alpha_U = \mu(U)$. We compute $f(\alpha_U)$, if $f(\alpha_U) = \lambda(G) - \alpha_U$ we are done, otherwise let $\delta(S)$ be a minimum T-cut in $G - \alpha_U U$. Let $k = |U \cap \delta_G(S)|$. Notice that here we use $\delta_G(S)$ because in $G - \alpha_U U$ we might have had deleted some edges with capacity 0. Let $\bar{\alpha}$ be the solution of $\lambda(G) - \alpha = \theta(S) - k\alpha$. We set $\alpha_U \leftarrow \bar{\alpha}$ and continue. See Figure 2.

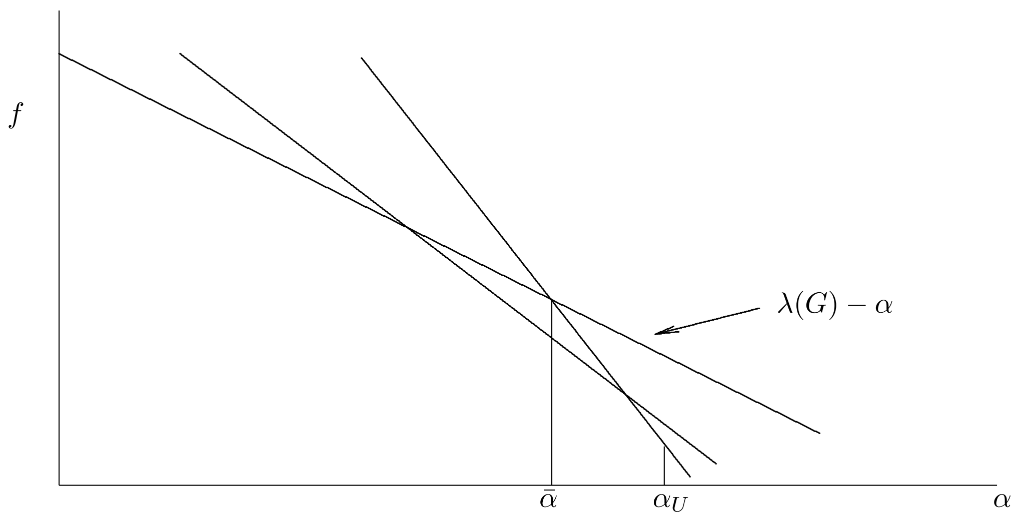

Figure 2

A formal description of this algorithm is below.

Find α_U

Step 0. Set $\alpha_U \leftarrow \mu(U)$.
Step 1. Find a minimum T-cut $\delta(S)$ in $G - \alpha_U U$. If $\lambda(G - \alpha_U U) = \lambda(G) - \alpha_U$ stop. Otherwise continue.
Step 2. Compute $\bar{\alpha}$ as the solution of of $\lambda(G) - \alpha = \theta(S) - k\alpha$.
Where $k = |U \cap \delta_G(S)|$.
Step 3. Set $\alpha_U \leftarrow \bar{\alpha}$ and go to Step 1.

The complexity of this algorithm is given below.

Lemma 1. *If $\alpha_U = \mu(U)$ this algorithm requires $O(n)$ minimum st-cut computations, otherwise it requires $O(n^2)$ minimum st-cut computations.*

Proof. If $\alpha_U = \mu(U)$ only one iteration is performed. Otherwise at each iteration the value of $k = |U \cap \delta_G(S)|$ decreases. Since $|U| \leq n-1$, the above algorithm takes at most $n-1$ iterations. At each iteration one has to find a minimum T-cut with Padberg-Rao's algorithm, this requires $n-1$ minimum st-cut computations, then the result follows. □

6. Final Analysis

Clearly the running time of the algorithm in Section 3 is dominated by the running time of Steps 1 and 2. Also notice that at most $2n-1+m$ iterations are performed. Thus the total running time of Step 1 is $O((n+m)n^3)$. For Step 2 there are at most m iterations where $\alpha_U = \mu(U)$ that require $n-1$ minimum st-cuts, and at most $2n-1$ iterations that require at most $(n-1)^2$ minimum st-cuts. The complexity of finding a minimum st-cut is $O(n^3)$, see [1]. Thus the total running time of Step 2 is $O((mn+n^3)n^3)$. Therefore the complexity of this algorithm is $O(n^6)$.

Since at each iteration one new T-join is produced, we have the following.

Theorem 1. *There is an optimal packing with at most $2n-1+m$ T-joins having a positive weight.* □

A vector $\bar{x}$ satisfying (2) and (3) can be decomposed into $\bar{x} = g + h$, where g is a convex combination of incidence vectors of T-joins and h is a nonnegative vector. This convex combination can be obtained as follows. Use the values $\bar{x}(e)$ as capacities and find an optimal packing of T-joins. Let $\{U_i\}$ be the family of T-joins with weights $y(U_i) > 0$. Let $\alpha = \sum y(U_i)$. Set $y'(U_i) = y(U_i)/\alpha$ for all i, then the vector g is

$$g = \sum y'(U_i) x^{U_i}.$$

Acknowledgments. I am grateful to an anonymous referee for several helpful comments.

References

[1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network flows. Theory, algorithms, and applications*, Prentice Hall, Englewood Cliffs, NJ, 1993.
[2] F. Barahona, *Planar multicommodity flows, max cut, and the chinese postman problem*, in DIMACS Ser. Discrete Math. Theoret. Comput. Sci., 1, RI, 1990, Amer. Math. Soc., pp. 189–202.
[3] J. Cohen and C. Lucchesi, *Minimax relations for T-join packing problems*, in Fifth Israeli Symposium on Theory of Computing and Systems (ISTCS'97), 1997, pp. 38–44.
[4] M. Conforti and E. Johnson, *Two min-max theorems for graphs not contractible to a 4-wheel*, Technical report, IBM T. J. Watson Research Center, Yorktown Heights, NY, 1987.
[5] J. Edmonds and E. L. Johnson, *Matching, euler tours and the chinese postman*, Math. Programming, 5 (1973), pp. 88–124.
[6] D. R. Fulkerson, *Blocking and anti-blocking pairs of polyhedra*, Math. Programming, 1 (1971), pp. 168–194.
[7] H. N. Gabow, *An efficient implementation of Edmonds' algorithm for maximum matching on graphs*, J. Assoc. Comput. Mach., 23 (1976), pp. 221–234.
[8] H. N. Gabow and K. S. Manu, *Packing algorithms for arborescences (and spanning trees) in capacitated graphs*, Math. Programming Ser. B, 82 (1998), pp. 83–109.
[9] R. E. Gomory and T. C. Hu, *Multi-terminal network flows*, J. Soc. Indust. Appl. Math., 9 (1961), pp. 551–570.

[10] M. Grötschel, L. Lovász, and A. Schrijver, *Geometric algorithms and combinatorial optimization*, Springer-Verlag, Berlin, 1993.

[11] I. Holyer, *The NP-completeness of edge-coloring*, SIAM J. Comput., 10 (1981), pp. 718–720.

[12] E. L. Lawler, *Combinatorial optimization: networks and matroids*, Holt, Rinehart and Winston, New York-Montreal, 1976.

[13] M. W. Padberg and M. R. Rao, *Odd minimum cut-sets and b-matchings*, Math. Oper. Res., 7 (1982), pp. 67–80.

[14] P. D. Seymour, *On multicolourings of cubic graphs, and conjectures of Fulkerson and Tutte*, Proc. London Math. Soc., 38 (1979), pp. 423–460.

[15] ———, *On odd cuts and plane multicommodity flows*, Proc. London Math. Soc., 42 (1981), pp. 178–192.

[16] P. G. Tait, *On the colouring of maps*, Proc. Roy. Soc. Edinburgh, 10 (1878), pp. 501–503.

(F. Barahona) IBM T. J. Watson research Center, Yorktown Heights, NY 10589, USA

E-mail address, F. Barahona: `barahon@us.ibm.com`