An Algorithm for Determining a Maximum Packing of Odd Cuts and its Applications

A.V.Karzanov

Institute for Systems Studies
9, Prospect 60 Let Oktyabrya, 117312 Moscow, USSR

Abstract. We design an algorithm for finding a maximum packing of T-cuts and a minimum T-join in an edge-weighted undirected graph $G = (VG, EG)$ with a distinguished subset $T \subseteq VG$ of even cardinality. The running time of the algorithm is $O(pm \log n + p^3 \log p)$, where $n = |VG|$, $m = |EG|$ and $p = |T|$. Applications of the algorithm include the Chinese postman problem, the multicommodity flow problem in a planar graph, and the problem of finding a negative circuit in an undirected edge-weighted graph.

1. Introduction

Throughout the paper by a graph we mean a finite undirected graph without loops and multiple edges. The vertex-set and the edge-set of a graph H are denoted by VH and EH, respectively; an edge with end vertices u and v is denoted by uv. A *chain*, or an $s-t$ *chain*, of a graph is a subgraph L in it such that $VL = \{s = v_0, v_1, \ldots, v_k = t\}$ and $EL = \{v_{i-1}v_i \mid i = 1, \ldots, k\}$. A connected subgraph all the vertices of which have valency 2 is called a *circuit*.

We shall deal with a connected graph G whose edges $e \in EG$ have nonnegative rational-valued weights (*lengths*) $l(e) \in \mathbf{Q}_+$ and with a subset $T \subseteq VG$ of *even* cardinality $|T|$, called the set of *terminals* in G.

A subgraph J in G is called a *T-join* if the set of odd valency vertices of J is exactly T (such a definition slightly differs from that introduced in [Se2] since we admit circuits in J). Clearly a T-join can be represented as the union of pairwise edge-disjoint chains and circuits so that the ends of these chains are distinct and form the set T. Originally T-joins appeared in connection with the so-called "Chinese postman problem" [Me,Ed] that consists in determining a closed route of minimum length in G passing through each edge at least once. The length of such a route is equal to $l(EG) + l(EJ)$, where J is a minimum length T'-join for T' to be the set T' of odd valency vertices of G. (For a subset $S' \subseteq S$ and a mapping $g : S \to \mathbf{Q}$, $g(S')$ denotes $\sum(g(e) \mid e \in S')$.)

There is a minimax relation between T-joins and packings of special cuts of G. More precisely, for $X \subseteq VG$ let $\delta X = \delta^G X$ denote the set of edges of G with one end in X and the other in $VG - X$. We say that $X \subset V$ is an *odd-terminus set* if $|X \cap T|$ is odd; the cut δX for such an X is usually called a *T-cut* [Se3]. Let $D(G,T)$ denote the set of odd-terminus sets for G and T. When $V = T$, we say that $X \in D(G,T)$ is an *odd* set. For a collection $D' \subseteq 2^{VG}$ of subsets of vertices in G, we call a mapping $f : D' \to \mathbf{Q}_+$ an *l-packing* of D' if the corresponding "cuts" weighted by f satisfy the packing condition:

$$(1) \qquad \lambda^f(e) := \sum(f(X) \mid X \in D', e \in \delta^G X) \le l(e) \quad \text{for all } e \in EG.$$

An l-packing f is *maximum* (for given D') if the value $1 \cdot f := \sum(f(X) \mid X \in D')$ is as great as possible.

A simple fact is that a subgraph J of G is a T-join if and only if $|EJ \cap \delta X|$ is odd for all $X \in D(G,T)$. This implies for an l-packing $f : D(G,T) \to \mathbf{Q}_+$ and a T-join J:

$$1 \cdot f \le \sum_{X \in D(G,T)} f(X)(|\delta X \cap EJ|) = \sum_{e \in EJ} \sum(f(X) \mid X \in D(G,T), e \in \delta X)$$
$$= \sum_{e \in EJ} \lambda^f(e) \le l(EJ).$$

Edmonds and Johnson proved that there exist f and J for which the inequalities in this expression hold with equality.

Theorem 1 [EJ] $\max 1 \cdot f = \min l(EJ)$, *where f runs over the l-packings of $D(G,T)$ and J runs over the T-joins in G.*

The proof of this theorem given in [EJ] follows from an algorithm developed there to find optimal f and J. An analysis of this algorithm shows that it can be implemented with running time (counted in elementary arithmetical operations and data transfers) $O(n^4)$ for $n := |VG|$. Whenever l is integer-valued, the algorithm determines an optimal f which turns out to be *half-integral* (an independent proof of the existence of a half-integral optimal packing of T-cuts appeared in [Lo]).

The latter result was strengthened by Seymour as follows. We say that $l \in \mathbf{Z}_+^{EG}$ is *cyclically even* if the length $l(EC)$ of every circuit C in G is even ($\mathbf{Z}_+$ is the set of nonnegative integers).

Theorem 2 [Se3]. *If l is cyclically even then the equality in Theorem 1 is achieved on an integral l-packing f.*

Note that the proof of Theorem 2 given in [Se3] is "non-constructive". On the other hand, in the case of cyclically even l the algorithm from [EJ] guarantees only a half-integral rather than integral optimal packing of T-cuts.

The problem of determining optimal f and J will be denoted $\mathcal{P}(G,T,l)$. In the present paper we describe an algorithm to solve $\mathcal{P}(G,T,l)$ for $l \in \mathbf{Q}_+^{EG}$, in running time $O(pnm + p^4)$, where $m := |EG|$ and $p := |T|$ (Sections 2 and 3). In Section 4 a modification of this algorithm is developed which is based on a dynamic data structure and has running time $O(pm \log n + p^3 \log p)$. The algorithm, as well as its modification, determines an integral optimal f whenever l is cyclically even. This gives an alternative proof of Theorem 2. The algorithm uses the reduction method from [Ka3, Sect. 5] that was developed there to solve a certain larger class of cut packing problems. Namely, the problem $\mathcal{P}(G,T,l)$ in question is reduced to the "smaller" problem $\mathcal{P}(K_T,T,h)$. Here K_T is the complete graph with the vertex-set T, and $h(st)$ is the distance $\mathrm{dist}_l(s,t)$ between terminals $s,t \in T$ in the graph G with length l of edges, that is, $\mathrm{dist}_l(s,t) := \min\{l(EL) \mid L$ is an $s-t$ chain in $G\}$. The fact that h is a metric implies that $\mathcal{P}(K_T,T,l)$ is, in essense, a variant of the minimum weight perfect matching problem, as we shall explain in Section 3; therefore it can be solved by use of alternating chains techniques.

Two applications of the problem in question are well-known.

I. Suppose that U is a distinguished subset of edges of G, and $\mathcal{X}$ is the collection of all sets $X \subset VG$ such that $|\delta X \cap U| = 1$. Seymour studied the problem of the existence of an l-packing $f' : \mathcal{X} \to \mathbf{Q}_+$ satisfying the equality $\lambda^{f'}(e) = l(e)$ for all $e \in U$ (the problem $\mathcal{A}(G,U,l)$).

Theorem 3 [Se1]. *$\mathcal{A}(G,U,l)$ has a solution if and only if the inequality*

$$l(EC \cap U) \le l(EC - U) \tag{2}$$

holds for any circuit C in G; in other words, when the graph G with edges weighted as $w(e) := l(e)$ for $e \in EG - U$ and $w(e) := -l(e)$ for $e \in U$ has no circuit of negative w-length.

The problem $\mathcal{A}(G,U,l)$ is immediately reduced to the problem $\mathcal{P}(G,T,l))$ with T to be the set of vertices in G covered by an odd number of edges from U. More precisely, let f and J be optimal solutions for the latter problem. Since U generates a T-join for given T, one has $l(EJ) \le l(U)$. If $l(EJ) < l(U)$ then $\mathcal{A}(G,U,l)$ has no solution (since the subgraph induced by the edge set $(EJ - U) \cup (U - EJ)$ obviously contains a circuit C which violates (2)). But if $l(EJ) = l(U)$ then f determines a solution of $\mathcal{A}(G,U,l)$ (since $1 \cdot f = l(U)$ easily implies that: (i) $|\delta X \cap U| = 1$ whenever $X \in D(G,T)$ and $f(X) > 0$, and (ii) $\lambda^f(e) = l(e)$ for all $e \in U$). Theorem 2 implies also that if l is cyclically even and $\mathcal{A}(G,U,l)$ is solvable then it has an integral solution [Se3] (note that, as it was shown in [Ka4, Sect. 8], this, stronger, version of Theorem 3 can be derived directly from Theorem 3 itself).

Thus the algorithm can be apply to solve the problem $\mathcal{A}(G,U,l)$ and, as a consequence, to recognize a circuit of negative length in an undirected edge-weighted graph. In the latter case, the running time of the algorithm is $O(\min\{pm\log n, pn^2\} + p^3 \log p)$, where p is the number of vertices covered by an odd number of edges of negative weight; the time estimate becomes smaller because the algorithm does constructs no packing of cuts.

II. Let us be given a planar graph G (explicitly embedded in the plane), a subset U of its edges, a vector $c \in \mathbf{Q}_+^{EG-U}$ of edge *capacities*, and a vector $d \in \mathbf{Q}_+^U$ of *demands*. It is required to find a multicommodity flow $\{F_u \mid u \in U\}$ such that:

(i) F_u is a flow in the the graph $(VG, EG-U)$ which connects the ends of the edge $u \in U$ and has value $d(u)$, and (ii) the total flow through an edge $e \in EG-U$ does not exceed $c(e)$. Let G^* denote the planar graph dual to G, and U^* be the set of edges of G^* corresponding to U. Define $l(e^*)$ to be $c(e)$ for $e \in EG-U$ and to be $d(e)$ for $e \in U$, where e^* denotes the edge in G^* corresponding to $e \in EG$. One can see that the above multicommodity flow problem is equivalent to $\mathcal{A}(G^*, U^*, l)$, and hence Theorems 2 and 3 imply the following result.

Theorem 4 [Se3]. *For G, U, c, d as above, a required multicommodity flow exists if and only if the cut condition*

$$c(\delta X - U) - d(\delta X \cap U) \geq 0 \tag{3}$$

holds for any $X \subset VG$. Moreover, if c and d are integer-valued and the value of the left hand side in (3) is nonnegative and even then the problem has an integral solution.

2. Reduction

For $x, y \in T$, $x \neq y$, the unordered pair xy will be identified with the edge connecting x and y in the complete graph K_T.

The algorithm for solving $\mathcal{P}(G, T, l)$ consists of three *stages*.

The *first stage* is to determine the distances $\mathrm{dist}_l(s,t)$ for all $s, t \in T$. It takes $O(pn^2)$ time assuming that a shortest path procedure of complexity $O(n^2)$ is used, e.g., the Dijkstra's method.

Let h denote the restriction of the distance function dist_l on EK_T. Notice that if l is cyclically even then h is cyclically even as well (h concerns K_T).

The *second stage*, the core of the algorithm, will be described in Section 3. The aim of this stage is to solve the reduced problem $\mathcal{P}(K_T, T, h)$. More precisely, we shall construct an h-packing $g : D(K_T, T) \to \mathbf{Q}_+$ of *odd* sets in K_T and a T-join of a special form, namely, a perfect matching M in K_T so that

$$1 \cdot g = h(M). \tag{4}$$

(Recall that a *matching* in a graph is a subset of its edges all ends of which are distinct; a matching is *perfect* if it covers all vertices of the graph; clearly the subgraph induced by a perfect matching in K_T is a T-join.)

Define $Q := Q(g) := \{A \in D(K_T, T) \mid g(A) > 0\}$. The function g that will be found at the second stage satisfies the following properties:

(5) g is integral whenever h is cyclically even;

(6) for any distinct $A, B \in Q$, either $A \subset B$, or $B \subset A$, or $A \cap B = \emptyset$.

The aim of the *third stage* is to transform g and M to an l-packing $f : D(G,T) \to \mathbf{Q}_+$ and T-join J in G such that

$$1 \cdot f = 1 \cdot g \quad \text{and} \quad l(EJ) = h(M).$$

This and (4) imply $1 \cdot f = l(EJ)$, and hence, f and J give an optimal solution of $\mathcal{P}(G,T,l)$.

We now describe the third stage. A required T-join J is formed in a natural way. Namely, for each $st \in M$ we choose an $s-t$ chain L_{st} in G with $l(L_{st}) = h(st)$; one may take for L_{st} the shortest chain found at the first stage of the algorithm. Let J be the subgraph induced by the edges in G occurring in odd number of these chains. It is easy to see that J is a T-join, and $l(EJ) \le h(M)$ (actually this inequality holds with equality).

Finding a required packing f is a bit more complicated. This is solved by the following algorithm (it does no matter for the algorithm that Q consists of odd subsets, but it is important that Q satisfies property (6)).

Algorithm (of constructing f). Choose a *minimal* set A in Q. Define

$$X := \{x \in VG \mid \text{dist}_l(s,x) = 0 \ \text{ for some } s \in A\};$$

and

$$a := \min\{g(A), \min\{l(e) \mid e \in \delta X\}\}.$$

Put $f(X) := a$, and change g and l by putting $g(A) := g(A) - a$ and $l(e) := l(e) - a$ for $e \in \delta X$ (preserving the old values of g and l on the other elements). By the definition of a, the new l and g are nonnegative. If $g(A)$ becomes 0, remove A from Q. Repeat these steps until the current Q becomes empty.

Let f be the resulting function (extended by zero to the sets $X \in D(G,T)$ not appeared on the steps of the algorithm). It follows from the definition of a that f is an l-packing (for the initial l), that is, f satisfies (1). We prove that

(7) $$g(A) = \sum (f(X) \mid X \in D(G,T), X \cap T = A)$$

holds for each $A \in Q$, which implies $1 \cdot f = 1 \cdot g$. The proof is divided into several claims.

Claim 1. $X \cap T = A$.

*Proof.*If $x \in A$ then $\text{dist}_l(x,x) = 0$ implies $x \in X$, while if $x \in T - A$ then for any $s \in T$ we have

$$\mathrm{dist}_l(s,x) = h(s,x) \geq \lambda^g(sx) = \sum(g(B) \mid B \in Q, sx \in \delta B) \geq g(A) > 0,$$

therefore, $x \notin X$. •

In particular, it follows from Claim 1 that every set X occurring in the algorithm belongs to $D(G,T)$.

Claim 2. *Let l' and g' be the functions obtained from current l and g as a result of one step. Let $\lambda^g(pq) \leq \mathrm{dist}_l(p,q)$ for all $p,q \in T$. Then $\lambda^{g'}(pq) \leq \mathrm{dist}_{l'}(p,q)$ for all $p,q \in T$ (in the other words, g' is an h'-packing if g is an h-packing, where h' is the distance function in K_T with respect to l').*

*Proof.*Put $\lambda := \lambda^g$ and $\lambda' := \lambda^{g'}$. One has to prove that for a pq-chain L in G with $p,q \in T$,

(8) $$l'(EL) \geq \lambda'(pq).$$

Apply induction on $k(L) := |EL \cap \delta X|$ (for any p, q and L).

(i) If $k(L) = 0$ then $l'(EL) = l(EL) \geq \mathrm{dist}_l(p,q) \geq \lambda(pq) = \lambda'(pq)$.

(ii) Let $k(L) = 1$. Then exactly one of p and q is in A. We have $l'(EL) = l(EL) - a$ and $\lambda'(pq) = \lambda(pq) - a$, and (8) follows.

(iii) If $p,q \in A$ then, by (6) and the minimality of A, $\lambda(pq) = \lambda'(pq) = 0$.

(iv) Suppose we are in a case different from (i)-(iii). Then L contains a vertex x such that $x \in X$, $k(L') \geq 1$ and $k(L'') \geq 1$, where L' and L'' are the parts of L from p to x and from x to q, respectively. By the definition of X, there is a terminal $s \in A$ such that $\mathrm{dist}_l(sx) = 0$. Choose an $s-x$ chain P with $l(EP) = 0$; clearly, $VP \subseteq X$. Let L_1 be a $p-s$ chain in the graph $L' \cup P$ and L_2 be an $s-q$ chain in the graph $L'' \cup P$. Obviously, $k(L') = k(L_1) < k(L)$ and $k(L'') = k(L_2) < k(L)$, whence by induction $l'(EL_1) \geq \lambda(ps)$ and $l'(EL_2) \geq \lambda(sq)$. We have

$$l'(EL) = l'(EL_1) + l'(EL_2) \geq \lambda'(ps) + \lambda'(sq) \geq \lambda'(pq)$$

(the latter inequality follows from the fact that if $B \subset T$ and $pq \in \delta B$ then $\{ps, sq\} \cap \delta B \neq \emptyset$), as required. •

Claim 3. *Let the same set $A \in Q$ be chosen on i-th and j-th steps of the algorithm, $i < j$, and let X_i and X_j be the sets determined on these steps respectively. Then $X_i \subset X_j$.*

*Proof.*Since l can only decrease during the algorithm, we have $X_i \subseteq X_j$. Moreover, this inclusion is strict because, after the i-th iteration, $g(A)$ remains positive and therefore $l(xy)$ becomes 0 for some $x \in X_i$ and $y \in VG - X_i$ (by the definition of a), whence $y \in X_j$. •

Claims 1-3 prove correctness of the algorithm for the third stage. Claim 3 shows that the algorithm finishs in no more than $n|Q|$ steps, and that all sets X found on the steps are distinct. Now Claim 2 implies (7).

To estimate the number of steps of the algorithm, note that the cardinality of Q is bounded by a linear function in the number of terminals. Namely, the following claim is easily proved by induction on $|T|$.

Claim 4. *If $Q \subset 2^T$ satisfies (6) then $|Q| \le 2|T| - 3$.*

Thus the algorithm for the third stage consists of $O(pn)$ steps. Obviously, a step can be designed to take $O(m)$ operations. This gives the running time of the third stage to be $O(pnm)$. (Note also that, by Claim 3, the sets X found for the same $A \in Q$ can be stored as corresponding initial parts of a certain list of vertices. This enables us to decrease space needed to write the output of the algorithm.)

3. Algorithm for the Reduced Problem

In what follows the graph K_T and its edge set EK_T will be briefly denoted by K and E, respectively. Current objects of the algorithm solving the problem $\mathcal{P}(K,T,h)$ (the second stage of the algorithm for $\mathcal{P}(G,T,l)$) that we develope in this section will be a (not necessarily perfect) matching $M \subseteq E$, a collection $D \subseteq D(K,T)$ of odd sets in K, and an h-packing $g \in \mathbf{Q}_+^D$ satisfying (5) and (6).

An edge $e \in E$ is called *saturated* (with respect to g) if $\lambda^g(e) = h(e)$. Let M^A be the set of edges in M with both ends in a subset $A \subseteq T$. The following properties will be maintained during the algorithm:

(9) all edges in M are saturated;

(10) D is *regular*; this means that:
(i) for any distinct $A, B \in D$, either $A \subset B$ or $B \subset A$ or $A \cap B = \emptyset$;
(ii) $|M^A| = (|A| - 1)/2$ for any $A \in D$.

Let r_A denote the unique vertex in $A \in D$ not covered by M^A (the *root* of A). One can see that (9) and (10) imply the following properties of M:

(11) $|M \cap \delta A|$ is at most 1 for any $A \in D$, and it is exactly 1 if and only if r_A is covered by M;

(12) if M is perfect then $1 \cdot g = h(M)$.

The algorithm consists of $|T|/2$ *iterations*. Initially one puts $M := \emptyset$ and $D := \emptyset$. An iteration starts with choosing a vertex $r \in T$ not covered by the current M. The purpose of the iteration is to transform the current M, D and g (preserving validity of (9)-(10)) in such a way that r becomes covered by M. As soon as M becomes perfect the current g (extended by zero on $D(G,T)-D$) and M turns into an optimal solution of $\mathcal{P}(K,T,h)$, by (12). Note that (10)(i) provides validity of (6).

We need some terminology and notations.

1) Let $\mathcal{V}=\mathcal{V}^D$ be the set whose elements are the vertices of K and the sets of D. Define a partial order $\prec$ on $\mathcal{V}$ by setting $v\prec v'$ if either $v,v'\in D$ and $v\subset v'$, or $v\in T$, $v'\in D$ and $v\in v'$ (in particular, $s\prec\{s\}$ if $s\in T$ and $\{s\}\in D$). Note that any two non-comparable elements in $\mathcal{V}$ contain no common vertex, by (10)(i). When $v\prec v'$, we say that v *precedes* v'; if, in addition, there is no v'' such that $v\prec v''\prec v'$, we say that v *immediately precedes* v'.

2) For $S\subseteq T$, let W_S denote the set of elements $v\in\mathcal{V}$ such that v is *maximal* provided that either $v\in S$ or v is strictly included in S. For $S\subseteq T$ and $s\in S$, $w_S(s)$ denotes the (unique) element v of W_S for which $s\preceq v$. Let $F_S=F_S^D$ be the multigraph on the vertex set W_S in which elements $v,v'\in W_S$ are connected by k edges, where k is the number of *saturated* edges $st\in E$ with $v=w_S(s)$ and $v'=w_S(t)$. The edge in F_S corresponding to $st\in E$ will be denoted by $\tau_S(st)$. A vertex v in F_S is called *simple* if $v\in T$. When $S=T$, we use notations W, $w(s)$, F and $\tau(st)$ for W_S, $w_S(s)$, F_S and $\tau_S(st)$, respectively.

3) In the algorithm we shall deal with (current) multigraphs F and F_A, $A\in D$. Let $M(F)$ $(M(F_A))$ denote the set of edges e of F (F_A) such that $\tau^{-1}(e)$ belongs to the matching M.

The property (11) and the fact that each set in D has odd cardinality imply:

(13) $|W|$ is even, and $|W_A|$ is odd for all $A\in D$;

(14) $M(F)$ $(M(F_A))$ is a matching in F (in F_A).

A chain L in F (F_A) is called *alternating* (with respect to M) if it contains $\lfloor |EL|/2\rfloor$ edges in $M(F)$ $(M(F_A))$. During the algorithm the following additional property holds:

(15) for each $A\in D$ with $|A|>1$ there is a circuit C_A in F_A which passes through all the vertices of F_A and contains $(|W_A|-1)/2$ edges in $M(F_A)$.

For $v\in W_A$ the chain in C_A that joins v with $w(r_A)$ and has an even number of edges is denoted by $L_A(v)$ (clearly such a chain is alternating); if $|A|=1$, we put $L_A(v):=(\{r_A\},\emptyset)$).

Iteration. Like the majority of matching algorithms, the main work on the iteration consists in "growing" an alternating tree. We say that a subgraph H in F is an *alternating tree rooted at* $w(r)$ if: (i) $w(r)\in VH$, H is connected and has no circuits; (ii) for each $v\in VH$ the chain in H joining v with $w(r)$ is alternating; this chain is denoted by $L(v)$; (iii) for each one-valency vertex v in H, $L(v)$ has even number of edges. Let VH^+ (VH^-) denote the set of vertices of H for which $|EL(v)|$ is even (respectively, odd), and let $W^0 := W - VH$.

An iteration is a sequence of steps. A *step* is an execution of one of the procedures P0-P5 below. At the beginning of an iteration we put $H := (\{w(r)\}, \emptyset)$ and start with P0.

Procedure P0. Choose an edge e in F with ends u and v such that either (i) $u \in VH^+$ and $v \in W^0$, or (ii) $u, v \in VH^+$. Let $e = \tau(st)$, $u = w(s)$ and $v = w(t)$. In case (i), go to P1 (increasing the matching) if v is not covered by $M(F)$, and go to P2 (increasing the tree) if it is. In case (ii), go to P3 (shrinking an odd circuit). If there is no edge e as above, choose in VH^- a non-simple vertex $A \in D$ with $g(A) = 0$ and go to P4 (destroying a non-simple vertex). If such an A does not exist, go to P5 (changing the packing g).

Procedure P1 (*increasing* M). Let $r' := t$ if $v = t$ (that is, if v is a simple vertex), and $r' := r_A$ if $v = A \in D$; then r' is not covered by M. Add the edge e and the vertex v to the alternating chain $L(u)$ in H, forming an alternating chain L in F connecting the vertices $w(r)$ and $v = w(r')$ (these vertices are not covered by $M(F)$). If L contains a non-simple vertex $A' \in D$, we replace A' by the alternating chain with even number of edges from the circuit $C_{A'}$, forming an alternating chain in $F^{D'}$ which connects vertices not covered by $M(F^{D'})$; here $D' := D - \{A'\}$. Repeat such replacements, one by one, until an alternating $r - r'$ chain $\widetilde{L}$ in the graph K will be obtained. Now change M along $\widetilde{L}$ by putting $M := M \triangle E\widetilde{L}$ ($X \triangle Y$ denotes the symmetric difference $(X - Y) \cup (Y - X)$ of sets X, Y).

Procedure P1 completes the iteration. The resulting M has become larger and the vertices r and r' have been covered by M. One can check that (9),(10) and (15) are true as before.

Procedure P2 (*increasing* H). Let e' be the edge in $M(F)$ incident to v, and let v' be the other end of e'. (It follows from properties of H that $v' \in W^0$.) Expand H by adding the vertices v, v' and the edges e, e'. Return to P0.

Procedure P3 (*shrinking an odd circuit*). Let C be the circuit in the graph H' obtained by adding the edge e to H; C is formed by e and the corresponding parts of the chains $L(u)$ and $L(v)$ in H, and it contains $(|EC|-1)/2$ edges in $M(F)$. Form a new odd set A to be $\{s' \in T \mid w(s') \in VC\}$, and put $D := D \cup \{A\}$, $g(A) := 0$ and $C_A := C$. The new tree H is obtained from H' by shrinking C.

Procedure P4 (*destroying a non-simple vertex*). Let $e = \tau(st)$ and $e' = \tau(s't')$ be the edges in H incident to A, and let $e \in M(F)$. Let for definiteness s and s' are in A; then $s = r_A$. Take the chain $L := L_A(w_A(s'))$ in C_A connecting $w_A(t)$ with the "root" $w_A(r_A)$ of F_A. Delete the set A from D and correct H by replacing the vertex v in it by the chain L. Return to P0.

Define

$$E^{0+} := \{st \in E \mid w(s) \in W^0, w(t) \in VH^+\};$$
$$E^{0-} := \{st \in E \mid w(s) \in W^0, w(t) \in VH^-\};$$
$$E^{++} := \{st \in E \mid w(s), w(t) \in VH^+, w(s) \neq w(t)\};$$
$$E^{--} := \{st \in E \mid w(s), w(t) \in VH^-, w(s) \neq w(t)\}.$$

Procedure P5 (*changing* g). Determine the value $\varepsilon := \min\{\varepsilon^{0+}, \varepsilon^{++}, \gamma\}$, where

$$\varepsilon^{0+} := \min\{h(e) - \lambda^g(e) \mid e \in E^{0+}\};$$
$$\varepsilon^{++} := \frac{1}{2}\min\{h(e) - \lambda^g(e) \mid e \in E^{++}\};$$
$$\gamma := \min\{g(A) \mid A \in VH^- \cap D\}$$

Add to D all the one-element sets $\{s\}$ such that s is a simple vertex in VH^+. Transform g as $g(A) := g(A) + \varepsilon$ for $A \in VH^+$ and $g(A) := g(A) - \varepsilon$ for $A \in VH^- \cap D$. Return to P0.

Correctness and complexity of the algorithm.

Lemma 3.1. *When applying P5, all the vertices in VH^- are non-simple.*

*Proof.*As it follows from the description of P0, when we go from P0 to P5, the current g satisfies:

$$\lambda^g(e) < h(e) \quad \text{for any edge } e \in E^{0+} \cup E^{++} \tag{16}$$

(otherwise one must go from P0 to one of P1-P3 rather than to P5). Suppose that the lemma is not valid, and let v be a simple vertex in VH^-, that is, $v \in T$. Consider edges sv, vt in K such that $\tau(sv), \tau(vt)$ are the edges in H incident to v. Since $\{v\} \notin D$ and $w(s) \neq w(t)$, there is no set $A \in D$ such that $sv, vt \in \delta A$, whence $|\{sv, vt\} \cap \delta A'| = |\{st\} \cap \delta A'|$ for any $A' \in D$. This implies $\lambda^g(st) = \lambda^g(sv) + \lambda^g(vt)$. But $\lambda^g(e) = h(e)$ for $e = sv, vt$, and now we conclude from (16) that $h(st) > h(sv) + h(vt)$, which is impossible because h is a metric. •

Lemma 3.1 and the definition of ε easily imply that the function g resulting in procedure P5 is an h-packing. We leave to the reader to check that M, D, g, H resulting in each of procedures P1-P5 are correct; in particular, (9),(10) and (15) are true for them.

Suppose that the iteration is completed in a finite number of steps, and denote by N_i the number of occurences of procedure Pi on the iteration. Let $N := \sum_{i=0}^{5} N_i$. Then $N_1 \leq 1$ and $N_0 = \sum_{i=1}^{5} N_i$. One can see from the definition of ε in P5 that after application of this procedure at least one of the two possibilities occurs: $\lambda^g(e) = h(e)$ for some $e \in E^{0+} \cup E^{++}$; or $g(A) = 0$ for some non-simple vertex $A \in VH^-$. (The case when all the sets E^{0+}, E^{++}, VH^- are empty is impossible;

otherwise H would consist of a single vertex, and $VH = W$, whence $|W| = 1$ would follow, contrary to (13).) Hence P5 will be immediately followed by P0 and then by one of P1-P4. Thus $N_5 \leq \sum_{i=1}^4 N_i$, and therefore, N is estimated as $O(N_2 + N_3 + N_4)$. To estimate the latter quantity, introduce the sets:

$$\widetilde{T} := \{s \in T \mid w(s) \in VH^+\}; \quad \text{and}$$

$$\widetilde{D} := D - \{A \in D \mid A \preceq v \ \text{ for some } v \in VH^+\}$$

One can observe that:

(i) after every application of P0-P5, the new set $\widetilde{T}$ contains the previous one and the new set $\widetilde{D}$ is contained in the previous one;

(ii) every application of P2 or P3 increases $\widetilde{T}$;

(iii) every application of P4 decreases $\widetilde{D}$.

Thus $N_2 + N_3 \leq |T| - 1$, and N_4 does not exceed the cardinality of D at the beginning of the iteration. By Claim 4 in Section 2, this cardinality is $O(|T|)$. Hence the iteration is terminated after $O(p)$ steps.

In order to estimate the running time of the iteration we need to specify data structures used in it. The elements $A \in D$ are given in the current set $\mathcal{V}$ as identificators (references) rather than subsets of T. Elements $v, v' \in \mathcal{V}$ such that v immediately precedes v' are joined by references to each other; such references define structure of a (directed) forest on $\mathcal{V}$. The multigraphs F, F_A ($A \in D$) and the tree H are designed in a natural way.

Clearly each of procedures P1-P4 can be executed using $O(p)$ operations. P0 consists, in fact, in examination of vertices and edges of F (and/or F_A's) and it takes $O(p^2)$ operations. When P5 applies, we have to calculate efficiently the values $\lambda^g(e)$ for $e \in E^{0+} \cup E^{++}$, required for determining ε and for correction of the edge-set of F (according to the new g). To do this, define for $v \in \mathcal{V}$ the set $Vscr(v)$ to be $\{v\} \cup \{v' \in \mathcal{V} \mid v' \prec v\}$ and the set $T(v)$ to be $T \cap Vscr(v)$; define the value $\rho(v) := \rho^g(v)$ to be $\sum\{g(A) \mid A \in D, v \preceq A\}$. One can see that $\lambda^g(st) = \rho(s) + \rho(t)$ for any $s, t \in T$ such that $w(s) \neq w(t)$. Note also that, for fixed $v \in W$, the numbers $\rho(v')$ can be recursively calculated for all $v' \in Vscr(v)$ using $O(|V(v)|)$, or $O(|T(v)|)$, operations. Hence, for $u, v \in W$, $u \neq v$, determining the values $\lambda^g(st)$ for all edges st in the set $S := \{st \in E \mid s \in T(u), t \in T(v)\}$ takes $O(|T(u)||T(v)|)$, or $O(|S|)$, operations. This gives the estimate $O(p^2)$ for the running time of P5.

Thus the iteration can be executed within $O(p^3)$ operations, whence the running time for the algorithm to solve $\mathcal{P}(K_T, T, h)$ is $O(p^4)$. This implies that the running time of the algorithm for the initial problem $\mathcal{P}(G, T, l)$ is exactly as mentioned in the Introduction.

It remains to show that whenever h is cyclically even the algorithm finds an integral optimal h-packing. It suffices to prove that when h is cyclically even and P5 applies to an integral g, the resulting function g' will be integral too. In other words, one has to prove that the value ε^{++} defined in the description of P5 is an integer. To see this, consider arbitrary $s, t \in T$ such that $w(s), w(t) \in VH^+$ and $w(s) \neq w(t)$. Let C be the circuit in F formed by the edge $\tau(st)$ and the corresponding parts of the chains $L(w(s))$ and $L(w(t))$. Replacing successively non-simple vertices in C by appropriate alternating chains (in a similar way as in P1) we get a circuit $\widetilde{C}$ in K all of whose edges except st are saturated by g. Then

$$h(st) - \lambda^g(st) = h(E\widetilde{C}) - \lambda^g(E\widetilde{C}) = h(E\widetilde{C}) - \sum_{A \in D} g(A)|E\widetilde{C} \cap \delta A|$$

Now evenness of $h(E\widetilde{C})$ and $|E\widetilde{C} \cap \delta A|$ together with integrality of g imply that $h(st) - \lambda^g(st)$ is even. Hence, ε^{++} is an integer.

4. A faster modification

In this section we describe a faster modification of the above algorithm. It is based on certain dynamic data structures. As a consequence, the running time of the second stage becomes $O(p^3 \log p)$ (instead of $O(p^4)$) and that of the third stage becomes $O(pm \log n)$ (instead of $O(pnm)$).

More precisely, in the modification we have to search, as fast as possible, for a minimal element in a dynamic ordered set. Formally, the problem can be stated as follows. Suppose we are given a current set S whose elements $e \in S$ have rational weights $a(e)$. At moments $1, \ldots, N$ the set S is changed by removing some of its elements and inserting new ones. At certain moments it is required to find an element e in the current set such that $a(e)$ is minimum. How fast can this be done? One approach to do this is to apply a well-known way of data design, the so-called AVL-tree [AL]. When S is designed as an AVL-tree, the above task can be executed with the total amount of operations to be $O(\eta \log \omega)$, where η is the cardinality of the initial S plus the number of elements inserted at moments $1, \ldots, N$, and ω is the maximum cardinality of current S (note that for our purposes one can use the method developed in [Ka1] which is simpler to implement but requires $O(\eta \log \eta)$ operations). We shall use the term "order structure" for a set S together with a design of it which enables to execute the above task in time at most $O(\eta \log \eta)$.

First of all we explain how to modify the third stage of the algorithm, using notations from Section 2. One may assume that the steps on which the same set $A \in Q$ is chosen go in succession. We arrange the set of edges of the cut $e \in \delta^G X$ as an order structure $\mathcal{R} = \mathcal{R}(A)$. An element $e \in \mathcal{R}$ has a weight $q(e)$ (these weights determine the ordering in $\mathcal{R}$), and there is a number d associated with $\mathcal{R}$. The numbers $q(e)$ and d are assigned so that for each $e \in \delta^G X$, the current length $l(e)$ is equal to $q(e) - d$. The structure $\mathcal{R}(A)$ is created at the beginning of treatment of A ; at this moment we put $q(e) := l(e)$, $e \in \mathcal{R}(A)$, and $d := 0$.

Suppose that i steps with a given $A \in Q$ have been executed; let $\mathcal{R}_i$ and X_i stand for $\mathcal{R}$ and X, respectively, obtained on the i-th step. Then the new set X_{i+1} is constructed as follows. Firstly, find the set Δ of elements $e \in \mathcal{R}_i$ such that $q(e) - d$ $(= l(e))$ is 0 (obviously, Δ is the set of minimal elements in $\mathcal{R}$). Secondly, find the set Z of vertices $x \in VG - X_i$ such that there is a chain of zero length in G connecting x with a vertex incident to an edge from Δ. Then X_{i+1} is just $X_i \cup Z$.

To get the new $\mathcal{R} = \mathcal{R}_{i+1}$ corresponding to X_{i+1}, one should delete from $\mathcal{R}_i$ the edges with one end in X_i and the other in Z, and add the edges with one end in Z and the other in $V - X_{i+1}$. Each latter edge e is included in $\mathcal{R}$ with the weight $q(e) := l(e) + d$. The number a, defined for given X as in the algorithm of Section 2, is the minimum of values $q(e) - d$ among the elements $e \in \mathcal{R}_{i+1}$. Finally, we correct d as $d := d + a$; this corresponds to decreasing by a the lengths of the edges in δX_{i+1}.

Clearly, while working with the same $A \in Q$, each edge of G can be included in $\mathcal{R}$ at most once. Hence the total number of operations to handle with $\mathcal{R}$ during this period is $O(m \log m)$, or $O(m \log n)$. This implies the running time of the third stage to be $O(pm \log n)$, as required.

Now we show how to modify the second stage of the algorithm. The distinctions with what was developed in Section 3 are as follows.

(i) The set E^{++} is desined as an order structure ranged by numbers $b(e)$, $e \in E^{++}$, such that $b(e) - d$ is equal to the "excess" $h(e) - \lambda^g(e)$, where d is a number attached to E^{++} as a whole. Similarly, for each $s \in T$ such that $w(s) \in VH^- \cup W^0$, there is an order structure $\mathcal{R}(s)$ consisting of the edges $st \in E$ with $w(t) \in VH^+$; these edges are ranged in $\mathcal{R}(s)$ by numbers $c(st)$ such that $c(st) - d(s)$ is equal to $h(st) - \lambda^g(st)$, where $d(s)$ is a number attached to $\mathcal{R}(s)$. These order structures are created at the beginning of each iteration, and at this moment the numbers d and $d(s)$ are assigned to be zero.

(ii) The first part of the procedure P0 is executed by examination of minimal elements in the structures E^{++} and $\mathcal{R}(s)$ for $s \in T$ such that $w(s) \in W^0$; we determine whether or not there exists an edge st among them such that the excess $b(st) - d$ (if $st \in E^{++}$) or $c(st) - d(s)$ (if $st \in \mathcal{R}(s)$) is 0. This implies that each occurence of P0 requires running time $O(p)$. Next, when P5 applies, the number ε^{++} is determined as $(b(e) - d)/2$, and ε^{0+} is determined as $\min\{c(st) - d(s) \mid s \in T, w(s) \in W^0\}$, where e (respectively, st) is a minimal element in E^{++} (respectively, $\mathcal{R}(s)$). When changing the current function g in P5, one should correct d and $d(s)$ for $s \in T$ with $w(s) \in W^0$; namely, one has to put $d := d + 2\varepsilon$ and $d(s) := d(s) + \varepsilon$ (this corresponds to increasing $\lambda^g(e)$ by 2ε for $e \in E^{++}$ and by ε for $e \in \mathcal{R}(s)$).

(iii) Each application of the procedures P2-P4 has to be completed with correction of the corresponding order structures. We explain how to correct them for P4 that consists in destroying a non-simple vertex $A \in VH^-$ (for P2 and P3 the corresponding structures are corrected easier and this is left to the reader). Let $\widehat{D} := D - \{A\}$, $\widehat{W} := (W - \{A\}) \cup W_A$ and $\widehat{H}$ be the objects obtained from D, W and H as a result of application of P4. As it was explained earlier, $\widehat{H}$ is formed from H by replacing the vertex A by a chain L from C_A. Define L^+ (resp., L^-) to be the set of vertices v in L such that the part of L from $w_A(r_A)$ to v has even (resp., odd) number of edges. Let $W_A^0 := W_A - VL$. Then

$$V\widehat{H}^+ = VH^+ \cup L^-, \quad V\widehat{H}^- = (VH^- - \{A\}) \cup L^+, \quad \widehat{W}^0 = W^0 \cup W^0_A.$$

Using techniques given in the end of Section 3, determine the values $\rho(s), \rho(t)$ and then the values $\lambda^g(st)$ for all $s, t \in T$ such that $s \preceq v'$ and $t \preceq v''$, where v' runs over the set $V\widehat{H}^- \cup \widehat{W}^0$ and v'' runs over L^-; by arguments in Section 3, this takes the amount of operations proportional to the number of such edges st. Now for each $s \in T$ with $\widehat{w}(s) \in V\widehat{H}^- \cup \widehat{W}^0$, one has to insert in $\mathcal{R}(s)$ all the edges $st \in E$ for which $\widehat{w}(t) \in L^-$, and put $c(st) := h(st) - \lambda^g(st) + d(s)$ ($\widehat{w}(x)$ denotes the maximal element $v \in \widehat{W}$ such that $x \preceq v$). In addition, for each $s \in T$ with $\widehat{w}(s) \in L^-$, each element $st \in E$ for which $\widehat{w}(t) \in VH^+$ has to be transferred from $\mathcal{R}(s)$ to the structure E^{++}, with weight $b(st) := c(st) - d(s) + d$, after that the rest of $\mathcal{R}(s)$ is deleted.

It was noted in Section 3 that the current set $\widetilde{T} := \{s \in T \mid w(s) \in VH^+\}$ is monotonously extended during the iteration. This implies that each edge $st \in E$ can be included at most once in $\mathcal{R}(s)$ or $\mathcal{R}(t)$ and it can be included at most once in E^{++}. Hence, the total amount of operations spent on the iteration to support the above order structures is $O(p^2 \log p)$. This and arguments in (ii)-(iii) give the running time of the iteration to be $O(p^2 \log p)$, and the running time of the second stage to be $O(p^3 \log p)$. Thus the modified algorithm to solve the initial problem $\mathcal{P}(G, T, l)$ has running time as mentioned in the Introduction.

References

[AL] Adel'son-Vel'skii, G.M., Landis, E.M.: One algorithm for the organization of information. *Doklady Akad. Nauk SSSR* **146** (2) (1962) (Russian) (English transl.: *Soviet Math. Dokl.* **3** (1962) 1259-1262).

[Ed] Edmonds, J.: The Chinese postman problem. *Oper. Research* **13** Suppl. (1) (1965), p. 373.

[EJ] Edmonds, J., Johnson, E.L.: Matchings, Euler tours and Chinese postman. *Math. Programming* **5** (1973) 88-124.

[Ka1] Karzanov, A.V.: One dynamic data structure to search for maximal elements in a set and its applications. In: *Studies in Discrete Optimizations*. Nauka, Moscow, 1976, pp. 348-359, (Russian).

[Ka2] Karzanov, A.V.: Combinatorial methods to solve cut-determined multiflow problems. In: *Combinatorial methods for flow problems*. Inst. for System Studies, Moscow, 1979, iss. **3**, pp. 6-69 (Russian).

[Ka3] Karzanov, A.V.: Multicut problems and methods to solve them. *Preprint*. Inst. for System Studies, Moscow, 1982, 63p. (Russian).

[Ka4] Karzanov, A.V.: A generalized MFMC-property and multicommodity cut problems. In: Hajnal, L., Lovász, L., Sos, V.T. (eds.) *Finite and Infinite Sets* (Proc. 6th Hungar. Comb. Coll. (Eger,1981)). Vol. **2**. North-Holland, Amsterdam, 1984), pp. 443-486.

[Lo] Lovász, L.: 2-matchings and 2-covers of hypergraphs. *Acta Math. Acad. Sci. Hungar.* **26** (1975) 433-444.

[Me] Mei-Ko, K.: Graphic programming using odd or even points. *Chinese Mathematics* **1** (1962) 273-277.

[Se1] Seymour, P.D.: Sums of circuits. In: Bondy, J.A., Murty, U.S.R. (eds.) : *Graph theory and related topics*. Acad. Press, NY, 1978, pp. 341-355.

[Se2] Seymour, P.D.: On multi-colouring of cubic graphs, and conjectures of Fulkerson and Tutte. *Proc. Lond. Math. Soc.*, Ser.3 **38**(1979) 423-460.

[Se3] Seymour, P.D.: On odd cuts and planar multicommodity flows. *Proc. Lond. Math. Soc.*, Ser.3 **42**(1981) 178-192.