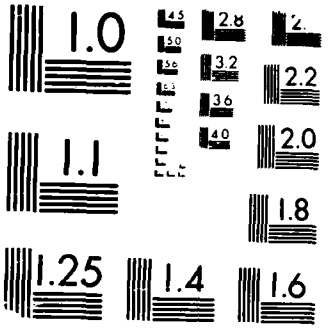
AD-A192 709 A MATROID ALGORITHM AND ITS APPLICATION TO THE EFFICIENT SOLUTION OF TWO.. (U) CARNEGIE-MELLON UNIV PITTSBURGH PA MANAGEMENT SCIENCES RESEAR.. 1/1

UNCLASSIFIED C BREZOVEC ET AL. FEB 88 MSSR-534(R) F/G 12/1 NL




MICROCOPY RESOLUTION TEST CHART

# A MATROID ALGORITHM AND ITS APPLICATION TO THE EFFICIENT SOLUTION OF TWO OPTIMIZATION PROBLEMS ON GRAPHS

by

Carl Brezovec*
Gerard Cornuejols**
and
Fred Glover***

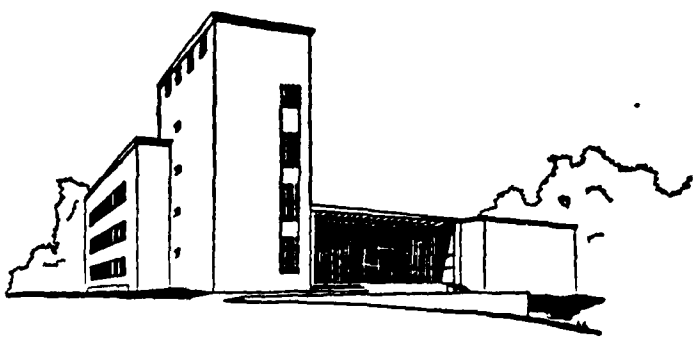
February, 1987
Revised Febraury, 1988

**Carnegie Mellon University**

PITTSBURGH, PENNSYLVANIA 15213

**GRADUATE SCHOOL OF INDUSTRIAL ADMINISTRATION**

WILLIAM LARIMER MELLON, FOUNDER



W.P. No. 29-86-87

Management Science Research Report MSRR 534(R)

# A MATROID ALGORITHM AND ITS APPLICATION TO THE EFFICIENT SOLUTION OF TWO OPTIMIZATION PROBLEMS ON GRAPHS

by

Carl Brezovec*
Gerard Cornuejols**
and
Fred Glover***

February, 1987
Revised Febraury, 1988

* Department of Mathematics
University of Kentucky, Lexington, KY

** Graduate School of Industrial Administration
Carnegie Mellon University, Pittsburgh, PA

*** Graduate School of Business Administration
University of Colorado, Boulder, CI



This report was prepared in part as part of the activities of the Management Sciences Research Group, Carnegie Mellon University, under Contract No. N00014-85-K-0198 NR 047-048 with the Office of Naval Research and in part by NSF Grant ECS 8601660. Reproduction in whole or in part is permitted for any purpose of the U. S. Government.

Management Sciences Research Group
Graduate School of Industrial Administration
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

## 0. Abstract.

We address the problem of finding a minimum weight base B of a matroid when, in addition, each element of the matroid is colored with one of m colors and there are upper and lower bound restrictions on the number of elements of B with color i, for i = 1, 2, ..., m. This problem is a special case of matroid intersection. We present an algorithm that exploits the special structure, and we apply it to two optimization problems on graphs. When applied to the weighted bipartite matching problem, our algorithm has complexity $O(|E|\,|V| + |V|^2 \log |V|)$. Here V denotes the node set of the underlying bipartite graph, and E denotes its edge set. The second application is defined on a general connected graph G = (V, E) whose edges have a weight and a color. One seeks a minimum weight spanning tree with upper and lower bound restrictions on the number of edges with color i in the tree, for each i. Our algorithm for this problem has complexity $O(|E|\,|V| + m^2\,|V| + m\,|V|^2)$. A special case of this constrained spanning tree problem occurs when $V^*$ is a set of pairwise nonadjacent nodes of G. One must find a minimum weight spanning tree with upper and lower bound restrictions on the degree of each node of $V^*$. Then the complexity of our algorithm is $O(|V|\,|E| + |V^*|\,|V|^2)$. Finally, we discuss a new relaxation of the traveling salesman problem.

## 1. Introduction.

Let E be a finite set, M a matroid of rank r defined on E, and $w: E \to R$ a weight function. The weight of a subset $F \subseteq E$ is defined by $w(F) = \sum\{w(e) : e \in F\}$. Let $E_1, E_2, ..., E_m$ be a partition of E into nonempty subsets. To each $E_i$ assign two integers $l_i$ and $u_i$ ($l_i \le u_i$). We address the problem of finding a base B of M which will

minimize w(B)

subject to

$$l_i \le |B \cap E_i| \le u_i \quad \text{for } i = 1, 2, \ldots, m.$$

This problem is denoted by (P). Since a base B cannot contain loops, we assume without loss of generality that M is loopless.

It is known that the set

$$B^* = \{X \subseteq E : l_i \le |X \cap E_i| \le u_i,\ i = 1, 2, \ldots, m,\ |X| = r\}$$

is the family of bases of a matroid M', which has rank r. Such a matroid is called a generalized partition matroid. Thus, (P) is the problem of finding a minimum weight common base of M and M', if one exists.

This problem has been solved in polynomial time by Edmonds ([E1] and [E2]). The algorithm presented here takes advantage of the special structure of the matroid M' and works on an auxiliary digraph with m nodes, instead of the $n = |E|$ nodes required by the general matroid intersection algorithms. This leads to a complexity of $O(r(nc + m \log m))$, where c is the complexity of circuit finding. By circuit finding we mean the following: Given independent set I and $e \in E - I$, list the elements of the unique circuit in $I \cup \{e\}$, or show that $I \cup \{e\}$ is independent. This complexity is to be compared with the $O(nr(r + c + \log n))$ required by the general algorithm (see Brezovec, Cornuéjols, and Glover ([BCG])).

To make this paper self-contained, we state without proofs some of the results of [BCG]. This is done in Section 2. The algorithm for solving (P) is presented in Section 3. We discuss its complexity in Section 4 and some variants of the algorithm and a postoptimizing procedure in Section 5.

Finally, in Sections 6 and 7, we provide two examples of how our algorithm may be applied to classical combinatiorial optimization problems. For the weighted bipartite matching problem on a graph $G = (V, E)$, we get an algorithm of complexity $O(|V|\,|E| + |V|^2 \log |V|)$, which is the best known for weighted bipartite matching (see Fredman and Tarjan [FT]). Note that the general matroid intersection algorithm only gives a bound of $O(|E|\,|V|^2)$.

In Section 7, we consider the case where M is the graphic matroid associated with a connected graph $G = (V, E)$. In this case, our algorithm can be made to run in time $O(|V|\,|E| + |V|^2 m + |V| m^2)$. We also consider the following problem. Let $V^* = \{v_1, v_2, \ldots, v_{m-1}\}$ denote a stable set of G (a set

of pairwise nonadjacent nodes). Let $E_i$ consist of the edges incident with node $v_i$ for $i = 1, 2, ..., m-1$, with $E_m$ containing all remaining edges. A minimum weight spanning tree of $G$ must be found with the property that, for $i = 1, 2, ..., m-1$, the number of edges incident with $v_i$ belongs to the interval $[l_i, u_i]$. This is of complexity $O(|V| |E| + m |V|^2))$. Note that, here as well, the general matroid intersection algorithm only gives a bound of $O(|E| |V|^2)$. If $l_i = u_i = 2$ for $i = 1, 2, ..., m-1$, the complexity can be further reduced to $O(m(|E| + m |V|) + |E| \log |V|)$, and such degree restricted spanning trees can be used to strengthen the 1-tree relaxation of Held and Karp for the Traveling Salesman Problem. We also show how this relaxation can be further strengthened and generalized to the case where $V^*$ is not a stable set.

## 2. Known results

Most of the material in this section can be found in standard references such as [W] (Proposition 1 below is Exercise 2 on page 15 of [W]) or [L] (Theorem 2 below is similar to Theorem 9.4 of Chapter 8 in [L]). However, for the purpose of this paper, we find it more convenient to follow the framework developed in [BCG].

Let $M$ be a matroid defined on the element set $E$, and let $B$ be a base of $M$. Consider $\beta \subseteq B$ and $\beta' \subseteq E - B$ such that $|\beta| = |\beta'|$. We say that $(\beta, \beta')$ is a **B-swap** if $(B - \beta) \cup \beta'$ is a base of $M$. We call $m(\beta, \beta')$ a **matching** if it represents a one-to-one mapping of $\beta$ onto $\beta'$; $m(\beta, \beta')$ is called a **B-matching** if every $(e, e')$ in $m(\beta, \beta')$ is a B-swap.

**Proposition 1.** [BCG, Lemma 2]. Let $B$ and $B'$ be two bases of $M$, and let $\beta = B - B'$ and $\beta' = B' - B$. Then there is at least one B-matching of $(\beta, \beta')$.

Consider two matroids $M_1$ and $M_2$ defined on the same finite element set E, and let $I \subseteq E$ be a k-intersection, that is, let I be independent in $M_1$ and $M_2$ such that $|I| = k$. Let $w: E \to R$ be a weight function. The problem of finding a minimum weight k-intersection is denoted $(P_k)$. To solve $(P_k)$, define a bipartite auxiliary digraph G(I) as follows. The nodes of G(I) are partitioned into V and V'. Construct a node in V for each $e \in I$ and a node in V' for each $e' \in E - I$. For each $e \in I$, we construct an arc (e, e') with weight $w(e, e') = w(e') - w(e)$ provided $(I - \{e\}) \cup \{e'\}$ is independent in $M_1$. Similarly, for each $e' \in E - I$, we construct an arc (e', e) with weight $w(e', e) = 0$ provided $(I - \{e\}) \cup \{e'\}$ is independent in $M_2$.

Let C be a directed cycle in G(I). In this paper, directed cycles (or *dicycles*, for short) can have repeated nodes but no repeated arcs. We define the weight of a dicycle to be the sum of the weights of the arcs in the dicycle.

**Theorem 1.** [BCG, Theorem 3]. Assume I is a k-intersection.

(i) I is optimal for $(P_k)$ if and only if there are no negative weight dicycles in G(I).

(ii) Let C be a negative weight dicycle in G(I) with no negative weight dicycle on a subset of its nodes, and let $\Sigma$ and $\Sigma'$ be the nodes of C in V and V', respectively. Then $I' = (I - \Sigma) \cup \Sigma'$ is a k-intersection such that $w(I') < w(I)$.

This theorem leads to an algorithm for solving $(P_k)$, given an initial k-intersection I: use G(I) to find an improved solution; continue until a G(I) is found which has no negative weight dicycles.

We modify this technique to develop a dual algorithm. Consider a set A of k artificial elements, where $E \cap A = \emptyset$. Let $F = E \cup A$ and define two matroids $M_1(F)$ and $M_2(F)$ on the element set F as follows. For $i = 1, 2$, $I \subseteq E$ and $J \subseteq A$, the set $I \cup J$ is independent in $M_i(F)$ if and only if I is independent in $M_i$. The problem $(P_k)$ relative to these new matroids is denoted $(P_k|\ F)$. In order to define $(P_k|\ F)$ completely we need to assign

weights to the artificial elements. By giving them large negative weights, we can guarantee that A is an optimal k-intersection for $(P_k|\ F)$.

Define the digraph Sp(I) from G(I) by splitting one of the nodes arising from an artificial element, say $\alpha \in A$, into a source node s and a destination node d. Let the arcs out of s in Sp(I) be those out of $\alpha$ in G(I) and the arcs into d in Sp(I) be those into $\alpha$ in G(I). The following result provides the foundation for an algorithm to solve $(P_k)$.

**Theorem 2.** [BCG, Theorem 5]. Let I be optimal for $(P_k|\ H)$, where $E \subseteq H \subseteq F$, and let Sp(I) be obtained from G(I) by splitting an artificial element $\alpha$ into s and d.

(i) Problem $(P_k|\ H - \alpha)$ has no solution if and only if there is no s-d dipath in Sp(I).

(ii) Let P be a minimum weight s-d dipath such that every s-d dipath defined on a subset of its nodes has a strictly larger weight, and let $\Sigma$ and $\Sigma'$ be the nodes of P in V and V', respectively. (s and d both give rise to $\alpha$ in $\Sigma$.) Then $I' = (I - \Sigma) \cup \Sigma'$ is optimal for $(P_k|\ H - \alpha)$.

We use this to solve $(P_k)$ as follows. Start with $H = F$ and the set $I = A$. Construct G(I) and then Sp(I) by splitting one node arising from an artificial element $\alpha$. Find a minimum weight dipath from the resulting source s to destination d, with the added condition that every s-d dipath on a proper subset of its nodes has a strictly larger weight. Then the set I' defined in Theorem 2(ii) is optimal for $(P_k|\ H - \alpha)$. Repeat this process with I' in place of I and $H - \alpha$ in place of H, until all artificial elements have been split. When this occurs, we have an optimal solution for $(P_k)$.

We close this section with a result that will be useful in the proof of Theorem 6.

**Theorem 3.** [BCG, Corollary]. Let $e \in V$ and $e' \in V'$.

(a) If P is an e-e' dipath in Sp(I) and $(I - \Sigma) \cup \Sigma'$ is a dependent set

in $M_1$, then there is an e-e' dipath $P_0$ in Sp(I) such that the intermediate nodes of $P_0$ constitute a proper subset of those of P, and $w(P_0) \leq w(P)$.

(b) If P is an e'-e dipath in Sp(I) and $(I - \Sigma) \cup \Sigma'$ is a dependent set in $M_2$, then there is an e'-e dipath $P_0$ in Sp(I) such that the intermediate nodes of $P_0$ constitute a proper subset of those of P, and $w(P_0) \leq w(P)$.

## 3. The Algorithm

In this section we show how to solve (P). We first assume that a feasible solution B is available.

For convenience, we index each element of E to identify the subset or **state** of the partition $E_1, E_2, ..., E_m$ to which it belongs. Thus, $e_i$ and $f_j$ are elements of $E_i$ and $E_j$, respectively. Also, the symbol ' is used to denote that an element is not in base B, so that $e_i \in B$ but $e_j' \in E - B$.

We define the **state digraph of B**, denoted S(B), as follows. The node set is $\{v_1, v_2, ..., v_m\}$, where node $v_i$ corresponds to state i, $i = 1, 2, ..., m$. For each $e_i \in B$ we construct an arc $a(e_i, e_j')$ with weight $w(a(e_i, e_j')) = w(e_j') - w(e_i)$ directed from node $v_i$ to node $v_j$ provided $(e_i, e_j')$ is a B-swap. We call these the **forward arcs** of S(B). Note that, in general, the digraph S(B) can have loops and parallel arcs. For each $e_j' \in E - B$, construct $a(e_j', e_l)$ from $v_j$ to $v_l$, with weight 0, provided $(B - \{e_l\}) \cup \{e_j'\}$ does not violate the cardinality conditions of $E_l$ and $E_j$. We call these the **backward arcs** of S(B). Note that the existence of a backward arc from $v_j$ to $v_l$ indicates that we can exchange <u>any</u> element $e_l \in B \cap E_l$ for <u>any</u> element $e_j' \in (E - B) \cap E_j$ without violating the cardinality conditions on $B \cap E_l$ and $B \cap E_j$. All such backward arcs $a(e_j', e_l)$ appear in parallel in S(B).

We could have constructed S(B) using G(B), the bipartite auxiliary digraph discussed in the previous section, as follows. Let $M_1 = M$ and let $M_2$ be the matroid whose bases are elements of the set

$$B^* = \{X \subseteq E:\ l_i \le |X \cap E_i| \le u_i,\ i = 1, 2, ..., m,\ |X| = r\}.$$

Construct G(B) as explained in Section 2. Consider, for $i = 1, 2, ..., m$, the set of nodes of G(B) corresponding to $E_i$ and identify this set as the node $v_i$. Thus, each arc $(e_i, e_j')$ or $(f_i', f_j)$ of G(B) becomes an arc joining $v_i$ to $v_j$, and this new digraph is clearly S(B).

**Lemma 1.** Let C be a dicycle of S(B). Either C is of the form

$$\{a(e_i, e_j'), a(e_j', e_k), ..., a(e_l, e_q'), a(e_q', e_i)\} \qquad (*)$$

or there exists a dicycle of S(B) of the form (*) with the same forward arcs as C (and, therefore, the same weight as C). Furthermore, there is a one-to-one correspondence between the dicycles of S(B) having the form (*) and the dicycles of G(B).

***Proof.*** *There are four ways in which C could fail to have the form (*): C could contain, as subsequences,*

(i) consecutive forward arcs $\{a(e_i, e_j'), a(f_j, e_k')\}$,

(ii) consecutive backward arcs $\{a(e_i', e_j), a(f_j', e_k)\}$,

(iii) a forward-backward pair with different elements from E - B, $\{a(e_i, e_j'), a(f_j', e_k)\}$, or

(iv) a backward-forward pair with different elements from B, $\{a(e_j', f_k), a(e_k, e_l')\}$.

In case (i) the loop $a(e_j', f_j)$ must be a backward arc in S(B) since swapping these two elements in B will not change $|B \cap E_j|$. So replace the consecutive forward arcs in C with $\{a(e_i, e_j'), a(e_j', f_j), a(f_j, e_k')\}$. (Recall that we only require dicycles to be arc-disjoint; so loops are permitted.)

Assuming i, j, and k to be distinct, the result of the consecutive backward arcs in case (ii) is that the cardinality of $E_i$ increases by one in $(B - \{e_j, e_k\}) \cup \{e_i', f_j'\}$, with the cardinality of $E_k$ decreasing by one. $E_j$ realizes no change in cardinality from this subsequence. Thus, $a(e_i', e_k)$ must be a backward arc in S(B), and we can replace the consecutive backward arcs in case (ii) with the single backward arc $a(e_i', e_k)$. Note that the same replacement is appropriate if any of the indices are not distinct.

In case (iii), the backward arc $a(f_j', e_k)$ can be replaced with the parallel arc $a(e_j', e_k)$. Similarly, $a(e_j', f_k)$ can be replaced with $a(e_j', e_k)$ in case (iv).

Since these steps (which result in a dicycle that has the form (*)) only affect the backward arcs in the dicycle, the weight of the dicycle remains unchanged.

The second statement in Lemma 1 is immediate. //

Define

$\beta(C) = \{e_i \in B : a(e_i, e_j')$ is a forward arc of $C\}$, and

$\beta'(C) = \{e_j' \in E - B : a(e_i, e_j')$ is a forward arc of $C\}$.

**Theorem 4.** Assume B is feasible for (P).

(i) B is optimal for (P) if and only if there are no negative weight dicycles in S(B).

(ii) Let C be a negative weight dicycle in S(B) with no negative weight dicycle D satisfying $\beta(D) \subset \beta(C)$ and $\beta'(D) \subset \beta'(C)$. Then $B' = (B - \beta(C)) \cup \beta'(C)$ is a base such that $w(B') < w(B)$.

**Proof.** This result follows directly from Lemma 1 and Theorem 1. //

Theorem 4 shows the equivalence of solving (P) and finding negative weight dicycles in S(B). This gives the foundation for a primal algorithm: start with a feasible B and use S(B) to find an improved solution. Continue in this manner until an S(B) is found with no negative weight dicycles.

We briefly discuss the complexity issues of this algorithm: Firstly, one must be able to find a negative dicycle in S(B) with the required property. Also, the number of iterations needed to reach the optimal solution can be relatively large; even though each iteration yields a better solution, the improvement may be slight. For these reasons the complexity is high, and we do not pursue our investigation of this primal algorithm. (We do note, however, that this approach may have merits in the context of sensitivity

analysis: Given an optimal solution for a certain set of weights, reoptimizing on a perturbed set of weights using this method may be more efficient in practice than starting from scratch, as other algorithms for matroid intersection would require.)

Instead we use these results to develop a dual algorithm. First, using a modification of the greedy algorithm, we can find an initial solution $B_0$. Start with $B_0 = \emptyset$ and, at each iteration, add an element $e \in E - B_0$ to $B_0$ if

(i) $B_0 \cup \{e\}$ is independent in M,

(ii) e has the smallest weight of all $f \in E - B_0$ satisfying (i), and

(iii) $|(B_0 \cup \{e\}) \cap E_i| \leq u_i$ for $i = 1, 2, ..., m$.

Stop when either no such e can be found, $|B_0| = r$, or, for the e chosen,

$$r - |B_0 \cup \{e\}| < \Sigma\{\max(l_i - |(B_0 \cup \{e\}) \cap E_i|, 0): i = 1, 2, ..., m\}$$

(in which case e should not be added to $B_0$). Violating this last condition would result in a $B_0$ which could not satisfy both $|B_0| = r$ and all lower bound conditions. At the termination of this procedure we clearly have the least weight independent set with $|B_0|$ elements.

Note that $B_0$ contains at least one element. If $|B_0| = r$, (P) is solved. Otherwise, consider a set A of $r - |B_0|$ artificial elements, where $E \cap A = \emptyset$. We assign to each artificial element a state as follows. Extend $E_1$ by unioning $\max\{0, l_1 - |B_0 \cap E_1|\}$ artificial elements. Then extend $E_2, E_3, ..., E_m$ similarly. Any remaining artificial elements can then be unioned to any of the states, as long as the resulting states satisfy $|(B_0 \cup A) \cap E_i| \leq u_i$. Note that, if this cannot be done, (P) is infeasible.

Let $F = E \cup A$ and define a matroid M(F) on the set F as follows: for $I \subseteq E$ and $J \subseteq A$, $I \cup J$ is independent in M(F) if and only if I is independent in M. The problem (P) relative to M(F) will be denoted (P| F); we now wish to find the least weight independent subset B of F such that $l_i \leq |B \cap E_i| \leq u_i$, for $i = 1, 2, ..., m$, and $|B| = r$.

To define (P| F) completely we need to assign weights to the artificial elements. We will give large negative weights to all artificial elements. Thus, the initial solution $B = B_0 \cup A$ will be optimal for (P| F).

Define the digraph $S^*(B)$ from $S(B)$ by creating two nodes, a source node s and a destination node d, from an artificial element, say $\alpha \in A$. We also allow s and d to denote the resulting elements. Assume $\alpha$ was assigned to $E_i$. Replace arcs of the form $a(\alpha, e_j')$ in $S(B)$ with arcs $a(s, e_j')$ in $S^*(B)$, and replace arcs of the form $a(e_i', \alpha)$ in $S(B)$ with $a(e_i', d)$ in $S^*(B)$.

Note that we could have constructed $S^*(B)$ by forming $Sp(B)$ from $G(B)$, and then shrinking groups of nodes (excluding s and d) corresponding to each of the states into the nodes $v_1, v_2, \ldots, v_m$. Thus the following theorem is a direct result of Theorem 2. For any s-d dipath P in $S^*(B)$, define

$$\beta(P) = \{e_i \in B : a(e_i, e_j') \text{ is a forward arc of } P\}, \text{ and}$$
$$\beta'(P) = \{e_j' \in E - B : a(e_i, e_j') \text{ is a forward arc of } P\}.$$

**Theorem 5.** Let B be optimal for $(P|\ H)$, where $E \subseteq H \subseteq F$, and let $S^*(B)$ be obtained from $S(B)$ by creating s and d from artificial element $\alpha$.

(i) Problem $(P|\ H - \alpha)$ has no solution if and only if there is no s-d dipath in $S^*(B)$.

(ii) Let P be a minimum weight s-d dipath such that every s-d dipath Q satisfying $\beta(Q) \subset \beta(P)$ and $\beta'(Q) \subset \beta'(P)$ has a strictly larger weight. Then $B' = (B - \beta(P)) \cup \beta'(P)$ is optimal for $(P|\ H - \alpha)$.

Theorem 5 provides the foundation for our algorithm. Start with $H = F$ and $B = B_0 \cup A$, which is optimal for $(P|\ F)$. Construct $S(B)$ and then $S^*(B)$ by creating s and d from artificial element $\alpha$. Find a minimum weight s-d dipath P such that every s-d dipath Q satisfying $\beta(Q) \subset \beta(P)$ and $\beta'(Q) \subset \beta'(P)$ has a strictly larger weight. Then the set B' defined in Theorem 5(ii) is optimal for $(P|\ H - \alpha)$. Repeat this process with B' in place of B and $H - \alpha$ in place of H, until the resulting B' contains no artificial elements. At this step, B' is optimal for (P).

The complexity of the algorithm depends on our ability to find a minimum weight s-d dipath in $S^*(B)$ with the added restriction stated in Theorem 5(ii). We concentrate on this issue in the next section.

## 4. Complexity of the Algorithm.

Note that the arc weights of $S(B_0 \cup A)$ are all nonnegative. To see this, first consider arcs of the form $a(\alpha, e_j')$, where $\alpha$ is artificial. Since $w(\alpha)$ has been chosen small enough, $w(a(\alpha, e_j')) = w(e_j') - w(\alpha) > 0$. Now consider an arc $a(e_i, e_j')$, where $e_i \in B_0$. Since this arc is in $S(B_0 \cup A)$, we have $(B_0 - \{e_i\}) \cup \{e_j'\}$ independent in M. But then $w(e_i) \leq w(e_j')$ by requirements (i) and (ii) of our modified greedy approach in the construction of $B_0$; so $w(a(e_i, e_j')) \geq 0$. Finally, when e is not in $B_0 \cup A$, $w(a(e, f)) = 0$.

All arc weights nonnegative in $S(B_0 \cup A)$ implies the same for $S^*(B_0 \cup A)$; so no minimum weight s-d dipath will repeat nodes. Thus, for all pairs of nodes, we only need to consider the arc of minimum weight joining $v_i$ to $v_j$ when searching for the minimum weight s-d dipath, and we can apply Dijkstra's algorithm to find such an s-d dipath P. Furthermore, the requirement that every s-d dipath Q satisfying $\beta(Q) \subset \beta(P)$ and $\beta'(Q) \subset \beta'(P)$ has a strictly larger weight can be obtained by adding a small positive $\varepsilon$ to all the arc weights of $S^*(B_0 \cup A)$.

We now show that nonnegative weights on the arcs of S(B) can be preserved throughout the algorithm. Define a variable $D(v_i)$ associated with every node of the digraph $S^*(B)$ and a reduced weight

$$w'(a(e_i, e_j')) = w(a(e_i, e_j')) + D(v_i) - D(v_j)$$

associated with each forward arc of S(B). Similarly, for each backward arc define

$$w'(a(e_k', e_l)) = D(v_k) - D(v_l).$$

Note that, in terms of the reduced weights, the length of an s-d dipath P is $w'(P) = w(P) + D(s) - D(d)$ since, for any intermediate node $v_i$, the variable $D(v_i)$ cancels out on the two arcs of P that contain $v_i$. Since D(s) - D(d) is a constant that does not depend on P, it is equivalent to solve the minimum weight s-d dipath problem in $S^*(B)$ with the reduced weights w' instead of the original weights w. We provide a choice for the variables $D(v_i)$ that guarantees nonnegative reduced weights from one iteration to another in the next theorem.

**Theorem 6.** Let B be optimal for (P| H) where $E \subseteq H \subseteq F$, and assume that an s-d dipath exists in $S^*(B)$. Set $D(v_i)$ to be the weight of a minimum weight s-$v_i$ dipath in $S^*(B)$, for $i = 1, 2, ..., m$. If there is no s-$v_i$ dipath, set $D(v_i) = M$, where M is some large constant. Let B' be as defined in Theorem 5(ii). Then, the reduced weights $w'(a(e_k', e_l)) = D(v_k) - D(v_l)$ and $w'(a(e_i, e_j')) = w(a(e_i, e_j')) + D(v_i) - D(v_j)$ are nonnegative for every backward arc $a(e_k', e_l)$ and every forward arc $a(e_i, e_j')$ of the digraph S(B') for problem (P| H - $\alpha$).

Using the same argument as in Lemma 1, we can show that, for every s-d dipath P in $S^*(B)$ which does not have the form

$$\{a(s, e_i'), a(e_i', e_j), a(e_j, e_k'), ..., a(e_l', d)\}, \qquad (**)$$

there is an s-d dipath of the form (**) with the same forward arcs and, hence, the same weight as P.

For the proof of Theorem 6 we need the following proposition, which is a direct result of Theorem 3. We also need to extend the definitions of $\beta(P)$ and $\beta'(P)$ to u-v dipaths P, where u and v are general nodes of $S^*(B)$:

$$\beta(P) = \{e_i \in B : a(e_i, e_j') \text{ is a forward arc of } P, \text{ or } a(e_k', e_i) \text{ is the last arc of } P\}, \text{ and}$$

$$\beta'(P) = \{e_j' \in E - B : a(e_i, e_j') \text{ is a forward arc of } P, \text{ or } a(e_j', e_k) \text{ is the first arc of } P\}.$$

**Proposition 2.** Let P be a minimum weight s-d dipath in $S^*(B)$ such that every s-d dipath Q satisfying $\beta(Q) \subset \beta(P)$ and $\beta'(Q) \subset \beta'(P)$ has a strictly larger weight than P, and let $P_0$ be some subpath of P.

(a) If the first and last arcs on $P_0$ are forward arcs, then $[B - \beta(P_0)] \cup \beta'(P_0)$ is independent in M.

(b) If the first and last arcs on $P_0$ are backward arcs, then $[B - \beta(P_0)] \cup \beta'(P_0)$ satisfies the cardinality conditions on each $E_i$.

**Proof of Theorem 6.** First we show that the choice for $D(v_i)$ gives nonnegative reduced weights on the arcs of S(B). Consider $a(e_i, e_j')$ in S(B),

and suppose $w(a(e_i, e_j')) + D(v_i) - D(v_j) < 0$. This gives

$$D(v_i) + w(a(e_i, e_j')) < D(v_j),$$

which implies that the minimum weight $s$-$v_i$ dipath together with the arc $a(e_i, e_j')$ gives a smaller weight from $s$ to $v_j$ than $D(v_j)$, a contradiction. Similarly, for each backward arc $a(f_i', f_j)$ in $S(B)$, $w(a(f_i', f_j)) + D(v_i) - D(v_j) \geq 0$. Thus, our choice of $D(v_i)$ gives nonnegative reduced weights on the arcs of $S(B)$.

Let $P$ be the minimum weight $s$-$d$ dipath that yields $B'$ in Theorem 5(ii), and let $P_i$ be the subpath of $P$ from $s$ to some node $v_i$. Let $\beta_1(i)$ and $\beta_1'(i)$ be the sets of elements of $B$ and $E - B$, respectively, represented by the forward arcs of $P_i$, and let $\beta_2(i)$ and $\beta_2'(i)$ be the corresponding sets of elements represented by the backward arcs of $P_i$. By Proposition 2, $B_1(i) = B - \beta_1(i) + \beta_1'(i)$ is independent in $M$, and $B_2(i) = B - \beta_2(i) + \beta_2'(i)$ satisfies the cardinality conditions on each $E_i$. For ease of notation we allow $v_i$ to represent $s$ and $d$, as well as the nodes *representing states* $E_1, E_2, \ldots, E_m$.

We construct a digraph $N_i$ as follows. The node set of $N_i$ is that of $S(B)$. If $B_1(i) - e_j + e_k'$ is independent in $M$, then $a(e_j, e_k')$ is a forward arc of $N_i$; if $B_2(i) - e_j + e_k'$ satisfies the cardinality conditions of each $E_i$, then $a(e_k', e_j)$ is a backward arc of $N_i$. Note that, in the latter case, we call $(e_k', e_j)$ a $B_2(i)$-swap with respect to the partition matroid.

We will show that this choice of $D(v_i)$ gives nonnegative reduced weights on the arcs of $N_i$ by induction on the nodes of $P$, starting from $v_i = s$. Note that when we reach $v_i = d$, we will have $N_i = S(B')$ for $(P| H - \alpha)$, and the theorem will be proved.

When $v_i = s$, $N_i = S(B)$, and the result has been proved above. Let $v_i$ be a node of $P$ for which the result holds, and let $v_j$ be the node following $v_i$ in $P$. There are two cases to consider, as the arc joining $v_i$ to $v_j$ can be a forward arc or a backward arc.

Suppose the arc joining $v_i$ to $v_j$ in $P$ is a forward arc $a(e_i, e_j')$. In this case the backward arcs of $N_j$ are the same as those of $N_i$; hence, they

have nonnegative reduced weights by induction. Thus, we only need to show $D(v_q) - D(v_p) \le w(e_q') - w(e_p)$ for every $B_1(j)$-swap $(e_p, e_q')$. Note that if $(e_p, e_q')$ is also a $B_1(i)$-swap we are done. So only consider $B_1(j)$-swaps which are not $B_1(i)$-swaps.

If $(e_p, e_q') = (e_j', e_i)$, then

$$D(v_j) = D(v_i) + w(e_j') - w(e_i) \qquad (\dagger)$$

since $a(e_i, e_j')$ is on the minimum weight s-d dipath P, and the desired inequality is immediate.

For the remaining cases, consider $B^* = B_1(i) - e_i + e_j' - e_p + e_q'$, which is independent in M. Suppose $e_p = e_j'$ and $e_q' \neq e_i$. Then $B^* = B_1(i) - e_i + e_q'$ and $(e_i, e_q')$ is a $B_1(i)$-swap. Therefore, $w(e_q') - w(e_i) + D(v_i) - D(v_q) \ge 0$ by induction. Combining this inequality with (†) and $e_p = e_j'$, we get $w(e_q') - w(e_p) + D(v_p) - D(v_q) \ge 0$, and the result follows. A similar argument applies when $e_p \neq e_j'$ and $e_q' = e_i$.

Finally, suppose $e_p \neq e_j'$ and $e_q' \neq e_i$. Since $(e_p, e_q')$ is not a $B_1(i)$-swap, yet $B^*$ is a base, it follows from Proposition 1 that both $(e_p, e_j')$ and $(e_i, e_q')$ are $B_1(i)$-swaps. Thus, $w(e_j') - w(e_p) + D(v_p) - D(v_j) \ge 0$, and $w(e_q') - w(e_i) + D(e_i) - D(e_q) \ge 0$. Summing these inequalities and applying (†), we get $w(e_q') - w(e_p) + D(v_p) - D(v_q) \ge 0$.

Now suppose the arc joining $v_i$ to $v_j$ in P is a backward arc $a(f_i', f_j)$. Then $B_1(j) = B_1(i)$, and the forward arcs of $N_j$ are the same as those of $N_i$. Hence, they have nonnegative reduced weights by induction. Thus, we only need to show $D(v_p) - D(v_q) \ge 0$ for every $B_2(j)$-swap $(e_p', e_q)$. Note that if $(e_p', e_q)$ is also a $B_2(i)$-swap we are done by induction. So only consider $B_2(j)$-swaps which are not $B_2(i)$-swaps.

If $(e_p', e_q) = (f_j, f_i')$, then

$$D(v_j) = D(v_i) \qquad (\dagger\dagger)$$

since $a(f_j', f_i)$ is on the minimum weight s-d dipath P, and the desired inequality is immediate.

For the remaining cases consider $B^{**} = B_2(i) - f_j + f_i' - e_q + e_p'$, which satisfies the cardinality conditions for each $E_i$ since $(e_p', e_q)$ is a $B_2(j)$-swap. Suppose $e_p' = f_j$ and $e_q \neq f_i'$. Then $B^{**} = B_2(i) - e_q + f_i'$, and

$(f_i', e_q)$ is a $B_2(i)$-swap. Hence, $D(v_i) - D(v_q) \geq 0$ by induction. This and equation (††) show that $D(v_p) - D(v_q)$. The proof is similar when $e_p' \neq f_j$ and $e_q = f_i'$.

Finally, suppose $e_p' \neq f_j$ and $e_q \neq f_i'$. Since $(e_p', e_q)$ is not a $B_2(i)$-swap, yet $B^{**}$ is independent in the partition matroid, it follows from Proposition 1 that both $(e_p', f_j)$ and $(f_i', e_q)$ are $B_2(i)$-swaps. Thus, induction yields $D(v_p) - D(v_j) \geq 0$ and $D(v_i) - D(v_q) \geq 0$. Summing these inequalities and applying (††), we get $D(v_p) - D(v_q) \geq 0$. //

We now summarize the complexity of the algorithm. $O(r)$ iterations will be needed, one iteration for the removal of each artificial element in the initial solution. In each iteration we need to construct $S(B)$, find a minimum weight s-d dipath, and update the variables $D(v_i)$. Recall that $|E| = n$, and $c$ represents the time required to find the circuit of $I \cup \{e'\}$ in $M$ (or show that none exists), where $I \subseteq E$ is independent and $e' \in E - I$.

To construct $S(B)$ for the problem $(P|H)$, $E \subseteq H \subseteq F$, we solve the following circuit recognition problem for each $e' \in H - B$: Check whether $B \cup \{e'\}$ is dependent in $M(H)$, and, if it is, find the unique circuit of $B \cup \{e'\}$. This task requires time $c$ for each $e' \in H - B$. So the complexity of constructing $S(B)$ is $O(nc)$. In some instances it is possible to speed up the construction of $S(B)$, as not all parallel arcs need to be constructed, but only the shortest. An example of this will be given in Section 7.

Using the fast implementation of Fredman and Tarjan (see [FT]), the complexity of finding a minimum weight s-d dipath by Dijkstra's algorithm is at most $O(\rho + m \log m)$, where $\rho$ is the cardinality of the set of arcs in $S(B)$.

Note that the variables $D(v_i)$ needed in Theorem 6 are actually computed in the course of finding a shortest s-d dipath. Thus, no extra computations are needed. Also, $nc$ is larger than $\rho$. Therefore, the overall complexity of the algorithm is $O(r(nc + m \log m))$.

## 5. Improved Starting Solution and Variants of the Algorithm.

In this section we provide two variants of our algorithm. Although they do not improve our complexity bounds, they may yield a more successful implementation. Let B' be a minimum weight base of M, found using the greedy algorithm. Both variants use B' to obtain a full starting base B instead of the initial solution $B_0$ to which artificials are appended.

In the first variant we make copies of selected elements of B' which replace their counterparts, thereby producing a base B that is feasible for (P) relative to an extended matroid. The copies, labeled artificial elements, are assigned to the sets $E_1, E_2, ..., E_m$ by the following rules so that the cardinality restrictions are satisfied.

Initialize with $B = B'$. For any pair $\{i, j\}$ such that $l_i > |B \cap E_i|$ and $u_j < |B \cap E_j|$, define $\alpha_i$ to be a copy of $e_j$, the maximum weight element in $B \cap E_j$. Assign the artificial element $\alpha_i$ to $E_i$ with $w(\alpha_i) = w(e_j)$, and replace B with $(B - \{e_j\}) \cup \{\alpha_i\}$. Extend M so that $I \cup \{\alpha_i\}$ is independent in the extended matroid if and only if $I \cup \{e_j\}$ is independent in M.

Once no more pairs $\{i, j\}$ of this form exist, select a pair such that $l_i > |B \cap E_i|$ and $l_j < |B \cap E_j|$ or such that $u_i > |B \cap E_i|$ and $u_j < |B \cap E_j|$. (If none remain, all cardinality constraints are satisfied, or (P) has no feasible solution.) Then create the artificial element $\alpha_i$ and change B and M as above.

One advantage of this variant is that, in the process of eliminating an artificial element by solving a minimum weight s-d dipath problem, it is possible that one or more other artificials may leave B, in which case they can be dropped from the problem and disregarded thereafter. Thus, fewer state digraphs may be needed to eliminate the artificial elements. A simple version of this variant, one in which all artificial elements are assigned the same weight, is used in Section 7 for the special case in which M is a graphic matroid.

The second variant replaces the bounds for the cardinality constraints by relaxed bounds that are progressively tightened until the original bounds are achieved. Begin with the base $B = B'$, and, for each $B$ subsequently generated, define $u_i' = \max\{u_i, |B \cap E_i|\}$ and $l_i' = \min\{l_i, |B \cap E_i|\}$. Once $u_i' = u_i$ and $l_i' = l_i$ for $i = 1, 2, \ldots, m$, problem (P) is solved. Otherwise, identify a pair $\{i, j\}$ by the rules of the preceeding variant and create a single artificial $\alpha_i$ to replace an element $e_j \in B$, thus increasing $l_i'$ by 1 or decreasing $u_j'$ by 1, or both. Then eliminate $\alpha_i$ by solving a minimum weight s-d dipath problem and repeat. A noteworthy feature of this variant is that $B$ never contains an artificial element except as a device for creating an appropriately defined minimum weight dipath problem.

Our results lead to another postoptimization procedure, in addition to that given by Theorem 4, for finding a new optimal solution when element weights are perturbed. The basic steps of this postoptimizing method are as follows. Of the elements with new weights, first consider only those in the set $R$ defined to consist of: (1) elements in $B$ whose weights are decreased, and (2) elements not in $B$ whose weights are increased. The problem in which only the elements of $R$ receive their new weights is still solved optimally by the current solution. Correspondingly, update the distances $D(v_i)$ for the minimum weight dipath tree created at the last iteration of solving (P).

Now consider one at a time each remaining element whose weight is to be changed. If the element is in $B$, solve the minimum weight dipath problem that treats this element precisely as if it were an artificial element with the new given weight. If the element is not in $B$, determine the effect of forcing it to become part of $B$ by solving an analogously defined minimum weight dipath problem: Let $e_j'$ denote the element under consideration. Replace arcs of the form $a(e_j', e_i)$ in $S(B)$ with arcs $a(s, e_i)$ in $S^*(B)$, and replace arcs of the form $a(e_i, e_j')$ in $S(B)$ with arcs $a(e_i, d)$ in $S^*(B)$.

Our previous results hold for both cases, and, hence reduced arc weights obtained after the minimum weight dipath problem has been solved, and B is changed, will remain nonnegative. Thus, upon giving the element its new weight, if the resulting solution has a smaller weight than B, the new solution is kept. However, if the there is no improvement, or if no s-d dipath exists, B is unchanged.

It is, of course, unnecessary to solve the minimum weight dipath problem if, on tentatively assigning an element its new weight, the reduced weights for all arcs associated with the element are nonpositive when the element is in B and nonnegative when the element is not in B. In addition, when the current base B is replaced with a new one, some elements not yet assigned their new weights may enter or leave B. These may automatically be given their new weights, since they correspond to elements of R for the current iteration. As this occurs, minimum weight dipath distances are updated accordingly in the minimum weight dipath tree.

This process solves at most one minimum weight dipath problem for each element whose weight changes (excluding those in the original and subsequent sets R). Hence, the work is at most $O(nc + m \log m)$ for each perturbed weight; so we have an efficient postoptimizing procedure when the number of such elements is small.

## 6. Weighted bipartite matching.

Consider a graph $G = (V, E)$ and a weight function $w: E \rightarrow R$. The matching problem consists of finding a minimum weight subset $F \subseteq E$ such that each node of V is incident with exactly one member of F. In this section we assume that G is a bipartite graph where the partition $V_1 \cup V_2 = V$ is such that $|V_1| = |V_2|$. The weighted bipartite matching problem, also called the assignment problem, is a special instance of problem (P) where the partition of E is induced by the nodes of $V_2$.

Specifically, $e \in E_i$ if and only if it is incident with node $v_i \in V_2$. For each i, $l_i = u_i = 1$. The matroid M is also a partition matroid, namely the one induced by the node set $V_1$.

For this problem $r = |V|/2$, $n = |E|$, and $m = |V|/2$. To obtain the complexity of our algorithm for this problem, we have to determine the complexity of constructing S(B). Note that the circuits of M have length 2; so the number of forward arcs of S(B) is at most $|E|$. Similarly, there are at most $|E|$ backward arcs.

Thus, the construction of S(B) requires time $O(|E|)$. Since the Fredman Tarjan algorithm finds the minimum weight dipath in time $O(|E| + |V| \log |V|)$, the overall complexity is $O(|V|\,|E| + |V|^2 \log |V|)$.

Note that our algorithm is closely related to the usual augmenting path algorithm. Although this algorithm and the associated bound are not new, it is interesting that they are obtained using the state digraph approach. Recall that the general matroid intersection algorithm only gives the bound $O(|V|^2\,|E|)$.

## 7. Minimum spanning trees.

Let $G = (V, E)$ be a connected graph, with $E_1, E_2, \ldots, E_m$ a general partition of the edge set, and let each edge of G be assigned a weight. For $i = 1, 2, \ldots, m$ assign two integers $l_i$ and $u_i$, where $l_i \le u_i$. In this section, we consider the problem of finding a minimum weight spanning tree $B \subseteq E$ with the restriction that $l_i \le |B \cap E_i| \le u_i$, for $i = 1, 2, \ldots, m$. The minimum spanning tree problem with restrictions on the number of edges in each $E_i$ is an instance of (P): M is the graphic matroid defined on the edge set E, and the generalized partition matroid is induced by the partition of E. We will refer to $E_1, E_2, \ldots, E_m$ as the m *color classes*.

An interesting feature of this instance is that, by using the special structure of the tree, we do not need to construct the entire state graph S(B). However, our method requires that we work with spanning trees. So, instead of simply extending the graphic matroid M to M(F) as described in Section 3, create an artificial edge in G for each artificial element so that the resulting first solution, $B_0 \cup A$, is a spanning tree in the extended graph $G^* = (V, E \cup A)$. This is an application of our first variant in Section 5.

We now outline a procedure for constructing the state graph S(B). For each color i, label each $e \in B$ with the edge $f \in E - B$ of color i that gives the smallest weight B-swap (e, f). Thus, we will find the minimum weight arc in S(B) from $v_i$ to $v_j$ for each pair $(v_i, v_j)$ of states.

Begin by sorting the edges in each color class in ascending order by weight. At worst, this work is $O(|E| \log |V|)$. Note that we need to sort the edge weights only once since the reduced weights only add a constant to all the edges in the same color class. At each iteration of our algorithm, in the construction of each new state graph, proceed as follows:

1. Set up (or update) the current tree B with an arbitrary root and predecessor indexing.

2. For $i = 1, 2, \ldots, m$, start with all edges in B unlabeled. Examine the edges $f = (x, y)$ of color i in $E - B$ in ascending weight order. Find the first common ancestor of x and y in B, and label each unlabeled edge of B in the cycle of $B \cup \{f\}$ with f. Stop the labeling procedure for color i when either all tree edges have been labeled, or all edges of color i in $E - B$ have been examined.

Since we examine the edges of color i in $E - B$ in ascending weight order, each edge of B will be labeled with the edge of $E - B$ which results in the least weight B-swap. Using the set merging technique of Gabow and Tarjan (see [GT2]), $O(|E_i| + |V|)$ work is required for this labeling procedure

for each color. So the complexity of the construction of each state graph is $O(|E| + m|V|)$. The complexity due to the Fredman-Tarjan minimum weight dipath algorithm is at most $O(m^2)$, and the graphic matroid has rank at most $|V| - 1$. Hence, the overall complexity is $O(|V| m^2 + |V| |E| + m |V|^2)$. Note that the $O(|E| \log |V|)$ work required for sorting the elements in each color class is dominated by this complexity.

We note that, for small values of m, much faster algorithms are known. For $m = 1$, the problem is nothing but the classical minimum spanning tree problem, and there is an algorithm of complexity $O(|E| \log \log |V|)$ due to Yao ([Y]). More recent developments can be found in [GGST]. For $m = 2$, the bound $O(\sigma + |V| \log |V|)$ has been achieved by Gabow and Tarjan ([GT1]), where $\sigma$ denotes the complexity of finding a minimum spanning tree.

Let $V^* = \{v_1, v_2, \ldots, v_{m-1}\}$ be a stable set of G; that is, no edge of E has both endpoints in $V^*$. For $i = 1, 2, \ldots, m-1$, let $E_i$ denote the set of edges which are incident with node $v_i$ in G, and let $E_m$ be the set of remaining edges. In this case the $O(m^2)$ due to the Fredman-Tarjan minimum weight dipath algorithm is dominated by the complexity of constructing the state graph. Hence, the overall complexity is $O(|V| |E| + m |V|^2)$.

In the special stable set case where $l_i = u_i = 2$ for $i = 1, 2, \ldots, m-1$, and $l_m = 0$ and $u_m = |V| - 2m + 1$, the greedy phase of finding a starting solution $B_0$ can substantially reduce the number of iterations needed. For $i = 1, 2, \ldots, m-1$, add a constant $c_i$ to the weights of the edges incident with node $v_i \in V^*$. We assign large enough values to the constants so that the greedy phase will first choose $|V| - 2m + 1$ edges in $E_m$ to construct the spanning tree. Let w be the weight of the last edge added. For $i = 1, 2, \ldots, m-1$, we adjust the constant $c_i$ so that the minimum weight edge incident with $v_i$ has weight w and, therefore, can be chosen next in the greedy phase.

This extended greedy solution $B_0$ contains $|V| - m$ edges, so only $m - 1$ artificial elements are needed. They can be chosen so that $B_0 \cup A$ is a tree having degree 2 at each of the nodes of $V^*$. Furthermore, if the weights of the artificial elements are chosen small enough, $B_0 \cup A$ is an optimum solution of $(P|\ E \cup A)$ and, therefore, a valid start for our dual algorithm. Hence, only $m - 1$ iterations of the algorithm are needed. This yields an algorithm of complexity $O(m(|E| + m|V|) + |E| \log |V|)$. (Note that the sorting-time complexity is not dominated in this case.)

In this case (P) is the problem of finding the minimum weight spanning tree in G such that every node in $V^*$ is adjacent to exactly two edges of the tree. Such a spanning tree is a relaxation of a hamiltonian path, namely, a simple path which contains each node of G.

Consider the following variant of spanning trees. A **1-tree** is a graph with nodes $\{1, 2, ..., |V|\}$ consisting of a spanning tree on $\{2, 3, ..., |V|\}$ together with two edges incident with node 1. The traveling salesman problem on $G = (V, E)$ seeks a minimum weight tour, a cycle which passes through each node exactly once. Observing that a tour is precisely a 1-tree in which each node has degree 2, Held and Karp explored approaches to the traveling salesman problem which involve 1-tree relaxations (see [HK1] and [HK2]).

Our technique permits a generalization of Held and Karp's method to yield a constrained 1-tree as follows. Suppose the nodes are indexed so that $\{1, 2, ..., m\text{-}1\}$ is a stable set of G, and let $V^* = \{2, 3, ..., m\text{-}1\}$. Then, instead of using a general spanning tree on $\{2, 3, ..., |V|\}$, find a minimum weight spanning tree such that every node in $V^*$ has degree 2. Add the two edges incident with node 1 of minimum weight. Clearly, this constrained 1-tree is still a relaxation of a minimum weight tour. It is tighter than the 1-tree relaxation, which is obtained when $V^* = \varnothing$. Recently, however, [LR] have shown that the lagrangian duals based on these two relaxations have the same value. Next, we consider a further strengthening of the 1-tree relaxation.

Define the **star** of a node to be the set of edges incident with that node. Suppose the nodes are ordered from 1 to |V|. For node 1 define its **net star** to be its star. Then, for $i = 2, 3, ..., |V|$, define the **net star** of node $i$ to be its star, excluding edges which appear in the stars of nodes $1, 2, ..., i-1$.

Consider the following constrained spanning tree problem. For each node $i$, let $r_i$ be the number of edges deleted from its star to create its net star. Then we can stipulate that the spanning tree on $\{2, 3, ..., |V|\}$ must contain at least $\max\{0, 2 - r_i\}$ and at most 2 edges from the net star of each node $i \geq 2$. When $\{1, 2, ..., m-1\}$ is a stable set of G, as we assumed earlier, this procedure introduces additional constraints to the requirement that every node in $V^*$ has degree 2. This is a valid relaxation of the traveling salesman problem even if $\{1, 2, ..., m-1\}$ is not a stable set. The cardinality restriction on the net stars can be imposed on all the nodes $\{2, 3, ..., |V|\}$ or simply on a subset, say $\{2, 3, ..., m-1\}$. The complexity of the algorithm in this case is still $O(|V|\,|E| + m\,|V|^2)$.

There may be merit in choosing net stars adaptively, in succession. Consider first solving the spanning tree problem, selecting as node 2 one whose net star constraint is violated, and then solving (P) as if $E_2$ and its complement are the only two sets to consider. Then select as node 3 one whose net star constraint is now violated (the net stars changing as new nodes are selected), and now solve (P) as defined by $E_2$, $E_3$, and all remaining edges. Repeat this process until no net star constraint is violated. Note that the net star selected at each step may be determined according to some measure of constraint violation.

At each iteration the state digraph is obtained by splitting a node, say $v_k$, into two new nodes. Using $D(v_k)$ for each new node guarantees nonnegative reduced costs, so that the Fredman Tarjan algorithm can be used to solve the resulting minimum weight dipath problems. These observations also provide an alternative solution method for the general matroid problem in which the sets $E_i$ are fixed beforehand. Using the above technique, these

sets can be incorporated successively into the solution in any chosen sequence. We note that the complexity bounds for this procedure are the same as the bounds for the procedures that consider all sets simultaneously.

**Acknowledgement.**

We thank Hal Gabow for pointing out the labeling procedure of Section 7, which, as a result, reduced the complexity bounds of a preliminary version of this paper.

**References.**

[BCG] C. Brezovec, G. Cornuéjols, and F. Glover, "Two Algorithms for Weighted Matroid Intersection," *Mathematical Programming* **36** (1986), 39-53.

[E1] J. Edmonds, "Submodular Functions, Matroids and Certain Polyhedra," in: R. Guy, ed., *Combinatorial Structures and their Applications, Proceedings of the Calgary International Conference*, (Gordon and Breach, New York, 1970), 69-87.

[E2] J. Edmonds, "Matroid Intersection," *Annals of Discrete Mathematics* **4** (1979), 39-49.

[FT] M.L. Fredman and R.E. Tarjan, "Fibonacci Heaps and their Uses in Improved Network Optimization Algorithms," *Proceedings of the 25th Annual IEEE Symposium on the Foundations of Computer Science* (1984), 338-346.

[GGST] H.N. Gabow, Z. Galil, T. Spencer and R.E. Tarjan, "Efficient Algorithms for Finding Minimum Spanning Trees in Undirected and Directed Graphs," *Combinatorica* **6** (1986), 109-122.

[GT1] H.N. Gabow and R.E. Tarjan, "Efficient Algorithms for a Family of Matroid Intersection Problems," *Journal of Algorithms* **5** (1984), 80-131.

[GT2] H.N. Gabow and R.E. Tarjan, "A Linear-time Algorithm for a Special Case of Disjoint Set Union," *Journal of Computer and System Sciences* **30** (1985), 209-221.

[HK1] M. Held and R. Karp, "The Traveling-Salesman Problem and Minimum Spanning Trees," *Operations Research* **18** (1970), 1138-1162.

[HK2] M. Held and R. Karp, "The Traveling-Salesman Problem and Minimum Spanning Trees: Part II," *Mathematical Programming* **1** (1971), 6-25.

[L] E. Lawler, *Combinatorial Optimization: Networks and Matroids*, Holt, Rinehart and Winston, New York (1976).

[LR] M. Leclerc and F. Rendl, "Constrained Spanning Trees and the Traveling Salesman Problem," Research Report, Department of Combinatorics and Optimization, University of Waterloo (1987).

[W] D. J. A. Welsh, *Matroid Theory*, Academic Press, London (1976).

[Y] A. Yao, "An O(|E| log log |V|) Algorithm for Finding Minimum Spanning Trees," *Information Processing Letters* **4** (1975), 21-23.

END

DATE

FILMED

6-1988

DTIC