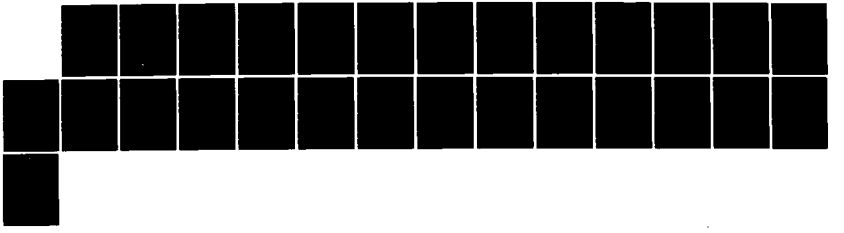
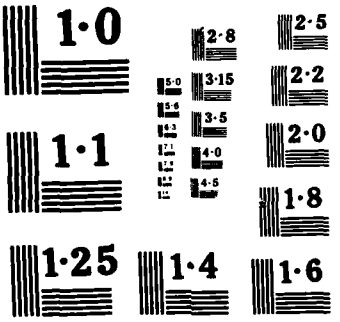
AD-A156 525 TWO ALGORITHMS FOR WEIGHTED MATROID INTERSECTION(U) 1/1
CARNEGIE-MELLON UNIV PITTSBURGH PA MANAGEMENT SCIENCES
RESEARCH GROUP C BREZOVEC ET AL. MAR 85 MSRR-517
UNCLASSIFIED N00014-82-K-0329 F/G 12/1 NL





TWO ALGORITHMS FOR WEIGHTED MATROID INTERSECTION

by

Carl Brezovec,
Gérard Cornuéjols,
Carnegie-Mellon University

and

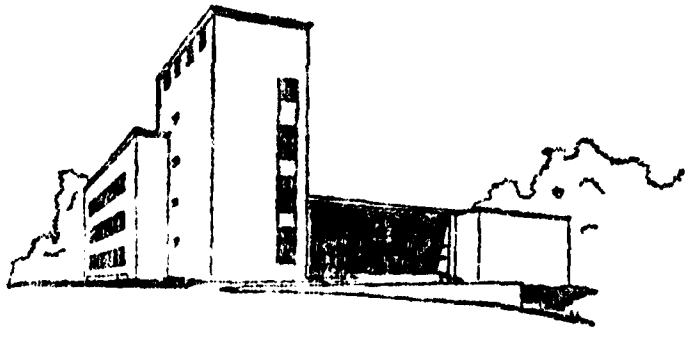
Fred Glover
University of Colorado, Boulder

Carnegie-Mellon University

PITTSBURGH, PENNSYLVANIA 15213

GRADUATE SCHOOL OF INDUSTRIAL ADMINISTRATION

WILLIAM LARIMER MELLON, FOUNDER



W.P. #29-84-85

Management Science Research Report No. 517

TWO ALGORITHMS FOR WEIGHTED MATROID INTERSECTION

by

Carl Brezovec,
Gérard Cornuéjols,
Carnegie-Mellon University

and

Fred Glover
University of Colorado, Boulder

March 1985

This report was prepared as part of the activities of the Management Sciences Research Group, Carnegie-Mellon University under Contract No. N00014-82-K-0329 with the U. S. Office of Naval Research and in part by NSF Grant ECS-8205425. 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

Abstract:

Consider a finite set E, a weight function $w : E \to \mathbb{R}$, and two matroids M_1 and M_2 defined on E. The weighted matroid intersection problem consists of finding a set $I \subseteq E$, independent in both matroids, that maximizes $\Sigma(w(e): e \text{ in } I)$. We present an algorithm of complexity $O(nr(r + c + \log n))$ for this problem, where $n = |E|$, $r = \min(\text{rank}(M_1), \text{rank}(M_2))$, $c = \max(c_1, c_2)$ and, for $i = 1, 2$, c_i is the complexity of finding the circuit of $I \cup \{e\}$ in M_i (or show that none exists) where e is in E and $I \subseteq E$ is independent in M_1 and M_2. This improves the complexity of earlier algorithms by a factor r, when $c \leq O(n)$, and by a factor n when $\max(c, \log n) \leq O(r)$. A related problem is to find a maximum weight set, independent in both matroids, and of given cardinality k (if one exists). Our algorithm also solves this problem. In addition, we present a second algorithm which, given a feasible solution of cardinality k, finds an optimal one of the same cardinality. A sensitivity analysis on the weights is easy to perform using this approach.

1. Introduction:

Let E be a finite set, M_1 and M_2 two matroids defined on E and $w : E \to \mathbb{R}$ a weight function. The weight of a subset $F \subseteq E$ is defined by $w(F) = \Sigma(w(e): e \text{ is in } F)$. The weighted matroid intersection problem consists of finding a maximum weight set $I \subseteq E$ which is independent in both matroids.

Let k be a positive integer. Call $F \subseteq E$ a **k-set** if $|F| = k$. Define F to be **feasible** if it is independent in both M_1 and M_2. We also consider the

problem of finding a maximum weight feasible k-set $I \subseteq E$, if one exists. This problem is denoted by (P_k).

Problem (P_k) and the weighted matroid intersection problem have been solved in polynomial time by Edmonds [1970, 1979], Lawler [1976], and Frank [1981]. Their algorithms start with $I = \emptyset$ and increase the cardinality of I by one at each iteration until an optimum set is found.

We discuss a new algorithm for (P_k), one which operates in a "horizontal" fashion. It starts with a feasible k-set I. The set I is used to construct a digraph related to Glover's state graphs [1985] and Lawler's border graphs [1976]. We then use negative length cycles in our digraph to find an improved feasible k-set. (In this paper all cycles and paths in digraphs are assumed to be directed.) The procedure is repeated until an optimum set is found. In constructing our digraph from I at each iteration, the following circuit recognition problem arises in matroids M_1 and M_2. Given a set $I \subseteq E$, independent in both matroids, and e in E - I, find the circuit of $I + e$ in M_i (or show that none exists) for $i = 1, 2$. We denote by c the complexity of this circuit recognition problem. Note that we use summation notation to represent set union.

This "horizontal" approach, in conjunction with the introduction of artificial elements, yields a second algorithm for problem (P_k). It has complexity $O(nk(k + c + \log n))$, where $n = |E|$. Our second algorithm also solves the weighted matroid intersection problem. It is closely related to Lawler's primal algorithm, the main difference being that, at each iteration, it only requires finding a shortest path between two prespecified nodes in a

digraph with nonnegative arc weights and therefore does not need the more complex labelling procedure used by Lawler. This accounts for the term $nk(k + \log n)$ in our complexity bound instead of Lawler's n^2k^2.

Our development uses concepts and results from Glover's paper on the Generalized Quasi-Greedy Algorithm [1985]. We include them here to make this report self-contained.

2. Preliminary Results:

Let M_1 and M_2 be two matroids defined on the same element set E. Throughout this section we assume that I is independent in both matroids. Let Σ be a subset of I and let Σ' be a subset of E - I such that $|\Sigma| = |\Sigma'|$. We say that (Σ, Σ') is an **I swap** if $I - \Sigma + \Sigma'$ is independent in M_1 and an **I back-swap** if $I - \Sigma + \Sigma'$ is independent in M_2. Also, we call $m(\Sigma, \Sigma')$ a **matching** if it represents a one-one mapping of Σ onto Σ'; $m(\Sigma, \Sigma')$ is called an **I matching** if every (e, e') in $m(\Sigma, \Sigma')$ is an I swap, and an **I back-matching** if every (e, e') in $m(\Sigma, \Sigma')$ is an I back-swap.

Lemma 1: Let I and I' be independent sets in a matroid M. Let $\Sigma = I - I'$ and $\Sigma' = I' - I$. Choose any e in Σ such that $I' + e$ is dependent. Then there is an e' in Σ' such that both $I' + e - e'$ and $I + e' - e$ are independent in M.

Proof: Since $I' + e$ is dependent, it contains a unique circuit $I'(e)$. Clearly, every e' in $I'(e) - e$ will satisfy the requirement that $I' + e - e'$ be independent. We need to show that at least one such e' yields $I + e' - e$ independent.

Note that $I'(e) - e$ is not contained in I and consider $I + I'(e) - e$. But $r(I + I'(e) - e) \geq r(I)$ since e is dependent on $I'(e) - e$, where r denotes the rank function of the matroid. Thus, we can find a subset I'' of $I + I'(e) - e$ which is independent and has the same cardinality as I. Since $I - e$ is independent, we can take $I'' = I - e + e'$ for some e' in $I'(e) - I$. □

Lemma 2: Let I and I' be two feasible k-sets, and let $\Sigma = I - I'$ and $\Sigma' = I' - I$. Then there is at least one I matching and one I back-matching of (Σ, Σ').

Proof: First we show that at least one I matching exists. Choose an e in Σ and select e' in Σ' such that both $I + e' - e$ and $I' + e - e'$ are independent in M_1. Note that such an e' exists by Lemma 1. We then have an I swap (e, e'). We can repeat this arguement with $(\Sigma - e, \Sigma' - e')$ in the role of (Σ, Σ') to get the desired I matching recursively.

To show that an I back-matching exists, it suffices to interchange the role of M_1 and M_2 in the definitions. The result follows from the existence of an I matching for this transformed problem. □

Lemma 3: Let I be an independent set in matroids M_1 and M_2, and consider $\Sigma \subseteq I$ and $\Sigma' \subseteq E - I$ where $|\Sigma| = |\Sigma'|$.

(a) If $m(\Sigma, \Sigma')$ is an I matching but not an I swap, then there also exists an I matching $m'(\Sigma, \Sigma')$ different from $m(\Sigma, \Sigma')$.

(b) If $m(\Sigma, \Sigma')$ is an I back-matching but not an I back-swap, then there also exists an I back-matching $m'(\Sigma, \Sigma')$ different from $m(\Sigma, \Sigma')$.

Proof: Since (b) follows from (a) by interchanging the role of M_1 and M_2 in the definitions, we only have to prove (a).

Let F and F' be contained in Σ and Σ', respectively, such that

(i) the I matching $m(\Sigma, \Sigma')$ induces an I matching of (F, F'),

(ii) $I' = I + F' - F$ is independent in M_1, and

(iii) $I' + f' - f$ is dependent in M_1 for every (f, f') in $m(\Sigma, \Sigma') - m(F, F')$.

Note that $I_0 = I + f' - f$ is independent in M_1. Thus we can apply Lemma 1 to I_0 and I'. Since $I' + f'$ is dependent in M_1, there exists e' in $F' - f'$ such that $I' + f' - e'$ $(= I + F' + f' - e' - F)$ and $I_0 + e' - f'$ $(= I + e' - f)$ are independent in M_1. Thus, $(F, F' + f' - e')$ is an I swap, as is (f, e'). Applying Lemma 2 to $(F, F' + f' - e')$ and adding (f, e') yields an I matching of $(F + f, F' + f')$, containing (f, e'). If we add the remainder of $m(\Sigma, \Sigma')$ to this I matching, we get an I matching different from $m(\Sigma, \Sigma')$. □

Lemma 4: Given a digraph, let C and C' be the arc sets of two different cycles. Assume that the two cycles contain exactly the same nodes and let $C'' = C + C'$, where an arc occurs twice in C'' if it occurs in both C and C'. Then C'' may be written as $C'' = C_1 + \ldots + C_u$, $u \geq 3$, where each C_i induces a cycle containing at least one fewer node than C and C'.

Proof: Assume the nodes contained in the two cycles have been indexed $v_1, v_2, \ldots, v_h$ so that the arcs of C are (v_i, v_{i+1}) for $i = 1, 2, \ldots, h$

(mod h). Since C' ≠ C, there is at least one arc (v_q, v_p) in C' such that q > p, p ≠ 1. Consider the paths connecting the nodes v_1, v_p, and v_q in each of the two cycles, say

(P_{1p}, P_{pq}, P_{q1}) in the cycle induced by C, and

$(P_{1q}', (v_q, v_p), P_{p1}')$ in the cycle induced by C'.

Recombine these paths as follows.

$C_1 = (P_{1p}, P_{p1}')$, $C_2 = (P_{pq}, (v_q, v_p))$, and $C_3 = (P_{1q}', P_{q1})$.

Note that C_1 does not contain v_q, C_2 does not contain v_1, and C_3 does not contain v_p. In addition C_2 and C_3 are cycles since their nodes are indexed in ascending order and, therefore, are not repeated. To decompose C_1 into cycles it suffices to trace it, starting from v_1. Whenever a node is visited for the second time in this process, a cycle of the decomposition is identified. □

Lemma 5: Let s and d be two nodes of a digraph, and let P and P' be the arc sets of two different paths joining the source s to the destination d and containing the same intermediate nodes, but in a different order. Let P" = P + P' where an arc occurs twice in P" if it occurs in both P and P'. Then P" may be written as $P" = P_1 + P_2 + C_3 + \ldots + C_u$, $u \geq 3$, where P_1 and P_2 are paths joining the source s to the destination d, but containing fewer intermediate nodes than P and P', and each C_i is a cycle.

Proof: Identify the nodes s and d of the digraph into a single node x. Then P and P' become cycles, and we can apply Lemma 4. Exactly two of the cycles in the decomposition $C_1 + \ldots + C_u$, $u \geq 3$, must contain x since there are two arcs of $P + P'$ out of x and two into x. Assume these two cycles are C_1 and C_2. Going back to the original digraph, C_1 and C_2 yield the required paths P_1 and P_2. □

3. The First Algorithm :

In this section, we show how to solve problem (P_k), given a feasible k-set I. (Finding the initial set I is not considered in this section; see the beginning of Section 4.)

We construct a bipartite digraph B(I) = (V, V', A) from I as follows. We get a node for each e in E, namely a node in V for each e in I and one in V' for each e' in E - I. For each e_i in I, we construct an arc (e_i, e_j') with weight $w(e_i, e_j') = w(e_i) - w(e_j')$ provided $I - e_i + e_j'$ is independent in M_1. Similarly, for each e_j' in E - I, we construct an arc (e_j', e_i) with weight $w(e_j', e_i) = 0$ provided $I - e_i + e_j'$ is independent in M_2. Note that our digraph B(I) is closely related to Lawler's border graph. The difference is that, if $I + e_j'$ is independent in M_1, then, in B(I), an edge (e_i, e_j') exists for all e_i in I and, if $I + e_j'$ is independent in M_2, then (e_j', e_i) exists for all e_i in I. (In Lawler's border graph, none of these edges exist.)

Theorem 1: Let I and I' be two feasible k-sets and let $\Sigma = I - I'$ and $\Sigma' = I' - I$. Then the nodes of B(I) associated with $\Sigma + \Sigma'$ induce a set of disjoint cycles in B(I).

Proof: By Lemma 2, there exists at least one I matching and one I back-matching of (Σ, Σ'). For every I swap (e_i, e_j') in the I matching, $I - e_i + e_j'$ is independent in M_1, and, therefore, by construction there is an arc (e_i, e_j') in B(I). Similarly, for every I back-swap (e_i, e_j') in the I back-

matching, $I - e_i + e_j'$ is independent in M_2, and, therefore, by construction there is an arc (e_j', e_i) in B(I). Clearly, when we consider the arcs that arise from the above I matching and I back-matching, we get a union of disjoint cycles of B(I). □

Let C be a cycle of B(I). We define the weight of this cycle, w(C), to be the sum of the weights of the arcs in the cycle. Note that $w(C) = w(\Sigma) - w(\Sigma')$, where Σ and Σ' represent the nodes of C in V and V', respectively.

Theorem 2: Let I be a feasible k-set, and let C be a cycle of B(I). Let the I matching $m(\Sigma, \Sigma')$ represent the arcs of C directed from V to V', and let $m_b(\Sigma, \Sigma')$ be the I back-matching representing those directed from V' to V. If $I - \Sigma + \Sigma'$ is not feasible, then there is a cycle C_0 in B(I) such that

(i) the nodes of C_0 constitute a proper subset of the nodes of C, and

(ii) $w(C) < 0$ implies $w(C_0) < 0$.

Proof: Since $I - \Sigma + \Sigma'$ is not feasible, either (Σ, Σ') is not an I swap, or it is not an I back-swap, or both.

Suppose (Σ, Σ') is not an I swap. Then, by Lemma 3, there is another I matching $m'(\Sigma, \Sigma')$. If we replace the arcs of $m(\Sigma, \Sigma')$ in C by the arcs of $m'(\Sigma, \Sigma')$, we get a new set C' with $w(C') = w(C)$. But C' will be either a set of disjoint cycles or a single cycle. In the first case, each of the disjoint cycles satisfies (i), and (ii) must hold for at least one of these cycles. In

the second case we apply Lemma 4 with cycles C and C'. Then C" = C + C' can be written as $C_1 + \ldots + C_u$, where each C_i satisfies (i). Again, (ii) must hold for at least one of the cycles C_i since $w(C") < 0$ implies $w(C_i) < 0$ for at least one $i = 1, \ldots, u$.

On the other hand, if (Σ, Σ') is not an I back-swap, the theorem follows by a similar proof, using the I back-matching $m_b(\Sigma, \Sigma')$ and another I back-matching $m_b'(\Sigma, \Sigma')$, which we know exists by Lemma 3. □

Combining Theorems 1 and 2, we get the following theorem. It consolidates our previous results and shows the equivalence of solving problem (P_k) and finding negative weight cycles in B(I).

Theorem 3: Assume I is a feasible k-set.

(i) I is optimal for (P_k) if and only if there are no negative cycles in B(I).

(ii) Let C be a negative cycle in B(I) with no negative cycle on a subset of its nodes, and let Σ and Σ' be the nodes of C in V and V', respectively. Then $I' = I - \Sigma + \Sigma'$ is a feasible k-set such that $w(i') > w(I)$.

Proof: Statement (ii) follows directly from Theorem 2 and from the fact that the weight of the new solution is $w(I') = w(I) - w(C)$, which is greater than w(I).

Now we prove (i). Assume there is a negative cycle in B(I). If

$I' = I - \Sigma + \Sigma'$ is not feasible, where Σ and Σ' are the nodes of C in V and V', respectively, then, by Theorem 2, there is another cycle C_0 on a subset of the nodes of C such that $w(C_0) < 0$. Repeating this arguement with C_0 in place of C, we must eventually be able to find a negative cycle such that $I' = I - \Sigma + \Sigma'$ is feasible. But then $w(I') = w(I) - w(C) > w(I)$.

Now assume that I is not optimal. Consider I' such that $w(I') > w(I)$. From Theorem 1 we get a set of disjoint cycles in B(I), the sum of whose weights must be negative. Thus at least one of these cycles, say C, must be a negative cycle in B(I). □

Theorem 3 yields an obvious primal algorithm. We start with a feasible k-set I and use B(I) to find an improved solution. We continue until we find a B(I) with no negative cycles.

The algorithmic issues are (i) the construction of B(I), (ii) when a negative cycle exists, how to find one such that there is no other negative cycle on a subset of its nodes, and (iii) the number of iterations needed to reach optimality. We treat these questions briefly now.

To construct B(I), we must solve the following circuit recognition problem for each e_j' in E - I and for both of the matroids M_i, i = 1, 2: Check whether $I + e_j'$ is dependent in M_i, and, if it is, find the unique cycle of $I + e_j'$. This task requires time c for each e_j' in E - I. So the complexity of constructing B(I) is O(nc).

To find a negative cycle with the required property, one possibility is to use the so-called "matrix multiplication" algorithm for finding shortest paths between all pairs of nodes of a digraph. Define the nxn matrix A so

that $a_{ii} = 0$ and, for $i \neq j$, a_{ij} is the weight of arc (e_i, e_j) if it exists in B(I), ∞ otherwise. It is well-known that, if the usual addition and multiplication are replaced by the operations min and +, then the t^{th} power of A has the following interpretation. Let $a_{ij}(t)$ be an element of A^t. Then $a_{ij}(t)$ is the smallest weight of a path from i to j that contains at most t arcs; see Lawler [1976]. In particular, if $a_{ii}(t) < 0$, then a negative cycle with at most t arcs can be identified in B(I). Also, the smallest value of t such that A^t has a negative diagonal element provides a negative cycle such that no negative cycle exists on a subset of its nodes. Note that, since $|I| = k$ and B(I) is bipartite, at most 2k matrix multiplications are needed. So the complexity of finding a negative cycle with the required property is $O(n^3k)$. Actually, the factor n^3 can be decreased by using fast matrix multiplication algorithms.

Finally, we consider the number of iterations needed to reach optimality. We assume that w(e) is integer for all e in E. Let

$$W = \max\{w(e): e \text{ in } E\} - \min\{w(e): e \text{ in } E\},$$

and assume $W > 0$ (otherwise the problem is trivial). At first it appears that the number of iterations could be as large as kW; however, by using a classical scaling technique on the weights we need at most $k(1 + \lceil \log W \rceil)$ iterations. This is achieved as follows. Add $\min\{w(e): e \text{ in } E\}$ to all the weights. The new weights w'(e) are nonnegative. Let

$$w'(e) = a_p(e)2^p + \ldots + a_0(e),$$

where $p = \lceil \log W \rceil$ and $a_i(e) = 0$ or 1 for $i = 0, 1, \ldots, p$. Instead of solving the problem (P_k) with the weights w'(e), we first solve it with the weights

$a_p(e)2^p$. With these weights at most k iterations are needed to reach an optimum solution, since an improvement of at least 2^p is achieved each iteration and the optimum value is at most $k2^p$. Then we solve the problem (P_k) with the weights $a_p(e)2^p + a_{p-1}(e)2^{p-1}$, starting from the optimum solution found for the previous weights. Again, at most k iterations are needed. We continue this process, including one more digit in the binary expansion of w'(e) each time, until the problem (P_k) is solved for the actual weights w'(e).

Therefore, the overall complexity of the algorithm is $O((n^3k^2 + nkc)(1 + \log W))$. Although this complexity is high, the approach may have merits in the context of sensitivity analysis. Given an optimum solution for some set of weights, Theorem 3 provides an optimality condition that can be tested for a perturbed set of weights. If the optimality condition is violated, reoptimizing using the above algorithm may be more efficient in practice than starting from scratch, as the other algorithms for matroid intersection require.

This algorithm may also be useful when an optimization problem, such as the traveling salesman problem, is solved using a Lagrangian relaxation that happens to be the problem (P_k) for some matroids M_1 and M_2 (the weights being a function of the Lagrange multipliers). To get a tight relaxation one often optimizes the Lagrange multipliers by an iterative method, such as subgradient optimization. Thus, at each iteration the weights are modified. We plan to try this approach for the traveling salesman problem in a forthcoming paper.

4. The Second Algorithm :

In this section we show how the ideas developed in the previous algorithm can be modified to (i) avoid assuming that we are given an initial set to start the algorithm, (ii) reduce the complexity, and (iii) solve the weighted matroid intersection problem as well as problem (P_k).

First, consider the question of getting an initial set for problem (P_k). We can start by applying the greedy algorithm. Initially $I_0 = \emptyset$, and , at each iteration, an element e in $E - I_0$ is added to I_0 if $I_0 + e$ is independent in both matroids and e has maximum weight over all elements f in $E - I_0$ such that $I_0 + f$ is independent in at least one of the two matroids. The greedy algorithm is stopped when such an element e cannot be found. Note that, at termination, $|I_0| \geq 1$. If $|I_0| \geq k$, we are done. Otherwise, consider a set A of $k - |I_0|$ artificial elements, where $E \cap A = \emptyset$. Let $F = E + A$. We 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 $I + J$ 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 will be denoted by $(P_k | F)$.

Note that to define $(P_k | F)$ completely we need to give weights to the artificial elements. By giving them large negative weights we can guarantee that, if (P_k) is feasible, then an optimum solution to $(P_k | F)$ does not contain any artificial elements and therefore is also optimum for (P_k). Thus, the question of finding an initial solution for the algorithm of Section

3 is resolved: it suffices to solve $(P_k|\ F)$ starting from $I_0 + A$ as defined above.

However, in this section we have a different algorithm in mind. We will give large positive weights to all the artificial elements. Then, obviously, the initial solution $I = I_0 + A$ is optimal for $(P_k|\ F)$. So, by Theorem 3, the digraph $B(I)$ relative to problem $(P_k|\ F)$ contains no negative cycles.

Define the digraph $S(I)$ from $B(I)$ by splitting one of the nodes that arose from an artificial element, say a in A, into a source node s and a destination node d, where the arcs out of s in $S(I)$ are those that were out of a in $B(I)$ and the arcs into d in $S(I)$ are those into a in $B(I)$.

Theorem 4: Let I be an optimal k-set of $(P_k|\ H)$, where $E \subseteq H \subseteq F$. Let P be an s-d path in $S(I)$, and let the I matching $m(\Sigma, \Sigma')$ represent the arcs of P directed from V to V' and the I back-matching $m_b(\Sigma, \Sigma')$ represent those directed from V' to V (recall that s and d are both associated with element a). If $I - \Sigma + \Sigma'$ is not feasible, then there is an s-d path P_0 in $S(I)$ such that

(i) the intermediate nodes of P_0 constitute a proper subset of those of P, and

(ii) $w(P_0) \leq w(P)$.

Proof: Since $I - \Sigma + \Sigma'$ is not feasible, either (Σ, Σ') is not an I swap or it is not an I back-swap, or both. Assume it is not an I swap. Then, by Lemma 3, there is another I matching $m'(\Sigma, \Sigma')$. If we replace the arcs of

$m(\Sigma, \Sigma')$ in P by those of $m'(\Sigma, \Sigma')$, we get a new set P' with $w(P') = w(P)$, and P' will be either the union of cycles and one s-d path which are all node disjoint, or a single s-d path. In the first case, the s-d path satisfies (i) and (ii), where (ii) follows from the fact that S(I) does not contain negative cycles. In the second case we apply Lemma 5 with paths P and P'. Then the path P_1 defined in the statement of Lemma 5 satisfies conditions (i) and (ii), again using the fact that S(I) does not contain negative cycles.

When (Σ, Σ') is not an I back-swap the result is proved in a similar fashion. □

Observe that the fact that P joins s to d is unimportant in the proof. Thus, the statement of the above theorem holds for any path P that contains *the same number of elements from V and V'.*

Theorem 5: Let I be an optimal k-set for problem $(P_k | H)$, where $E \subset H \subseteq F$, and let S(I) be obtained from B(I) by splitting an artificial element a into s and d.

(i) Problem $(P_k | H - a)$ has no solution if and only if there is no s-d path in S(I).

(ii) Let P be a shortest s-d path such that every s-d path defined on a proper subset of its nodes has a strictly larger weight, and let Σ and Σ' be the nodes of P in V and V', respectively. (s and d both give rise to a in Σ.) Then $I' = I - \Sigma + \Sigma'$ is an optimal k-set for problem $(P_k | H - a)$.

Proof: We first prove (i). If there is an s-d path P such that the set $I - \Sigma + \Sigma'$ defined in Theorem 4 is not feasible, then there exists another s-d path going through a proper subset of the nodes of P. Repeating Theorem 4 with this new path, we must eventually find a set $I' = I - \Sigma + \Sigma'$ which is feasible for $(P_k|\ H)$. Since a is in Σ, $I' \subseteq H - a$ and, therefore, I' is also feasible for $(P_k|\ H - a)$. Conversely, assume that $(P_k|\ H - a)$ has a solution, say I', and let $\Sigma = I - I'$ and $\Sigma' = I' - I$. Then the I swap (Σ, Σ') induces an s-d path and possibly a set of disjoint cycles in S(I), as a consequence of Theorem 1.

Now we prove (ii). Consider an optimal set I* for $(P_k|\ H - a)$. Define $S = I - I^*$ and $S^* = I^* - I$. The I swap (S, S^*) induces an s-d path, say P*, and possibly a set of disjoint cycles in S(I). Since there are no negative cycles in S(I), it follows that $w(S) - w(S^*) \geq w(P^*)$. In addition, $w(P^*) \geq w(P) = w(\Sigma) - w(\Sigma')$ since P is a minimum length s-d path in S(I). So $w(I^*) \leq w(I')$ since $I^* = I - S + S^*$ and $I' = I - \Sigma + \Sigma'$. Also note that I' is feasible in $(P_k|\ H)$ by Theorem 4, and, since a is in Σ, it follows that I' is feasible in $(P_k|\ H - a)$. So I' is optimal in $(P_k|\ H - a)$. □

Theorem 5 provides the basis for an algorithm. Start with the set $I = I_0 + A$. This set is optimal for problem $(P_k|\ F)$ if we assign a large positive weight to all the artificial elements. Construct B(I), and then S(I) by splitting some artificial element a of B(I). Find a shortest path P from the resulting source s to destination d, with the added condition that every s-d path defined on a proper subset of the nodes of P has a strictly larger

weight. Then the set I' defined in Theorem 5(ii) is an optimal k-set for problem $(P_k|\ F - a)$. Repeat this process with I' in place of I, until all the artificial elements have been split. When this occurs, the resulting k-set is optimal in (P_k).

Note that this algorithm requires at most k iterations. In fact the number of iterations is $k - |I_0|$ since exactly one artificial element is removed at each iteration. (If at some iteration $\Sigma = I - I'$ contained at least one other artificial element b in addition to a, then $I'' = I + b - e$ would be a feasible k-set with larger weight than I' for some element e of $I' - A$, a contradiction.)

We now consider the complexity of finding a shortest s-d path with the required property.

In the first iteration the arc weights in $B(I_0 + A)$ are all nonnegative. To see this, consider first an arc (a, f), where a is artificial. Then $w(a, f) = w(a) - w(f) > 0$ since w(a) has been chosen large enough. Now consider an arc (e, f), where e is in I_0. Since the arc (e, f) occurs in $B(I_0 + A)$, the set $I_0 + f - e$ is independent in M_1. But then $w(e) \geq w(f)$ since e has been chosen by the greedy algorithm in the construction of I_0. So, again, $w(e, f) \geq 0$. Finally, when e is not in $I_0 + A$, $w(e, f) = 0$.

Since all the arc weights are nonnegative in $B(I_0 + A)$, and thus also in $S(I_0 + A)$, a shortest s-d path P can be found by Dijkstra's algorithm. Using the fast implementation of Fredman and Tarjan [1984], the complexity of this step is at most $O(kn + n \log n)$. Furthermore, the restriction that every

path on a subset of the nodes of P has a strictly larger weight can be obtained by adding a very small positive ϵ to all the arc weights of $S(I_0 + A)$.

Now we show that nonnegative arc weights in B(I) can be preserved throughout the algorithm.

Define a variable D(e) associated with each node of the digraph B(I) and a reduced weight $w'(e, f) = w(e, f) + D(e) - D(f)$ associated with each arc of B(I). In terms of the reduced weights, the length of an s-d path P is $w'(P) = w(P) + D(s) - D(d)$ since, for any intermediate node e of P, the variable D(e) cancels out on the two arcs of P that contain e. But D(s) - D(d) is a constant that does not depend on P. This means that it is equivalent to solve the shortest path problem in B(I) with the reduced weights w' instead of the original weights w. The next theorem provides a choice for the variables D(e) that guarantees nonnegative reduced weights from one iteration to the next. This result is closely related to Theorem 4 in Glover [1985].

Theorem 6: Let I be an optimal k-set for problem $(P_k | H)$ where $E \subset H \subseteq F$, and assume that an s-d path exists in S(I). Set D(e) to be the length of a shortest s-e path in S(I), for every e in H. Let I' be defined as in Theorem 5(ii). Then the reduced weights $w'(e, f) = w(e, f) + D(e) - D(f)$ are nonnegative for every arc of the digraph S(I').

Proof: First we show that the choice of D(e) gives nonnegative reduced weights on the arcs of S(I). Consider f in H. Since D(f) is the length of a shortest s-f path, we must have $D(f) \leq D(e) + w(e, f)$ for every e in H such that (e, f) is an arc of S(I). This proves the result for S(I).

Let P be the shortest s-d path that gives rise to l' in Theorem 5(ii), and let P_i be a subpath of P from s to some node e_i. Consider the l matching $m(\Sigma_1(i), \Sigma_1'(i))$ induced by the arcs of P_i going from V to V' and the l back-matching $m_b(\Sigma_2(i), \Sigma_2'(i))$ induced by the arcs of P_i going from V' to V. The elements of these two matchings induce subpaths of P. By Theorem 4, $l_1(i) = l - \Sigma_1(i) + \Sigma_1'(i)$ and $l_2(i) = l - \Sigma_2(i) + \Sigma_2'(i)$ are feasible. (The statement of Theorem 4 holds for any path P that contains the same number of elements from V and V', as was already observed.)

We construct a digraph N_i as follows. The nodes of N_i are the same as those of S(l). If (e, e') is an $l_1(i)$ swap, then (e, e') is an arc of N_i; and if (e, e') is an $l_2(i)$ back-swap, then (e', e) is an arc of N_i. Note that N_i is not bipartite in general.

We will show that the choice of D(e) gives nonnegative reduced weights on the arcs of N_i. This is done by induction on the nodes of P, starting from $e_i = s$. Note that when we reach $e_i = d$, we will have $N_i = S(l')$, and the theorem will be proved.

When $e_i = s$, we have $N_i = S(l)$, and the result has been proved above. Now let e_i be a node of P for which the result holds, and let e_j be the node following e_i on the path P. There are two possibilities: $e_i \in V$ and $e_j \in V'$, or $e_i \in V'$ and $e_j \in V$.

First suppose that e_i is in V and e_j is in V'. Then $l_2(j) = l_2(i)$ and, therefore, the arcs that arise from $l_2(j)$ back-swaps in N_j are the same as

those in N_i. Thus, they have nonnegative reduced weights by the induction hypothesis. On the other hand, $I_1(j) = I_1(i) - e_i + e_j$. Since the arc (e_i, e_j) is on the shortest s-d path P, we have the following equation:

$$D(e_j) = D(e_i) + w(e_i) - w(e_j). \qquad (*)$$

Consider any arc (e_p, e_q) that is an $I_1(j)$ swap, but not an $I_1(i)$ swap. So $I^* = I_1(j) - e_p + e_q = I_1(i) - e_p + e_q - e_i + e_j$ is independent in the first matroid.

If $(e_p, e_q) = (e_j, e_i)$, then $w(e_p) - w(e_q) + D(e_p) - D(e_q) = 0$, as a consequence of equation (*).

If $e_p = e_j$ and $e_q \neq e_i$, then (e_i, e_q) is an $I_1(i)$ swap. Therefore, $w(e_i) - w(e_q) + D(e_i) - D(e_q) \geq 0$ by induction. Combining this inequality with (*) we get $w(e_p) - w(e_q) + D(e_p) - D(e_q) \geq 0$.

A similar arguement 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 an $I_1(i)$ swap, yet I^* is independent in the first matroid, it follows from Lemmas 2 and 3 that both (e_p, e_j) and (e_i, e_q) are $I_1(i)$ swaps. This gives the two inequalities

$$w(e_p) - w(e_j) + D(e_p) - D(e_j) \geq 0 \text{ and } w(e_i) - w(e_q) + D(e_i) - D(e_q) \geq 0.$$

Summing and using (*) we get $w(e_p) - w(e_q) + D(e_p) - D(e_q) \geq 0$. This completes the proof when e_i is in V and e_j is in V'.

Now suppose that e_i is in V', and e_j is in V. Then $I_1(j) = I_1(i)$, and,

therefore, the arcs of N_j that arise from $I_1(j)$ swaps have nonnegative reduced weights. On the other hand, $I_2(j) = I_2(i) - e_j + e_i$. Since (e_i, e_j) is on the shortest s-d path P, we have the equation

$$D(e_i) = D(e_j). \qquad (**)$$

Consider any arc (e_p, e_q) that is an $I_2(j)$ back-swap but not an $I_2(i)$ back-swap.

If $(e_p, e_q) = (e_j, e_i)$, then the reduced weight of arc (e_p, e_q) is $D(e_p) - D(e_q) = 0$ as a consequence of (**).

If $e_p = e_j$ and $e_q \neq e_i$, then (e_i, e_q) is an $I_2(i)$ back-swap. Therefore, its reduced weight satisfies $D(e_i) - D(e_q) \geq 0$ by induction. This and equation (**) show that $D(e_p) - D(e_q) \geq 0$. The proof is similar when $e_p \neq e_j$ and $e_q = e_i$.

Finally, if $e_p \neq e_j$ and $e_q \neq e_i$, it follows from Lemmas 2 and 3 that (e_p, e_j) and (e_i, e_q) are $I_2(i)$ back-swaps. Thus, $D(e_p) - D(e_j) \geq 0$ and $D(e_i) - D(e_q) \geq 0$ by induction. Using (**) once more, we get $D(e_p) - D(e_q) \geq 0$.

Thus, the reduced weights of all the arcs of N_j are nonnegative. □

The variables D(e) needed in Theorem 6 to guarantee nonnegative reduced arc weights are actually computed in the course of finding a shortest s-d path in S(I). Thus no extra computations are needed. We summarize the complexity of the algorithm: O(k) iterations and, in each iteration, time O(nc) for constructing S(I) and time O(nk + n log n) for

finding a shortest s-d path and updating the variables D(e). Therefore, the overall complexity of the algorithm is $O(nk(k + c + \log n))$.

The algorithm provides an optimum solution to $(P_k|\ H)$ for every $E \subseteq H \subseteq F$. In terms of the initial set E, this means that we have an optimum solution of (P_h) for every $h \leq k$. Therefore, by putting $k = r$, the algorithm can be used to solve the weighted matroid intersection problem. It suffices to keep the best solution generated in the course of the algorithm.

In this section our motivation for introducing $k - |I_0|$ artificial elements with large positive weights was to capitalize on the results of Section 3. However, in practice, the construction of S(I) can be simplified by adding only one artificial source node s and one artificial sink node d to B(I) at each iteration. The arcs (e_i, e_j') can be removed from S(I) if $I + e_j'$ is independent in M_1 for any e_i in I. The reason for this is that a shortest s-d path would go directly from s to e_j' as a consequence of the existence of nonnegative reduced weights. Similarly, the arcs (e_j', e_i) can be removed if $I + e_j'$ is independent in M_2, for any e_i in I, as a shortest s-d path would go directly from e_j' to d. To sum up, our algorithm can be applied to Lawler's border graph with an added source node s joined to every node e_j' in $E - I$ such that $I + e_j'$ is independent in M_1, and a destination node d joined to every node e_j' such that $I + e_j'$ is independent in M_2. The goal is the same at each iteration as in Lawler's primal algorithm. The increased efficiency of our algorithm can be attributed to the use of nonnegative reduced weights in the search for shortest paths.

References :

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

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

A. Frank, "A Weighted Matroid Intersection Theorem," Journal of Algorithms 2 (1981), 328 - 336.

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.

F. Glover, "The Generalized Quasi-Greedy Algorithm for Constrained Minimum Weight Matroid Bases," MSIS Series, Report No. 85-1, Graduate School of Business, University of Colorado, Boulder (1985).

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

END

FILMED

8-85

DTIC