Operations Research Letters 9 (1990) 315–318
North-Holland

September 1990

AN ALGORITHM FOR COMPUTING MAXIMUM SOLUTION BASES

Refael HASSIN

Statistics Department, Tel-Aviv University, Tel-Aviv 69978, Israel

Received February 1989
Revised July 1989

We consider a family of problems defined on a common solution space. A problem is characterized by a subset of the solution space whose elements are defined to be feasible for that problem. Each solution is associated with a cost. Solving a problem means finding a feasible solution of minimum cost. It is assumed that an algorithm for solving any single problem is available. We show how to solve all of the problems in the family by selecting and solving a small subset of them.

computational complexity * combinatorics * flow algorithms

1. Introduction

This note is concerned with the solution of a set of problems defined on a common solution space. One can imagine a set of candidate *solutions* such as a set of points in R^n. Each solution is associated with a *cost*. There is also a set of *problems* and each of them is associated with a set of *feasible solutions*. The *value* of a problem is defined to be the minimum cost of a feasible solution to that problem. Solutions with 'low' cost will typically be optimal for several problems. For example, a solution with a minimum cost among all candidate solutions will be optimal for all of the problems for which it is feasible.

In another paper (Hassin [8]) the author stated a theorem characterizing the number of *distinct* values such a set of problems may possess. The theorem also supplies a compact representation of these solution values, called a *maximum solution basis*, that allows to compute easily the value of any of the problems. However, to construct this representation one may have to solve all of the problems. This can be a time consuming task as the problems may be hard to solve.

The first special case of the above setting that appears in the literature is the *multiterminal cut problem* that was solved by Gomory and Hu [3]. They characterized the maximum solution bases for the set of minimum cut problems with different source–sink pairs, defined on undirected networks with edge capacities. They also showed how to construct a maximum solution basis by solving a number of problems equal to the upper bound on the number of distinct solution values the problems may have. Gusfield [5] has shown how to simplify the algorithm by avoiding node contractions. Gomory and Hu's results were extended by Granot and Hassin [4] and Gusfield and Naor [7] to networks with both edge and node capacities and by Gusfield and Naor [6] to a compact representation of *all* of the minimum cuts for each pair of elements. Hassin [8] extended these results to cuts of arbitrary costs, but with a procedure that requires to solve $O(n \log n)$ problems where n is an upper bound on the number of distinct solution values.

In this note it is shown how to construct a maximum solution basis by solving a number of problems that is at most twice the size of the upper bound. This procedure applies to any case that fits into the general model.

2. Solution bases

Let c be a real valued function on a set X. Let X_i, $i \in S$, be nonempty subsets of X, where S is an index set with finite cardinality. Define the *cost* (or *value*) of X_i by

$$c_i = \min\{ c(x) \mid x \in X_i \}.$$

Let $M(S)$ be the number of distinct values of c_i, $i \in S$. For $x \in X$, let $a_i(x) = 1$ if $x \in X_i$ (i.e., x is *i-feasible*) and $a_i(x) = 0$ otherwise.

For our purpose the set X is the set of candidate solutions to a family of problems, and X_i is the set of i-feasible solutions, $i \in S$. The binary matrix $A = (a_i(x))$, called the *solution matrix*, defines the feasibility relations, and the cost c_i is the minimum cost of an i-feasible solution (i.e., the cost of an *i-optimal* solution).

Call a set $S' \subseteq S$ *dependent* if there exists $S'' \subseteq S'$ such that $\sum_{i \in S''} a_i(x) = 0 \pmod 2$ $\forall x \in X$. Otherwise S' is *independent*. Let $r(A)$ be the *rank* of A in the binary field, i.e., the maximum cardinality of an independent subset of S.

Theorem 2.1 (Hassin [8]). $M(S) \leqslant r(A)$.

Let $S' \subseteq S$ correspond to a maximal set of independent rows of A; then we call S' a *solution basis*. The cardinality of each solution basis is equal to $r(A)$. Define the *value of a solution basis* S' as $\sum_{i \in S'} c_i$. A *maximum solution basis* is a solution basis with maximum value.

Theorem 2.2 (Hassin [8]). *Let $S' \subseteq S$ be a maximum solution basis. Let $k \in S \setminus S'$, and let $S'' \subseteq S'$ satisfy $a_k(x) = \sum_{i \in S''} a_i(x) \pmod 2$ $\forall x \in X$. Then $c_k = \min\{c_i \mid i \in S''\}$ and there exists $p \in S''$ and $y \in X_p \cap X_k$ such that $c(y) = c_k$.*

In view of Theorem 2.2, a maximum solution basis contains all the information needed to compute c_k for every $k \in S$. It also gives clues to locate a k-optimal solution. As a matter of fact, if the values $c(x)$ are distinct for all $x \in X$ then this solution is the unique element $y \in X$ satisfying $c(y) = \min\{c_i \mid i \in S''\}$. If these values are not distinct then any solution $y \in X$ with cost c_k which has an odd column sum in the rows of A corresponding to S'' can be shown to be k-optimal.

When all of the values c_i, $i \in S$, are given, a maximum solution basis can be computed by applying the greedy algorithm. However, generating all of the c_i values by solving all of the problems $i \in S$ may be a time consuming task and we are interested in a procedure that will guarantee that a maximum solution basis will be found by solving just a portion of these problems.

3. Computing maximum solution bases

We start by making the following simplifying assumption:

Assumption 3.1. Distinct solutions have distinct costs.

We note that if the above assumption does not initially hold one can perturb the costs or rank solutions with identical costs in some way to comply with the requirement of the assumption. (See for example, Eaves and Rothblum [2].)

Our algorithm consists of two parts. In the first an initial set of solutions, X', is generated. Each member of this set is optimal for some problem in S. The problems to be solved are chosen sequentially so that none of the previously generated solutions is feasible for the current problem. This guarantees a new optimal solution in each iteration. This initialization algorithm terminates when X' is a *cover* of S in the sense of the following definition:

Definition 3.2. A subset $X' \subset X$ is said to cover a subset $S' \subset S$ if for every $j \in S'$, X' contains a j-feasible solution.

The second, and main, part of the algorithm applies a greedy algorithm to compute a maximum solution basis of A. The greedy algorithm finds in each iteration a problem of maximum value whose row in A can be added to the current set of rows and maintain independence of this set.

Algorithm 3.3.
Input: A.
Output: $X' \subset X$ [a cover of S].
3.3.1: Set $X' = \emptyset$, $S' = S$.
3.3.2: If $S' = \emptyset$, stop. Else, choose $i \in S'$.
3.3.3: Solve i to obtain an i-optimal solution $\hat{x}$.
3.3.4: Set $X \leftarrow X \cup \{\hat{x}\}$.
Set $S' \leftarrow S' \setminus \{j \in S' \mid a_j(\hat{x}) = 1\}$.
Go to 3.3.2.

Algorithm 3.4
Input: A, $X' \subset X$ [a cover of S].
Output: T [a maximum solution basis of A].
3.4.1: Set $T = \emptyset$.
3.4.2: Set
$S' = \{j \in S \mid T \cup \{j\}$ *is independent*$\}$.
If $S' = \emptyset$, stop.

3.4.3: For each $j \in S'$, let

$$c'_j = \min\{ c(x) \mid x \in X',\ a_j(x) = 1 \}.$$

Let $i \in S'$ satisfy

$$c'_i = \max\{ c'_j \mid j \in S' \}.$$

3.4.4: Solve i to obtain an i-optimal solution x.
Case (i). If $\hat{x} \notin X'$, set

$$X' \leftarrow X' \cup \{ \hat{x} \}$$

and go to 3.4.3.
Case (ii). If $\hat{x} \in X'$, set

$$T \leftarrow T \cup \{ i \}$$

and go to 3.4.2.

Theorem 3.5. *Algorithm* 3.4 *generates a maximum solution basis.*

Proof. The set of solution bases is exactly the set of bases of the binary matroid defined by A. Hence the greedy algorithm computes an optimal solution. To show that Algorithm 3.4 is indeed the greedy algorithm we must prove that in Case (ii) of 3.4.4,

$$c_i = \max\{ c_j \mid j \in S' \}.$$

Case (ii) obtains when $\hat{x} \in X'$. Clearly, for all $j \in S'$, $c'_j \geqslant c_j$, so that

$$c_i = c(\hat{x}) = c'_i = \max\{ c'_j \mid j \in S' \}$$
$$\geqslant \max\{ c_j \mid j \in S' \}.$$

Since $i \in S'$, equality holds, as required. □

Remark 3.6. The total number of problems solved in 3.3.3 and 3.4.4 is at most $M(S) + r(A)$, since each problem either has a distinct optimal solution that is used to augment X' (3.3.3 and Case (i) of 3.4.4), or the problem is used to augment the independent set T (Case (ii) of 3.4.4). By Assumption 3.1, the first case can occur at most $M(S)$ times, and by the definition of $r(A)$, the latter can occur at most $r(A)$ times. By Theorem 2.1, this number is at most $2r(A)$.

4. Concluding remarks

Algorithms 3.3 and 3.4 require $O(|S| r(A))$ operations to execute 3.3.4 and 3.4.3 (note that after the first time 3.4.3 is executed, c'_j is revised by a single comparison to $c(\hat{x})$ whenever $a_j(\hat{x}) = 1$). Most of the effort may be devoted to solve the $O(r(A))$ single problems in 3.3.3 and 3.4.4, and to construct the sets S' in 3.4.2. Our results are especially useful when the latter effort is small relative to the savings resulting from the reduction in the number of single problems to be solved.

As a result of this observation, we are interested in cases where the matrix A need not be represented explicitly, and the sets S' in 3.4.2 can be constructed efficiently. Consider, as an example, the multiterminal cut problem. Here, a set $N = \{1, \ldots, n\}$ is given with a cost $c(M, \overline{M})$ for every cut $(M, \overline{M})$ of N. The i–j problem is to find the cut of minimum cost among all cuts separating i and j. An explicit representation of A in this case has a dimension $\binom{n}{2}(2^{n-1} - 1)$. However, the following theorem can be used to avoid this explicit representation.

Theorem 4.1 (Hassin [8]). *Let $G = (N, E)$ be a complete graph with a node set N. Associate with edge (i, j) of G the i–j cut problem. A set of problems $R \subset S$ is independent if and only if the edges of G associated with R do not contain a cycle.*

It follows from the theorem that in this instance the sets S' of 3.4.2 are easily constructed in polynomial time, without explicitly scanning A. The number of problems to be solved is $O(n)$ improving the $O(n \log n)$ bound of Hassin [8]. This may be a considerable saving in computation time, especially when each problem is NP-hard (for example, in the max-cut case). Another example were the independent sets are compactly characterized is mentioned in Hassin [9].

Finally, we note that recent papers by Cheng and Hu [1,10] show how to further reduce the number of problems to be solved in a multiterminal cut problem with arbitrary costs to $n - 1$.

References

[1] C.K. Cheng and T.C. Hu, "Maximum concurrent flow and minimum ratio cut", Technical Report Number CS88-141, Department of Computer Science and Engineering, University of California, San Diego, CA, December 1988.

[2] B.C. Eaves and U.G. Rothblum, "A theory on extending algorithms for parametric problems", SOL 85-13, Department of Operations Research, Stanford University.

[3] R.E. Gomory and T.C. Hu, "Multi-terminal network flows", *J. SIAM* **9**, 551–570 (1961).

[4] F. Granot and R. Hassin, "Multi-terminal maximum flows in node-capacitated networks", *Discrete Appl. Math.* **13**, 157–163 (1986).

[5] D. Gusfield, "Very simple-methods for all pairs network flow analysis", *SIAM J. Comput.* **19**, 143–155 (1989).

[6] D. Gusfield and D. Naor, "Extracting maximal information on sets of minimum cuts" (Extended abstract), Computer Science Division, University of California, Davis, October 1988.

[7] D. Gusfield and D. Naor, "Efficient algorithms for generalized cut trees", Technical Report CSE-89-5, Computer Science Division, University of California, Davis, CA, 1989.

[8] R. Hassin, "Solution bases of multiterminal cut problems", *Math. Oper. Res.* **13**, 535–542 (1988).

[9] R. Hassin, "Multiterminal xcut problems", Presented in the NATO/ARW on *Topological Network Design*, June 1989, to appear in *Ann. Oper. Res.*

[10] C.K. Cheng and T.C. Hu, "Ancestor tree for arbitrary multi-terminal cut functions", to appear in *Ann. Oper. Res.*