41/2002

Raport Badawczy

Research Report

RB/67/2002

Computing the Sets of *K*-Best Solutions for Discrete Optimization Problems

M. Gajda

Instytut Badań Systemowych
Polska Akademia Nauk

Systems Research Institute
Polish Academy of Sciences

POLSKA AKADEMIA NAUK

Instytut Badań Systemowych

ul. Newelska 6

01-447 Warszawa

tel.: (+48) (22) 8373578

fax: (+48) (22) 8372772

Kierownik Pracowni zgłaszający pracę:
Prof. dr hab. inż. Krzysztof Kiwiel

Warszawa 2002

Computing the sets of K-best solutions for discrete optimization problems

Magdalena Gajda
System Research Institute
Polish Academy of Science

December 23, 2002

1 Abstract

We present Lawer[1] procedure for finding the K-best solutions of discrete optimization problem and alternative Hamacher and Queyranne [4] approach. Then we introduce a new algorithm which is based on *branch-and-bound* method.

2 Keywords

Discrete optimization, K-best solutions, branch-and-bound method.

3 Introduction

In some cases it is useful to determinate not only the best solution but also 2nd the best,...,Kth best solution to a given problem. For example, when we add some restrictions which are not included in original problem and verify obtained solutions [1, 3].

We will consider a discrete optimization problem (P):

$$min\{f(x) : x \in \mathcal{S}\} \qquad \text{(P)}$$

$$x = (x_1, x_2, \cdots, x_n) \in \mathcal{S} \subseteq \mathbb{B}^n,$$

where $\mathbb{B}^n = \{0, 1\}^n$. The set of K-best solutions for discrete optimization problem (P) is formulated as follows. For given positive integer K any set $\mathcal{S}(K) \subseteq \mathcal{S}$, such that for any $x \in \mathcal{S}(K)$ and $y \in \mathcal{S} \backslash \mathcal{S}(K)$ the inequality $f(x) \leq f(y)$ holds, is called the set of K-best solutions of the problem (P).

3.1 First Basic Computational Procedure

This simple computational procedure which ranks solutions from the first to the Kth has been proposed by Lawer [1]. Assume that if the feasible solution does not exist for some fixed values of variables, the value of an optimal solution is taken to be $+\infty$.

Step 0 (Start) Compute an optimal solution, without fixing the values of any variables, and place this solution in LIST as the only entry. Set $k = 1$.

Step 1 (Output ***kth*** solution) Remove the least costly solution from LIST and output this solution, denoted $x^{(k)} = (x_1^{(k)}, x_2^{(k)}, ..., x_n^{(k)})$, as the kth solution.

Step 2 (Test k) If $k = K$, stop; the computation is completed.

Step 3 (Augmentation of LIST) Suppose, without loss of generality, that $x^{(k)}$ was obtained by fixing the values of $x_1, x_2, ..., x_s$. Leaving these variables fixed as they are, create $n - s$ new problems by fixing the remaining variables as follows:

$$\begin{array}{ll}
(1) & x_{s+1} = 1 - x_{s+1}^{(k)}, \\
(2) & x_{s+1} = x_{s+1}^{(k)},\ x_{s+2} = 1 - x_{s+2}^{(k)}, \\
(3) & x_{s+1} = x_{s+1}^{(k)},\ x_{s+2} = x_{s+2}^{(k)},\ x_{s+3} = 1 - x_{s+3}^{(k)}, \\
\vdots & \vdots \\
(n-s) & x_{s+1} = x_{s+1}^{(k)},\ x_{s+2} = x_{s+2}^{(k)},\ \cdots,\ x_{n-1} = x_{n-1}^{(k)},\ x_n = 1 - x_n^{(k)}.
\end{array}$$

Compute optimal solutions to each of these $n - s$ problems and place each of the $n - s$ solutions in LIST, together with a record of the variables which were fixed for each of them. Set $k = k + 1$. Go to ***Step*** 1.

The branching operation (in *Step* 3) excludes $x^{(k)}$, from further consideration. Lawer [1] describes also an application of this procedure into the ranking of the K shortest paths between two designated nodes of a network.

3.2 Second Basic Computational Procedure

This procedure has been proposed by Hamacher and Queranne [4]. Let f be an objective function discrete optimization problem and let $\mathcal{S}$ be the finite set of all feasible solutions. For any subset $\mathcal{S}' \subseteq \mathcal{S}$ let $OPT(\mathcal{S}')$ be the set of all optimal solutions restricted to $\mathcal{S}'$, i.e. the objective value of any $x \in OPT(\mathcal{S}')$ is better than or equal to the objective value of any $y \in \mathcal{S}'$.

We start with partition $\{\mathcal{S}\}$ of $\mathcal{S}$ and calculate the best solution $x_1 \in OPT(\mathcal{S})$ and the second best solution y_1 of $\mathcal{S}$. In the i-step of the algorithm we have a partition PART of $\mathcal{S}$ into i sets $\mathcal{S}_1, \cdots, \mathcal{S}_i$, and $x_v \in OPT(\mathcal{S}_v)$ $(v = 1, \cdots, i)$ such that $\{x_1, \cdots, x_i\}$ is an i-optimal solution set of $\mathcal{S}$. Moreover, we know the second best solution $y_v \in \mathcal{S}_v$ for all $\mathcal{S}_v$ with $|\mathcal{S}_v| > 1$. Thus,

$$y_j \in OPT\{y_v : |\mathcal{S}_v| > 1, v = 1, \cdots, i\}$$

is an $(i+1)$-best solution in $\mathcal{S}$. Next, we part $\mathcal{S}_j$ into two sets $\mathcal{S}^{(1)}$ and $\mathcal{S}^{(2)}$ with $x_j \in \mathcal{S}^{(1)}$ and $y_j \in \mathcal{S}^{(2)}$. Thus, x_j and y_j are best solutions in $\mathcal{S}^{(1)}$ and $\mathcal{S}^{(2)}$. For $i = 1, 2$, if $|\mathcal{S}^{(i)}| > 1$ we calculate the second best solution, replace $\mathcal{S}_j$ by $\mathcal{S}^{(1)}$ and $\mathcal{S}^{(2)}$ and continue with a new partition.

4 *Branch-and-bound* method for determining K-best solutions

4.1 *Branch-and-Bound* tree

We consider modification of *branch-and-bound* method to compute K-best solutions. This method ranks solutions from the best to the K-best, for predetermined positive integer K.

The dynamically generated *Branch-and-Bound Tree* (**BBT**) consists of nodes which corresponds to fixed values of variables. At first, the **BBT** has only one node: *root*, which corresponds to the state when none of variables is fixed. The **BBT** is expanded by branching on fixed variables.

For example consider the problem:

$$min\{f(x) : x \in \mathcal{S}\}. \tag{1}$$

If $\mathcal{S} \subseteq \{0,1\}^3$, we first divide $\mathcal{S}$ into $\mathcal{S}_0 = \{x \in \mathcal{S} : x_1 = 0\}$
and $\mathcal{S}_1 = \{x \in \mathcal{S} : x_1 = 1\}$. Then we divide $\mathcal{S}_0$ in to $\mathcal{S}_{00}$ and $\mathcal{S}_{01}$ as well as $\mathcal{S}_1$ in to $\mathcal{S}_{10}$ and $\mathcal{S}_{11}$, and so on.

4.2 Description of the algorithm

Our problem (P) can be presented as follows:

$$min\{f(x) : x \in \mathbb{S} = \bar{\mathbb{S}} \cap \mathbb{B}^n\}, \qquad (P')$$

where:

$$min\{f(x) : x \in \bar{\mathbb{S}}\}$$

denotes the continuous relaxation of (P).

The modified *branch-and-bound* algorithm is started by calling the procedure EXPLORE, which has three parameters: the node X^0, the list of nodes X and the set L. Initially the list X as well as the set L are empty and node X^0 is a root. During calculations the set L stores the best of found solutions. When the algorithm is completed and $|\mathbb{S}| > K$, then L contains K-best solutions discrete optimization problem (P) otherwise L contains k-best solutions of problem (P) where $k = |\mathbb{S}|$. The set L is also used to give the treshold value U in following way: if L contains K solutions then U is equal to the $\max\{f(l) : l \in L\}$; otherwise U is defined as equal to infinity. The list X include nodes which will be evaluated. The **BBT** is expanded by fixing values of variables.

Procedure EXPLORE solves the discrete optimization problem relaxation i.e. the problem:

$$min\{f(x) : x \in \bar{\mathbb{S}} \cap X^0\},$$

denoted by $(f, \bar{\mathbb{S}}, X^0)$, where $X^0 = \{x \in \mathbb{S} :$ value x_i is fixed fore some $i \in \{1, \cdots, n\}\}$ is a node chose in X. If for the obtained solution s: $f(s) < U$ then X^0 is added to list X. If concurrently $s \in \{0,1\}^n$ then s is added to set L.

If X^0 is not a root then procedure EXPLORE is called recurrently with parameters: the node $X^0 \cup \{\bar{x}\}$, where $\bar{x} = 1 - x$, for last fixing value of variable x, the list X and the set L. Next the node X^0 is removed from the list X.

If a list X is not empty then algorithm chooses the new node X^0 from X as follows. The X^0 is this node from the X for which the solution $f(s)$ is minimal. Then a branching variable x and value of variable x are determined, and the procedure EXPLORE is called recurrently with parameters: the node $X^0 \cup \{x\}$, the list X and the set L.

The formal description of the modified *branch-and-bound* algorithm is given below.

```
input Discrete optimization problem (f, S) and some integer K.
output A set S(K) of K-best solutions (f, S) problem.

procedure EXPLORE(X^0, X, L)
  begin
    s ∈ argmin{f(x) : x ∈ S̄} for given X^0
    if f(s) < U then
      'add X^0 to X';
      if k = K then
        if s ∈ {0,1}^n, then
          'remove from L the most costly solution';
          L = L ∪ {s}; U = max{f(s) : s ∈ L};
      else
        if s ∈ {0,1}^n, then
        L = L ∪ {s}; k = k + 1;
    end
  if X^1 ∈ X and X^0 ≠ ∅ then
    'remove X^1 from list X';
    EXPLORE(X^1 ∪ {x̄}, X, L);
  if |X| ≠ 0 then
    begin
    'find X^0 ∈ X for which obtained solution f(s) is minimal';
    'determinate a branching variable x';
    X^1 = X^0;
    EXPLORE(X^0 ∪ {x}, X, L);
end;

begin
  U = +∞;
  S(K) = ∅;
  EXPLORE(∅, ∅, S(K));
end
```

5 Conclusions

In my future work I will modify the procedure of computing the solution of discrete optimization problem in $CPLEX6$. I intend to apply the above algorithm and obtain procedure which computes the K-best solutions of binary linear programming problem.

References

[1] E.L. Lawer - A procedure for computing the k-best solutions to discrete optimization problems and its applications to the shortest path problem. Managment Science 18 (1972) 401-407.

[2] P. J. Brucker, H. W. Hamacher - K-optimal solution sets for some polynomially solvable scheduling problems. European Journal of Operational Research 41 (1989) 194-202.

[3] E. S. van der Poort, M. Libura, G. Sierksma, J. A. A. van der Veen - Solving the k-best traveling salesman problem. Computers & Operations Research 26 (1999) 409-425.

[4] H. W. Hamacher, M. Queyranne - k-best solutions to combinatorial optimizations problems. Annals of Operations Research 4 (1985) 123-143.

[5] U. Derigs - Some basic exchange properties in combinatorial optimization and their application to constructing the K-bes solutions. Discrete Applied Mathematics 11 (1985) 129 – 141.