Journal of Global Optimization (2005) 31: 353–370
DOI 10.1007/s10898-004-1687-x

An Approach to the Subproblem of the Cutting Angle Method of Global Optimization

URFAT G. NURIYEV
Department of Mathematics, Ege University, 35100 Bornova, Izmir, Turkey
(e-mail: urfat@sci.ege.edu.tr)

(Received 1 August 2002; accepted in revised form 12 February 2004)

Abstract. The solution of the Subproblem of the Cutting Angle Method of Global Optimization for problems of minimizing increasing positively homogeneous of degree one functions is proved to be NP-Complete. For the proof of this fact we formulate another problem which we call "Dominating Subset with Minimal Weight". This problem is also NP-Complete. An $O(n^2)$-time algorithm is presented for approximate solution of Dominant Subset with Minimal Weight Problem. This problem can be expressed as a kind of Assignment Problem in which it is allowed to assign multiple tasks to a single processor. Experimental analysis of the algorithm is performed using the program implemented in ANSI-C. The results of the analysis show the efficiency of the proposed algorithm.

Mathematics Subject Classification (2000): 65K05; 90C27; 68Q25

Key words: Assignment Problem, Cutting Angle Method, Dominant Subset with Minimal Weight Problem, Global Optimization Problem, Heuristic Algorithm, Knapsack Problem, NP-Complete

1. Introduction

Cutting Angle Method described in papers [1, 2, 4–6, 10–12] was developed for solving a broad class of Global Optimization Problems. This method is an iterative one requiring the solution of a subproblem (minimizing functions $f(x)$, defined on the set $S = \{x : \sum_{i=1}^{n} x_i = 1, x_i \geqslant 0, i = 1, \ldots, n\}$, where $x = (x_1, \ldots, x_n)$), which is, generally, a Global Optimization Problem. Different algorithms based on discrete programming, dynamic programming and nonsmooth analysis techniques were proposed for the solution of this subproblem in papers [1, 4–6, 11, 12]. While the size of the input increases, it takes a long time to obtain the solution by those algorithms. In paper [3] two algorithms are given to solve the subproblem. The first algorithm produces optimal solution for a special case, and for all the other cases a heuristic way is pursued for the solution.

In this paper we study some properties of the optimal solutions of the subproblem and by means of these properties we prove that this problem

is equivalent to a problem of Boolean programming, which we call "Dominant Subset of Minimal Weight" (in the following DSMWP). The last problem can be used in other situations as well. By transformation of this problem to the Knapsack problem we prove that it is NP-Complete, therefore the subproblem considered above is also NP-Complete. Taking into account that this problem is NP-Complete, a heuristic algorithm of complexity of $O(n^2)$ is introduced.

This algorithm covers general cases while particular cases are considered in [3].

In order to test the efficiency of the algorithm, computational experiments have been carried. We generated randomly input matrices (u_i^j) and (l_i^k) of different sizes which do not include easy solutions. In experiments we study both subproblem and problem DSMWP for small and large matrices. The experiments show the high efficiency of the algorithm.

In this paper, first we formulate our Subproblem and then we give the main facts and definitions to be used later. By transforming subproblem into a Boolean variable problem, we show the equivalance of problem with Dominating Subset of Minimal Weight Problem, which in turn can be formulated as a Multiple-Choice Knapsack Problem. Therefore we conclude that Subproblem (1) and (2) is NP-Complete.

In last section, we give Algorithm A to solve DSMW Problem and results of computational experiments.

2. Formulation of the Problem

Let (l_i^k) be an $(m * n)$ matrix, $m \geqslant n$, with m rows l^k, $k = 1, \ldots, m$, and n columns, $i = 1, \ldots, n$. All elements l_i^k are nonnegative. The first n rows of (l_i^k) matrix form a diagonal matrix, i.e., $l_i^k > 0$, only for $k = i, i = 1, \ldots, n$.

Introduce the function

$$h(x) = \max_k \min_{i \in I(l^k)} l_i^k x_i, \text{ where } I(l^k) = \{i : l_i^k > 0\}.$$

The problem considered in this paper is formulated as follows:

Subproblem

$$\text{Minimize } h(x) \tag{1}$$

subject to

$$x \in S = \left\{ x : \sum_{i=1}^{n} x_i = 1, x_i \geqslant 0, i = 1, \ldots, n \right\}. \tag{2}$$

3. Some Results Concerning Optimal Solutions

The optimal solution for the case $m = n$ is as follows:

THEOREM 1 [3]. *If $m = n$, then Subproblem* (1) *and* (2) *has a unique solution*

$$x_i = h(x)/l_i^i, i = 1, \ldots, n, \text{ where} \tag{3}$$

$$h(x) = \min h(x) = 1 \Big/ \sum_{i=1}^{n} \frac{1}{l_i^i}. \tag{4}$$

COROLLARY 1 [3]. $\min h(x)$ *for $m = n$ is the lower bound of* $\min h(x)$ *for any $m > n$.* If $m > n$, *then two cases are possible.*

Case 1. For each $k > n$, there exists i such that $l_i^k \leqslant l_i^i$.

THEOREM 2 [3]. *If for every $k > n$, there exists i, such that $l_i^k \leqslant l_i^i$, then Subproblem possesses a unique solution, which coincides with the solution for $m = n$.*

Case 2. $\exists K$, such that $l_i^k > l_i^i, \forall i = 1, \ldots, n$, for $\forall k \in K$, i.e. the conditions of Theorem 2 are not satisfied. We will use the following notation:

$$h_k(x) = \min_{i \in I(l^k)} l_i^k x_i, k = 1, 2, \ldots, m, \tag{5}$$

$$h(x) = \max_{k=1,\ldots,m} h_k(x), \tag{6}$$

$$h^* = \min_{x \in S} h(x). \tag{7}$$

Clearly, if x^* is a solution of subproblem (1) and (2), then for each k $(k = 1, 2, \ldots, m)$ there exists i_k such that $h_k(x^*) = l_{i_k}^k x_{i_k}^*$ and for $k \leqslant n$ we have $i_k = k$, i.e. $h_k(x^*) = l_k^k x_k^*$. Let $x \in S$ and for each $i (i = 1, 2, \ldots, n)$ define

$$K_i(x) = \{k \in \{1, \ldots, m\} : h_k(x) = l_{i_k}^k x_{i_k}, i_k = i\}$$

and

$$k_i(x) = \arg \max_{k=1,\ldots,m} \{h_k(x) : k \in K_i(x)\}. \tag{8}$$

Obviously, if for the given i there is no $k > n$ with $h_k(x) = l_i^k x_i$, then $k_i(x) = i$, i.e.

$$h_i(x) = l_i^i x_i. \tag{9}$$

REMARK 1. If for a given x, we call the smallest element of each row (i.e. the element $l_i^k x_i$ which equals to $h_k(x)$) a *chosen element*, then for each column i the row number of the largest chosen element of this column will be $k_i(x)$.

EXAMPLE 1. Let matrix(l_i^k) $(k = 1, 2, \ldots, 8; i = 1, 2, \ldots, 4)$ be as follows:

Matrix (l_i^k).

$k \setminus i$	1	2	3	4
1	1			
2		2		
3			4	
4				5
5	3	5	6	8
6	2	3	5	12
7	5	4	8	10
8	4	6	9	7

Let us take $x_1 = 5/9$, $x_2 = 5/18$, $x_3 = 1/9$, $x_4 = 1/18$, then $\sum_{i=1}^{4} x_i = 1$. Calculating $l_i^k x_i$ $(k = 1, 2, \ldots, 8;\ i = 1,\ 2,\ 3,\ 4)$ and $h_k(x)$ $(k = 1, 2, \ldots, 8)$ by means of formula (5) we get the following matrix (L_i^k)

Matrix (L_i^k).

k	$h_k(x)$	$l_1^k(x_1)$	$l_2^k(x_2)$	$l_3^k(x_4)$	$l_4^k(x_4)$
1	$\frac{5}{9}$	$\frac{5}{9}$			
2	$\frac{10}{18}$		$\frac{10}{18}$		
3	$\frac{4}{9}$			$\frac{4}{9}$	
4	$\frac{5}{18}$				$\frac{5}{18}$
5	$\frac{8}{18}$	$\frac{15}{9}$	$\frac{25}{18}$	$\frac{6}{9}$	$\frac{8}{18}$
6	$\frac{5}{9}$	$\frac{10}{9}$	$\frac{15}{18}$	$\frac{5}{9}$	$\frac{12}{18}$
7	$\frac{10}{18}$	$\frac{25}{9}$	$\frac{20}{18}$	$\frac{8}{9}$	$\frac{10}{18}$
8	$\frac{7}{18}$	$\frac{20}{9}$	$\frac{30}{18}$	$\frac{9}{9}$	$\frac{7}{18}$

We call "a *chosen element*" the smallest element in each row (marked element in Matrix(L_i^k)). If we take the greatest element of the chosen

element in each column (dark element in Matrix(L_i^k)), the row number of this element will be $k_i(x)$ i.e. $k_i(x)$ will be the row number of the greatest chosen element in ith column.

For the matrix above the chosen element in first row will be L_1^1, in second row L_2^2, in third row L_3^3, in fourth row L_4^4, in fifth row L_4^5, in sixth row L_3^6, in seventh row L_4^7 and in eighth row L_4^8.

We have only one chosen element (L_1^1) in the first column, so $k_1(x) = 1$; there is only one chosen element (L_2^2) in the second column, therefore $k_2(x) = 2$. We have two chosen elements (L_3^3 and L_3^6) in third column, since $L_3^6 > L_3^3$, $k_3(x) = 6$. There are four chosen elements in fourth column: L_4^4, L_4^5, L_4^7, L_4^8: The greatest of them is L_4^7, therefore $k_4(x) = 7$.

For $k < n$ (in our example, $k = 1, 2, 3, 4$) there is only one element in each row, so these elements will be chosen ones. It means that there is at least one chosen element in each column. If there are no chosen elements for $k > n$ (in our example, $k = 5, 6, 7, 8$) we will have $k_i(x) = i$, i.e. the formula (9) is true. In the example above this holds for first and second columns, i.e. $k_1(x) = 1$ and $k_2(x) = 2$.

Let us define

$$\overline{l_i(x)} = l_i^{k_i(x)}, i = 1, 2, \ldots, n. \tag{10}$$

THEOREM 3. *If* $x^* = (x_1^*, x_2^*, \ldots, x_n^*)$ *is an optimal solution of the Subproblem* (1) *and* (2)*, then*

$$\overline{l_i(x^*)} x_i^* = h^*, \quad i = 1, 2, \ldots, n. \tag{11}$$

Proof. Let $I = \{1, 2, \ldots, n\}$. It is clear from (6), (8) and (10) that we have

$$\overline{l_i(x^*)} x_i^* \leqslant h^*, \quad i = 1, 2, \ldots, n. \tag{12}$$

Now suppose that Theorem 3 is not true, i.e., there are $i_1, i_2, \ldots, i_t \in I$ such that

$$\overline{l_{i_k}(x^*)} x_{i_k}^* < h^*, \quad k = 1, 2, \ldots, t. \tag{13}$$

Let $I_t = \{i_1, i_2, \ldots, i_t\}$ and $\widetilde{I}_t = I/I_t = \{i_{t+1}, i_{t+2}, \ldots, i_n\}$. Then we obtain from (12) and (13)

$$\overline{l_i(x^*)} x_i^* = h^*, \quad i \in \widetilde{I}_t. \tag{14}$$

Let us consider new variables $\overline{x_i}$, $i \in I$:

$$\overline{x_i} = x_i^* + \alpha_i, \ i \in I_t \tag{15}$$

$$\overline{x_i} = x_i^* - \alpha_i, i \in \widetilde{I}_t, \tag{16}$$

where $\alpha_i > 0$, $i \in I$ and the following conditions are satisfied:

$$\sum_{i \in I} \overline{x_i} = 1, \tag{17}$$

$$\overline{l_{i_1}(x^*)}\overline{x_{i_1}} = \overline{l_{i_2}(x^*)}\overline{x_{i_2}} = \cdots = \overline{l_{i_t}(x^*)}\overline{x_{i_t}} = \overline{l_{i_t+1}(x^*)}\overline{x_{i_t+1}} = \cdots = \overline{l_{i_n}(x^*)}\overline{x_{i_n}} \tag{18}$$

Substituting the values of the variables $\overline{x_i}$ defined by (15) and (16) in (17) and (18) we obtain:

$$\sum_{i\in I_t}(x_i^* + \alpha_i) + \sum_{i\in \widetilde{I_t}}(x_i^* - \alpha_i) = 1,$$

$$\sum_{i\in I} x_i^* + \sum_{i\in I_t}\alpha_i - \sum_{i\in \widetilde{I_t}}\alpha_i = 1.$$

Since $x_i^* \in S$ we have $\sum_{i\in I} x_i^* = 1$ and we obtain:

$$\sum_{i\in I_t}\alpha_i = \sum_{i\in \widetilde{I_t}}\alpha_i. \tag{19}$$

The expression (18) leads to the following system of equations:

$$\left.\begin{array}{c}
\overline{l_{i_1}(x^*)}(x_{i_1}^* + \alpha_{i_1}) = \overline{l_{i_n}(x^*)}(x_{i_n}^* - \alpha_{i_n}),\\
\overline{l_{i_2}(x^*)}(x_{i_2}^* + \alpha_{i_2}) = \overline{l_{i_n}(x^*)}(x_{i_n}^* - \alpha_{i_n}),\\
\cdots\cdots\cdots\cdots\cdots\cdots.\\
\overline{l_{i_t}(x^*)}(x_{i_t}^* + \alpha_{i_t}) = \overline{l_{i_n}(x^*)}(x_{i_n}^* - \alpha_{i_n}),\\
\overline{l_{i_t+1}(x^*)}(x_{i_t+1}^* - \alpha_{i_t+1}) = \overline{l_{i_n}(x^*)}(x_{i_n}^* - \alpha_{i_n}),\\
\cdots\cdots\cdots\cdots\cdots\cdots\\
\overline{l_{i_n-1}(x^*)}(x_{i_n-1}^* - \alpha_{i_n-1}) = \overline{l_{i_n}(x^*)}(x_{i_n}^* - \alpha_{i_n}).
\end{array}\right\} \tag{20}$$

If we add Equation (19) to this system (20) consisting of $(n-1)$ equations we will have n equations for finding n undetermined $\alpha_i > 0$, $i \in I$. Finding α_i's from this system we can calculate new values of $(\overline{x_i}, i \in I)$ by formulas (15) and (16).

Let $I_t \cap I(l^k) = I_t(l^k)$ and $\widetilde{I}_t \cap I(l^k) = \widetilde{I}_t(l^k)$.

Consider $h_k(\bar{x})$, where $\bar{x} = (\overline{x_1}, \overline{x_2}, \ldots, \overline{x_n})$.

$$h_k(\bar{x}) = \min_{i\in I(l^k)} l_i^k\overline{x_i} = \min\{\min_{i\in I_t(l^k)} l_i^k\overline{x_i}, \min_{i\in \widetilde{I_t}(l^k)} l_i^k\overline{x_i}\}. \tag{21}$$

Assume that

$$h_k(x^*) = l_{i_k^*}^k x_{i_k^*}^*, \quad k = 1, 2, \ldots, m. \tag{22}$$

For each k we have two cases:

Case 1. If $i_k^* \in \widetilde{I}_t(l^k)$ then from (22), (10), (16) and (14) we obtain:

$$\min_{i\in\widetilde{I}_t(l^k)} l_i^k\overline{x_i} \leqslant l_{i_k^*}^k\overline{x_{i_k^*}} \leqslant \overline{l_{i_k^*}(x^*)}\overline{x_{i_k^*}} = \overline{l_{i_k^*}(x^*)}x_{i_k^*}^* - \overline{l_{i_k^*}(x^*)}\alpha_{i_k^*} = h^* - \overline{l_{i_k^*}(x^*)}\alpha_{i_k^*} < h^* \tag{23}$$

Case 2. If $i_k^* \in I_t(l^k)$ then from (22), (10), (20), (16) and (14) we obtain:

$$\begin{aligned}\min_{i\in I_t(l^k)} l_i^k\overline{x_i} &\leqslant l_{i_k^*}^k\overline{x_{i_k^*}} \leqslant \overline{l_{i_k^*}(x^*)}\overline{x_{i_k^*}} = \overline{l_{i_n}(x^*)}\overline{x_{i_n}} = \overline{l_{i_n}(x^*)}x_{i_n}^* - \overline{l_{i_n}(x^*)}\alpha_{i_n} \\ &= h^* - \overline{l_{i_n}(x^*)}\alpha_{i_n} < h^*\end{aligned} \tag{24}$$

Now from (21), (23) and (24) we have $h_k(\bar{x}) < h^*$, $k = 1, 2, \ldots, m$. Therefore it follows from (6) and (7) that $h(\bar{x}) = \max_k h_k(\bar{x}) < h^*$ and $\bar{h} < h^*$. But this is a contradiction with optimality of the solution h^* of problem (1) and (2).

This completes the proof of Theorem 3. □

COROLLARY 2. *The optimal solution of Subproblem* (1) *and* (2) *is given as*

$$x_i^* = h^*/\overline{l_i(x^*)}, i = 1, \ldots, n, \text{ where} \tag{25}$$

$$h^* = 1\Big/\sum_{i=1}^{n}\frac{1}{\overline{l_i(x^*)}}. \tag{26}$$

Substituting (10) in (25) and (26) we obtain:

$$x_i^* = h^*/l_i^{k_i(x^*)}, \quad i = 1, \ldots, n, \text{ where} \tag{27}$$

$$h^* = 1\Big/\sum_{i=1}^{n}\frac{1}{l_i^{k_i(x^*)}} \tag{28}$$

4. Transformation of the Subproblem to an Equivalent Problem

Formulas (27) and (28) are obtained from (3) and (4) by substituting $l_i^{k_i(x^*)}$ instead of l_i^i. This means that if the condition of Theorem 2 is not satisfied, the optimal solution will be obtained by substituting some of l_i^i's (those i's for which $h_i(x^*) \neq l_i^i x_i^*$, i.e. for which the condition (11) is not satisfied) by $l_i^{k_i}$.

Now let us see how the function $h = 1/\sum_{i=1}^{n}\frac{1}{l_i^i}$ changes in this substitution. Without loss of generality we can assume that h' is obtained from h by substituting only two elements (say l_1^1 and l_2^2) and remaining left unchanged:

$$h = \frac{1}{\left(\frac{1}{l_1^1}+\frac{1}{l_2^2}\right)+\frac{1}{l_3^3}+\cdots+\frac{1}{l_n^n}}; \quad h' = \frac{1}{\left(\frac{1}{l_1^{k_1}}+\frac{1}{l_2^{k_2}}\right)+\frac{1}{l_3^3}+\cdots+\frac{1}{l_n^n}}.$$

Since the condition of Theorem 2 is not satisfied then $l_1^{k_1} > l_1^1$ and $l_2^{k_2} > l_2^2$. Denoting by $L = \frac{1}{l_1^1} + \frac{1}{l_2^2} + \cdots + \frac{1}{l_n^n}$ and $\frac{1}{l_i^i} - \frac{1}{l_i^{k_i}} = u_i^{k_i}$, we will have $h = \frac{1}{L}$ and

$$h' = \frac{1}{\left(\left(\frac{1}{l_1^{k_1}} - \frac{1}{l_1^1}\right) + \left(\frac{1}{l_2^{k_2}} - \frac{1}{l_2^2}\right)\right) + \left(\frac{1}{l_1^1} + \frac{1}{l_2^2}\right) + \left(\frac{1}{l_3^3} + \cdots + \frac{1}{l_n^n}\right)} = \frac{1}{-u_1^{k_1} - u_2^{k_2} + L},$$

$$h' - h = \frac{1}{L - u_1^{k_1} - u_2^{k_2}} - \frac{1}{L} = \frac{u_1^{k_1} + u_2^{k_2}}{L\left(L - \left(u_1^{k_1} + u_2^{k_2}\right)\right)} = \frac{u_1^{k_1} + u_2^{k_2}}{L^2 - L\left(u_1^{k_1} + u_2^{k_2}\right)}.$$

We see that if $u_1^{k_1} + u_2^{k_2}$ decreases then $(h' - h)$ also decreases. Therefore to obtain the optimal solution we must carry out changes $l_i^{k_i} \to l_i^i$, such that the sum above is minimal, i.e.

$$\sum_i u_i^{k_i} \to \min. \tag{29}$$

We will use the following notation for simplicity:

$$p = m - n, \quad u_i^j = \frac{1}{l_i^i} - \frac{1}{l_i^{j+n}}, \quad i = 1, 2, \ldots, n;\ j = 1, 2, \ldots, p.$$

Clearly u_i^j is the increment of the denominator of the fraction that expresses the function h in the substitution $l_i^{j+n} \to l_i^i$.

Let us define the following function:

$$Sg(x) = \begin{cases} 1, \textit{if } x \geqslant 0, \\ 0, \text{ if } x < 0 \end{cases}$$

and consider variables $x_i^j, i = 1, 2, \ldots, n; j = 1, 2, \ldots, p$:

$$x_i^j = \begin{cases} 1, \text{ if the substitution } l_i^{j+n} \to l_i^j \text{ is accomplished} \\ 0, \text{ otherwise} \end{cases}$$

So Subproblem (1) and (2) is transformed into the following Boolean (0–1) programming problem:

$$\sum_{i=1}^{n} \sum_{j=1}^{p} u_i^j x_i^j \to \min_{x_i^j} \tag{30}$$

$$\sum_{i=1}^{n} x_i^j \leqslant 1, \quad j = 1, 2, \ldots, p, \tag{31}$$

$$\sum_{j=1}^{p} x_i^j \leqslant 1, \quad i = 1, 2, \ldots, n, \tag{32}$$

$$\sum_{i=1}^{n} \sum_{j=1}^{p} x_i^j \geqslant 1, \tag{33}$$

$$\sum_{i=1}^{n} y_i^j \geqslant 1, \quad j = 1, 2, \ldots, p, \tag{34}$$

$$x_i^j = 0 \vee 1, \quad i = 1, 2, \ldots, n;\ j = 1, 2, \ldots, p, \tag{35}$$

$$y_i^j = Sg(\max_{k=1,\ldots,p} \{u_i^k x_i^k\} - u_i^j), \quad i = 1, 2, \ldots, n;\ j = 1, 2, \ldots, p, \tag{36}$$

where condition (30) is obtained from condition (29), conditions (31) and (32) from (5) and (33) from (6). Since the condition of Theorem 2 is not satisfied so condition (9) will not for all i's ($i = 1, 2, \ldots, n$), i.e. at least one substitution $l_i^{j+n} \to l_i^j$ will be accomplished in the optimal solution and this means condition (33). Condition (34) is obtained from (7), (8), (10) and the definition of the variables y_i^j (i.e. from (36)).

Thus we can obtain the optimal solution of Subproblem (1) and (2) by the substitution $l_{i^*}^{j^*}$ by $l_{i^*}^{j^*+n}$ in formulas (3) and (4) for all $x_{i^*}^{j^*} = 1$ in the optimal solution of the problem (30)–(36) and vice versa. In other words the following theorem holds.

THEOREM 4. *Subproblem* (1) *and* (2) *and Problem* (30)–(36) *are equivalent*.

5. Dominating Subset of Minimal Weight Problem

Let us call the problem (30)–(36) "Dominating Subset with Minimal Weight". We can interpret this problem as follows:
Let (u_i^j) be a $(p * n)$ matrix, with p rows, $j = 1, 2, \ldots, p$ and n columns, $i = 1, 2, \ldots, n$ and nonnegative u_i^j for all i, j.

The task is to choose some elements of the matrix such that:

- Each row contains a chosen element, or contains some element which is less than some chosen element located in its column;
- The sum of the chosen elements is minimal.

We can give the following applied interpretation of this problem:

A task consisting of $p(j = 1, 2, \ldots, p)$ operations can be accomplished by $n(i = 1, 2, \ldots, n)$ processors. Suppose that the matrix (u_i^j) gives the time necessary for accomplishment of the task as follows: If

$$u_i^{j_1} \leqslant u_i^{j_2} \leqslant \cdots \leqslant u_i^{j_p} \tag{37}$$

for column i, then $u_i^{j_1}$ is the time (or cost) for accomplishment of operation j_1 by processor i; $u_i^{j_2}$ is the time for the accomplishment of operations j_1 and j_2 by processor i, and so on. At last $u_i^{j_p}$ is the time for the accomplishment

of all operations $(j_1, j_2, \ldots, j_p)$ by processor i. The problem is to distribute operations among the processors minimizing the total time (or the total cost) required for the accomplishment of all tasks.

6. Complexity of the Subproblem

Now we transform problem (30)–(36) into an equivalent Multiple-Choice Knapsack problem with $p * n = q$ binary variables and p constraints.

Coefficients of the objective function of this problem are defined as

$$c_1 = u_1^1,\ c_2 = u_1^2, \ldots,\ c_p = u_1^p,\ c_{p+1} = u_2^1,\ c_{p+2} = u_2^2, \ldots,\ c_{2p} = u_2^p,$$

$$c_{2p+1} = u_3^1, \ldots,\ c_q = u_n^p.$$

Consider $1 \leqslant k \leqslant q$. Suppose $c_{k=u_i^j}$ for some i, j, i.e., c_k equals to some element in ith column and jth row of matrix (u_i^j) and for ith column of this matrix the condition (37) above is satisfied. Assume that c_k is in sth place in row (37) i.e. $c_k = u_i^{j_s}$ Then $a_k^{j_1} = a_k^{j_2} = \cdots = a_k^{j_s} = 1$ and $a_k^{j_s+1} = a_k^{j_s+2} = \cdots = a_k^{j_p} = 0$.

We obtain the following problem:

$$\sum_{i=1}^{q} c_i z_i \to \min \tag{38}$$

$$\sum_{i=1}^{q} a_i^j z_i \geqslant 1, \quad j = 1, 2, \ldots, p \tag{39}$$

$$z_i = 0 \vee 1, \quad i = 1, 2, \ldots, q. \tag{40}$$

To explain this transformation let us consider the following example:

EXAMPLE 2. Let matrix (u_i^j) be as follows:

$$(u_i^j) = \begin{pmatrix} 2 & 4 & 9 \\ 8 & 12 & 3 \\ 10 & 6 & 5 \end{pmatrix}.$$

Then $c = (2, 8, 10, 4, 12, 6, 9, 3, 5)$ and matrix (a_i^j) is:

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 \end{pmatrix}$$

Problem (38)–(40) is a Multiple-Choice Knapsack problem and since it is NP-Complete [7, 9] then problem (30)–(36) is also NP-Complete. So the following Theorem and Corollary hold:

THEOREM 5. *The problem* (30)–(36) *is NP-Complete*.

COROLLARY 3. *Subproblem* (1) *and* (2) *is NP-Complete*.

7. A Solution of the Dominating Subset of Minimal Weight Problem

Let u^j be chosen as follows for each row j;

$$u^j = \min_{i=1,\ldots,n} \{u_i^j\}, j = 1, 2, \ldots, p \text{ and } i_j \text{ as;}$$

$$i_j = \arg \min_{i=1,\ldots,n} \{u_i^j\}, j = 1, 2, \ldots, p.$$

It is obvious that $u^j = u_{i_j}^j$. Let us call $u_{i_j}^j$ the *critical element* for row j, and determine $\bar{u}_i$ for column i;

$$\bar{u}_i = \begin{cases} 0, \text{ if no critical element exists in column } i \\ \max_{j=1,\ldots,p}\{u_i^j : u_i^j = u^j\}, \text{otherwise} \end{cases}$$

Suppose that there is i such that $\bar{u}_i \neq 0$.

Then $\bar{u}_i = \max_{j=1,\ldots,p}\{u_i^j : u_i^j = u^j\} = \max_{j=1,\ldots,p}\{u_{i_j}^j : i_j = i\} = u_i^{j_i}$, and

$$j_i = \arg \max_{j=1,\ldots,p} \{u_{i_j}^j : i_j = i\} \quad .$$

We call $u_i^{j_i}$ the *dominant critical element* of column i.

For any two elements $u_{i_1}^{j_1}$ and $u_{i_2}^{j_2}$ if element u_m^n ($u_m^n = u_{i_2}^{j_1}$ or $u_m^n = u_{i_1}^{j_2}$) at the crossing of the row of the first element and the column of the second (or the row of the second and the column of the first) is greater than any of them but not their sum, then u_m^n is called the *common dominant* of $u_{i_1}^{j_1}$ and $u_{i_2}^{j_2}$. If both elements $u_{i_2}^{j_1}$ and $u_{i_1}^{j_2}$ are common dominants then the smaller one is called the *smaller common dominant* and the greater one is called the *greater common dominant*.

If for the element u_i^j, there exist the rows $j_1^i, j_2^i, \ldots, j_{t_i}^i$, such that $u_i^j \geqslant u_i^{j_1^i}, u_i^j \geqslant u_i^{j_2^i}, \ldots, u_i^j \geqslant u_i^{j_{t_i}^i}$, then u_i^j is called the *dominant of elements* $u_i^{j_1^i}, u_i^{j_2^i}, \ldots, u_i^{j_{t_i}^i}$ and row j is called the *dominant of rows* $j_1^i, j_2^i, \ldots, j_{t_i}^i$ for column i.

For each critical element u_i^j let us construct the structure as $u_i^j, j, j_1^i, j_2^i, \ldots, j_{t_i}^i$.

In this structure j indicates the row in which element u_i^j is placed and numbers $j_1^i, j_2^i, \ldots, j_{t_i}^i$ indicate the rows on which the row j is dominant for the column i.

7.1. A GENERAL SOLUTION ALGORITHM A FOR THE PROBLEM

Taking in account the properties of DSMW problem, we give the following heuristic algorithm to solve it.

ALGORITHM A.

A1. Find $u_i = \max_{j=1,\ldots,p}\{u_i^j\}$ and $j_i = \arg\max_{j=1,\ldots,p}\{u_i^j\}, i = 1, 2, \ldots, n$. It is clear that $u_i = u_i^{j_i}$.

A2. Find $U^* = \min_{i=1,\ldots,n}\{u_i\}$, $i_{j_i} = \arg\min_{i=1,\ldots n}\{u_i\}$. It is true that $U^* = u_{i_{j_i}}^{j_i}$.
Set the solution matrix X^* such as each decision variable x_i^j is 0 but $x_{i_{j_i}}^{j_i}$ is 1.

A3. If $U^* = \min_{i=1,\ldots,n}\{u_i^{j_i}\}$ (that is, if U^* is the minimal element of the row where it is placed.), then the optimal solution is found, go to A15.

A4. Determine critical elements for each row; $u^j = u_{i_j}^j$, $j = 1, 2, \ldots, p$.

A5. Find the dominant critical elements for each column; $\bar{u}_i$, $i = 1, 2, \ldots, n$. If no critical element exists in column i, then $\bar{u}_i$ is assumed to be 0, and is not taken into account in further steps.

A6. Sort the critical elements in descending order provided that the dominant ones are at the head.

$$\bar{u}_{i_1} \geqslant \bar{u}_{i_2} \geqslant \cdots \bar{u}_{i_q} \geqslant u^{j_1} \geqslant u^{j_2} \geqslant \cdots \geqslant u^{j_p} \tag{41}$$

In (41), leftmost q elements are dominant critical elements, $q \leqslant n$.

A7. Find a solution according to the current order using the following Algorithm B. Set the initial values $U = 0$, $X = 0$ and $N = \{1, 2, \ldots, p\}$ prior to the execution of Algorithm B.

A8. For the newly obtained solution, if the value of the objective function $U < U^*$ then set U^* and X^* according to the new solution.

A9. Perform the steps A10–A14 for $i = 2, 3, \ldots, n$.

A10. Perform the steps A11–A14 for $j = 1, 2, \ldots, i-1$.

A11. If no dominant element exists for $\bar{u}_i$ and $\bar{u}_j$ then go to A10.

A12. Let u_i^j be the dominant of $\bar{u}_i$ and $\bar{u}_j$. Insert u_i^j after the dominant elements in (41).

A13. Set the initial values $U = u_i^j, x_i^j = 1$ and $N = \{1..p\}\backslash\{j, j_1^i, j_2^i, \ldots, j_{t_i}^i\}$, apply Algorithm B.

A14. For the newly obtained solution, if the value of the objective function $U < U^*$ then Set U^* and X^* according to the new solution.

A15. End the algorithm with an output of U^*, X^*.

The above Algorithm A is a heuristic algorithm which produces a feasible solution, satisfying all constraints. In this algorithm firstly the largest element of each column is found and the smallest one is taken as a primary solution (steps A1 and A2).

If this solution satisfies the optimization condition (Proposition 3 [3]) then algorithm ends and declares that the solution is found (step A3).

Then for each row the critical elements are found and for each column the dominant critical element is determined, and they are taken in a nonincreasing (decreasing) sequence (41) (step A4–A6).

To find a feasible solution in each iteration of algorithm A, greedy algorithm B, which takes in an account the sequence (41), is used. At the beginning of iteration a solution corresponding to the sequence (41) is found (step A7) and then the ordered critical elements are compared pairwise trying to find a dominant element for them (steps A9–A14). These elements are located in the sequence (41) after the elements for which they are dominants (step A12). Those elements are assumed as to be chosen in the input of algorithm B (step A13). Each solution found in this way is compared with the best solution found earlier and the best solution among them is chosen (step A8).

Note that the number of steps of the algorithm A is bounded by the number of columns (n) of the matrix (u_i^j).

7.2. PARTICULAR SOLUTION ALGORITHM B

The algorithm B is greedy and uses the sequence (41). In each iteration of this algorithm a next element from the sequence (41) is chosen and if this element is removed in the set N, it is added to the set U and all row numbers for which this element is dominant. Here U is the set of row numbers of chosen elements and N is the set of row numbers of removed elements. The number of steps of this algorithm is bounded by the number n of the matrix (u_i^j).

ALGORITHM B.

Initial values of U, N, and X are set in the caller step in Algorithm A prior to execution of Algorithm B.

B1. $k \leftarrow 1$
B2. If $j^{i_k} \notin N$ then go to B7
B3. $U \leftarrow U + \bar{u}_{i_k}$
B4. $x_{i_k}^{j^{i_k}} \leftarrow 1$
B5. $N \leftarrow N \backslash \{j^{i_k}, j_1^{i_k}, j_2^{i_k}, \ldots, j_{t_k}^{i_k}\}$
B6. If $N = \varnothing$ then go to B9
B7. $k \leftarrow k + 1$
B8. If $k \leqslant n$ then go to B2.
B9. End the algorithm with an output of U, X.

7.3. COMPUTATIONAL EXPERIMENTS

In order to test its efficiency, the algorithm is applied to a number of inputs.

The codes have been written in ANSI-C. Numerical experiments have been carried out on an IBM Pentium-S CPU 166 MHz.

To provide the average case, randomly generated input matrices of different sizes are used. For the purpose of imitating inputs of general nature, the matrices are generated under the conditions stated below:

(a) $0 \leqslant u_i^j \leqslant 100$ and the u_i^j elements are not integers;
(b) None of the rows majors any other row, i.e., there are no rows j_1, j_1 that for every i, $l_i^{j_1} \geqslant l_i^{j_2}$.
(c) For any value of n, 10 input matrices are generated.

The tests are done in two stages. In the first stage, small values are chosen for n, and the solutions are found by both the described algorithm and Branch & Bound algorithm [8] in order to consider how close is the solution found by the algorithm to the optimum solution. Relative errors are estimated by ratio $\alpha = \frac{U-U^*}{U^*}$. In this stage values are chosen as $n = 5$, 6, 8, 10 and $p = 10$, 12, 16, 15, respectively.

In the second stage, the purpose is to evaluate the time required by the algorithm for solution of problems with larger scale input matrices. So the values are chosen as $p = 1000$ and $n = 5, 10, 15, \ldots, 50$. The results are presented in the following tables.

As it is seen from Table 1, the presented Algorithm A gives the optimal solutions for most of the inputs and very close results for the rest. In Table 2, the time required for solution of large scale inputs are given showing that the algorithm is fast even for large scale input matrices.

The computational experiment above was carried out for Subproblems (1) and (2) as well (i.e. the matrix (l_i^k) was taken as input data, a matrix (u_i^j) was obtained and the problem was solved). It is interesting that in all these experiments we have obtained optimal solutions (we have taken $n = 5$, 10 and $m = 2n, 3n, 4n$).

The experiments with large scale matrices for Subproblems (1) and (2) were carried out. In this experiment, the elements of matrix of problem are randomly produced according (a)–(c) conditions above. We have considered values $m = 100, 200, \ldots, 1000$ and $n = 20$, 30, 40, 50, 100. Solution for each of m and n is given in Table 3.

7.4. ON THE COMPLEXITY OF THE ALGORITHM

The algorithm costs most at steps A9–A14 which are of $O\left(\frac{n(n+1)}{2}\right) = O(n^2)$ in total, hence the time complexity of the algorithm is $O(n^2)$. Adding the dominant elements obtained in step A11 to the sequence (9) increases the time elapsed, and it is not necessary always to pursue this task, because it has been revealed during the tests that the solution produced by the algo-

Table 1. Results for small input matrices

No	$n = 5, p = 10$			$n = 6, p = 12$			$n = 8, p = 16$			$n = 10, p = 15$		
	The value of the Objective function		α	The value of the Objective function		α	The value of the Objective function		α	The value of the Objective function		α
	B&B	Heuristic		B&B	Heuristic		B&B	Heuristic		B&B	Heuristic	
1	40.09	40.09	0.00	61.94	64.84	0.05	83.86	83.86	0.00	68.01	68.01	0.00
2	86.61	86.61	0.00	85.56	86.11	0.01	70.88	70.88	0.00	75.75	80.73	0.07
3	65.70	65.70	0.00	76.38	76.38	0.00	87.03	88.05	0.01	72.40	72.40	0.00
4	58.07	62.36	0.07	56.54	58.50	0.03	79.33	79.33	0.00	53.03	53.03	0.00
5	43.54	43.54	0.00	80.18	80.18	0.00	64.31	64.31	0.00	85.13	88.36	0.04
6	82.28	87.92	0.07	78.82	78.82	0.00	74.43	78.87	0.06	35.27	37.97	0.08
7	62.04	62.04	0.00	71.36	74.85	0.05	61.54	61.54	0.00	74.59	74.59	0.00
8	55.01	55.01	0.00	82.58	82.58	0.00	72.73	74.96	0.03	64.70	65.94	0.02
9	53.59	53.59	0.00	65.72	65.72	0.00	58.38	58.38	0.00	64.89	64.89	0.00
10	87.99	87.99	0.00	68.43	68.43	0.00	79.71	80.00	0.00	80.79	81.54	0.01

Table 2. Results for large scale input matrices

No	n	p	Time (s)
1	5	1000	0.05
2	10	1000	0.05
3	15	1000	0.16
4	20	1000	0.16
5	25	1000	0.16
6	30	1000	0.22
7	35	1000	0.22
8	40	1000	0.27
9	45	1000	0.27
10	50	1000	0.27

Table 3. Results for large scale input matrices for subproblem

No	$m\backslash n$	20	30	40	50	100
1	100	0.00	0.00	0.00	0.05	-
2	200	0.00	0.05	0.05	0.05	0.05
3	300	0.05	0.05	0.05	0.05	0.16
4	400	0.05	0.05	0.05	0.05	0.27
5	500	0.05	0.05	0.11	0.11	0.33
6	600	0.11	0.11	0.11	0.11	0.38
7	700	0.11	0.11	0.11	0.16	0.44
8	800	0.16	0.11	0.16	0.22	0.44
9	900	0.16	0.22	0.16	0.22	0.55
10	1000	0.16	0.22	0.27	0.33	0.66

rithm is one of the solutions obtained in the early iterations of the algorithm. So, in these tests the limit for the number of iterations is chosen as n^2, but it will make sense if for extremely large inputs the limit for the number of iterations is chosen smaller to decrease the total time elapsed.

REMARK 2. In the algorithm given in paper [3], the number of iterations will decrease at a rate of $n(n-1)$ provided that the changes below are done in the algorithm:

Step 2. $j = \arg\min_i \left(\frac{1}{l_i^i} - \frac{1}{l_i^{q(i)}} \right)$

Step 3. If $\left(\frac{1}{l_j^j} - \frac{1}{l_j^{q(j)}} \right) = \min_i \left(\frac{1}{l_i^i} - \frac{1}{l_i^{q(j)}} \right)$ then let $l_j^j = l_j^{q(j)}$ and

8. Conclusion

In this paper a new heuristic algorithm for solving auxiliary problem in the Cutting Angle Method (CAM) of global optimization has been proposed and studied. The auxiliary problem is solved at each iteration of CAM and

the efficiency of CAM strongly depends on an algorithm for solving auxiliary problem. It is known that the latter problem is NP-hard.

The time complexity of the proposed algorithm is $O(n^2)$. If the conditions in Algorithm 1 from [3] (Proposition 3) are satisfied then this algorithm will be finished at the global solution. The new algorithm can be applied to more general situations than Algorithms 1 and 2 proposed and studied in [3]. The algorithm proposed in this paper can be considered as a generalization of Algorithms 1 and 2 from [3]. It should be noted that this algorithm requires more memory than Algorithms 1 and 2. A number of numerical experiments have been carried out using the proposed algorithm. The results of these experiments show its effectiveness. In numerical experiments, the CPU time was less than 0.01 s, when the conditions of [3] (Proposition 3) are satisfied, regardless dimensions of the matrix under consideration.

The study of new versions of CAM using the proposed algorithm for solving auxiliary problem is the subject of our future investigations.

9. Acknowledgements

The author would like to express his gratitude to Dj. A. Babayev for bringing his attention to this problem and for very useful discussions. Special thanks are due to the referees for their invaluable comments and suggestions.

References

1. Andramonov, M.Yu., Rubinov, A.M. and Glover B.M. (1997), Cutting angle methods for minimizing increasing convex along rays functions. *Research Report* 97/7, SITMS, University of Ballarat.
2. Andramonov, M.Yu., Rubinov, A.M. and Glover, B.M. (1999), Cutting angle methods in Global Optimization. *Applied Mathematics Letters* 12, 95–100.
3. Babayev, Dj.A. (2000), An exact method for solving the subproblem of the cutting angle method of global optimization. In: *Optimization and Related Topics*, *in Kluwer Academic Publishers*, *ser*. '*Applied Optimization*', Dordrecht, Vol. 47, pp. 15–26.
4. Bagirov, A.M. and Rubinov, A.M. (2000), Global minimization of increasing positively homogeneous functions over the unit simplex. *Annals of Operation Research* 98, 171–187.
5. Bagirov, A.M. and Rubinov, A.M. (2001), Modified versions of the cutting angle method. In: Hadjisavvas, N. and Pardalos, P.M. (eds.), *Nonconvex Optimization and Its Applications*, Vol. 54, *Advances in Convex Analysis and Global Optimization*, Kluwer Academic Publishers, Dordrecht.
6. Bagirov, A.M. and Rubinov, A.M. (2003), Cutting angle method and a local search, *Journal of Global Optimization* 27, 193–213.
7. Garey, R.M. and Johnson, D.S. (1979), *Computers and Intractability*: *A Guide to the Theory of NP-Completeness*, W.H. Freeman and Company, New York, NY.

8. Horst, R., Pardalos, P.M. and Thoai, N.V. (2000), *Introduction to Global Optimization, Nonconvex Optimization and its Applications*, Vol. 48, Kluwer Academic Publishers, Dordrecht.
9. Martello, S. and Toth, P. (1990), *Knapsack Problems: Algorithms and Computer Implementations*, John Wiley and Sons, Chichester.
10. Rubinov, A.M. (2000), *Abstract Convexity and Global Optimization*, Kluwer Academic Publishers, Dordrecht.
11. Rubinov, A.M. and Andramonov, M.Yu. (1999), Minimizing increasing star-shaped functions based on abstract convexity. *Journal of Global Optimization* 15, 19–39.
12. Rubinov, A.M. and Andramonov, M.Yu. (1999), Lipshitz programming via increasing convex along rays functions. *Optimization Methods and Software* 10, 763–781.