An efficient method for minimizing a strictly convex separable function subject to convex separable inequality constraint and box constraints

Stefan M. Stefanov

Department of Informatics, Neofit Rilski South-Western University, 2700 Blagoevgrad, BULGARIA, e-mail: stefm@aix.swu.bg

Abstract

A minimization problem with strictly convex separable objective function subject to a convex separable inequality constraint of the form "less than or equal to" and bounds on the variables is considered. Necessary and sufficient condition is proved for a feasible solution to be an optimal solution to this problem. An iterative algorithm of polynomial complexity for solving such problems is suggested and its convergence is proved. Modifications of this algorithm are proposed in connection with some extensions of the considered problem as well as in order to avoid some computational difficulties. Examples of important convex functions for the problem under consideration and computational results are presented.

Key words. Convex programmimg, separable programming, singly constrained program, algorithms, computational complexity.

2000 AMS Subject Classification. 90C25, 65K05, 49M37.

1 Introduction

Consider the convex program
(C)

$$\min\ c(\mathbf{x}) = \sum_{j\in J} c_j(x_j) \tag{1}$$

subject to

$$\sum_{j\in J} d_j(x_j) \le \alpha \tag{2}$$

$$a_j \le x_j \le b_j, \quad j \in J, \tag{3}$$

where $c_j(x_j)$ are twice differentiable strictly convex functions, and $d_j(x_j)$ are twice differentiable convex functions, defined on the open convex sets $X_j, j \in J$, respectively, $d'_j(x_j) > 0$ for every $j \in J$, $\mathbf{x} = (x_j)_{j\in J}$, and $J \equiv \{1, ..., n\}$.

The feasible region X (2) – (3) is a convex set because it is an intersection of the convex set (2) and the box (3) of dimension $|J| = n$.

Since the derivative of a convex differentiable function is a monotone nondecreasing function, then condition $d'_j(x_j) > 0$ with $a_j \leq x_j \leq b_j$, $j \in J$, is satisfied if and only if $d'_j(a_j) > 0$, when $a_j > -\infty$, or if and only if $\lim_{t\to-\infty} d'_j(t) > 0$, when $a_j = -\infty$. Set $d'_j(b_j) = \lim_{t\to\infty} d'_j(t)$ when $b_j = +\infty$.

Problem (C) and related to it arise in many cases, for example, in production planning and scheduling [3], in allocation of resources [3, 37], in allocation of effort resources among competing activities [17], in the theory of search [7], in subgradient optimization [13], in facility location problems [25, 26], in the implementation of projection methods when the feasible region is of the form (2) – (3) [25, 26, 29], etc. That is why we need efficient algorithms for solving such problems.

Problems like (C) are subject of intensive study. Related problems and methods for solving them are considered in [1] – [37], etc. The solution of knapsack problems with arbitrary convex or concave objective functions is studied in [3, 17, 21, 37], etc. Quadratic knapsack problems and related to them are studied in [5, 23, 24], etc. An indefinite version of these problems is considered in [35], and algorithms for the case of convex quadratic objective function were proposed in [5, 10, 14], etc. Algorithms for bound constrained quadratic programming problems are proposed in [9, 20, 22]. Algorithms for the least distance problem are suggested, for example, in [1, 36]. A polynomial time algorithm for the resource allocation problem with a convex objective function and nonnegative integer variables is suggested in [16]. Algorithms for solving a quadratic program with (strictly) convex objective function of the type $c(\mathbf{x}) = \frac{1}{2}\sum_{j\in J}(x_j - y_j)^2$ and constraints of the form (2) – (3) are suggested in [25, 28]. This problem itself is equivalent to projecting a point $\mathbf{y} = (y_1, \ldots, y_n)$ onto the convex set (2) – (3) and has always a unique solution when the feasible region is nonempty. Such algorithms are very useful for methods using projection onto region of the considered type, for example, gradient projection methods, methods of projection stochastic quasigradients [25], etc. Algorithms for finding a projection are proposed, e.g., in [19, 25, 29], etc., and projected Newton-type methods are suggested in [2, 12].

This paper is devoted to development of new efficient algorithm for solving Problem (C). The paper is organized as follows. In Section 2, a necessary and sufficient condition (characterization theorem) for a feasible solution to be an optimal solution to Problem (C) is proved (Theorem 1). In Section 3, an iterative algorithm of polynomial complexity for solving Problem (C), based on Theorem 1, is proposed, and convergence of this algorithm is proved as Theorem 2. The approach suggested in this paper can be extended to the case when $d'_j(a_j) = 0$ and/or $d'_j(b_j) = 0$, or $d'_j(x_j) \equiv 0$ ($d'_j(x_j) \equiv d_j = 0$ in the linear case) for some indices j. These topics as well as computational aspects of implementation of the algorithm are considered in Section 4. Section 5 contains examples of some strictly convex functions $c_j(x_j)$ and convex functions $d_j(x_j)$ which are involved in Problem (C), and some computational results.

This paper is a continuation and generalization of the approach suggested in author's previously published papers on the topic.

2 Main result. Characterization theorems

Suppose that the following assumptions are satisfied.

(I) $a_j \leq b_j$ for all $j \in J$. If $a_k = b_k$ for some $k \in J$, then the value $x_k := a_k = b_k$ is determined *a priori.*

(II) $\sum_{j\in J} d_j(a_j) \leq \alpha$. Otherwise the constraints (2) and (3) are inconsistent and $X = \emptyset$. In addition to this assumption, we suppose that $\alpha \leq \sum_{j\in J} d_j(b_j)$ in some cases which are specified below.

(III) (Slater's constraint qualification) There exists a point $\overline{\mathbf{x}} = (\overline{x}_1, \ldots, \overline{x}_n) \in X$ such that $\sum_{j\in J} d_j(\overline{x}_j) < \alpha$.

Let $h_j^{\leq}, j \in J$, be the value of x_j for which $c_j'(x_j) = 0$. If a finite value with this property does not exist, since $c_j'(x_j)$ is a monotone increasing function ($c_j(x_j)$ is strictly convex), we adopt $h_j^{\leq} = -\infty$.

The Lagrangian for Problem (C) is

$$L(\mathbf{x}, \mathbf{u}, \mathbf{v}, \lambda) = \sum_{j\in J} c_j(x_j) + \lambda \left(\sum_{j\in J} d_j(x_j) - \alpha \right) + \sum_{j\in J} u_j(a_j - x_j) + \sum_{j\in J} v_j(x_j - b_j), \quad (4)$$

where $\lambda \in \mathbb{R}^1_+; \mathbf{u}, \mathbf{v} \in \mathbb{R}^n_+$, and $\mathbb{R}^n_+$ consists of all vectors with n real nonnegative components.

The Karush-Kuhn-Tucker (KKT) conditions for the minimum solution $\mathbf{x}^* = (x_j^*)_{j\in J}$ to Problem (C) are

$$c_j'(x_j^*) + \lambda d_j'(x_j^*) - u_j + v_j = 0, \quad j \in J, \quad (5)$$

$$u_j(a_j - x_j^*) = 0, \quad j \in J, \quad (6)$$

$$v_j(x_j^* - b_j) = 0, \quad j \in J, \quad (7)$$

$$\lambda \left(\sum_{j\in J} d_j(x_j^*) - \alpha \right) = 0, \quad \lambda \in \mathbb{R}^1_+, \quad (8)$$

$$\sum_{j\in J} d_j(x_j^*) \leq \alpha, \quad (9) \equiv (2)$$

$$a_j \leq x_j^* \leq b_j,\ j \in J, \quad (10) \equiv (3)$$

$$u_j \in \mathbb{R}^1_+,\ v_j \in \mathbb{R}^1_+, \quad j \in J, \quad (11)$$

where $\lambda, u_j, v_j, j \in J$, are the Lagrange multipliers associated with the constraints (2), $a_j \leq x_j$, $x_j \leq b_j, j \in J$, respectively. If $a_j = -\infty$ or $b_j = +\infty$ for some $j \in J$, we do not consider the corresponding condition (6) ((7), respectively) and multiplier u_j (v_j, respectively).

Since $c_j(x_j), j \in J$, are strictly convex differentiable functions and $d_j(x_j), j \in J$, are convex differentiable functions in one variable, then $c_j'(x_j)$ and $d_j'(x_j)$ satisfy

$$[c_j'(x_j^1) - c_j'(x_j^2)](x_j^1 - x_j^2) > 0 \quad \forall\, x_j^1 \neq x_j^2, j \in J, \quad (12)$$

$$[d_j'(x_j^1) - d_j'(x_j^2)](x_j^1 - x_j^2) \geq 0 \quad \forall\, x_j^1, x_j^2, j \in J, \quad (13)$$

respectively.

Since $\lambda \geq 0, u_j \geq 0,\ v_j \geq 0, j \in J$, and since the complementary conditions (6), (7), (8) must be satisfied, in order to find x_j^*, $j \in J$, from system (5) – (11), we have to consider all possible cases for λ, u_j, v_j: all λ, u_j, v_j equal to 0; all λ, u_j, v_j different from 0;

some of them equal to 0 and some of them different from 0. The number of these cases is $2^{|J|+1} = 2^{2n+1}$, where $2n+1$ is the number of all $\lambda, u_j, v_j, j \in J$, and $|J| = n$. Obviously this is an enormous number of cases, especially for large-scale problems. For example, when $n = 1500$, we have to consider $2^{3001} \approx 10^{900}$ cases. (Only for comparison, recall that astrophysicists have determined that the age of the Universe is approximately 15 billion years, that is, approximately "only" 10^{18} seconds.) Moreover, in each case, we have to solve a large-scale system of (nonlinear) equations in $x_j^*, \lambda, u_j, v_j, j \in J$. Therefore, the *direct* application of the KKT theorem, using explicit enumeration of all possible cases, for solving large-scale problems of the considered form would not give a result and we need efficient methods to solve the problem under consideration.

The following Theorem 1 gives necessary and sufficient condition (characterization) of the optimal solution to Problem (C). Its proof is based on the KKT theorem. As we will see in Section 5, by using Theorem 1 we can solve Problem (C) with $n = 1500$ variables for about 0.001 seconds on a personal computer.

Theorem 1 (Characterization of the optimal solution to Problem (C)) *A feasible solution* $\mathbf{x}^* = (x_j^*)_{j \in J} \in X$ *is an optimal solution to Problem* (C) *if and only if there exists a* $\lambda \in \mathbb{R}^1_+$ *such that*

$$x_j^* = a_j, \quad j \in J_a^\lambda \overset{\text{def}}{=} \left\{ j \in J : \lambda \geq -\frac{c_j'(a_j)}{d_j'(a_j)} \right\} \tag{14}$$

$$x_j^* = b_j, \quad j \in J_b^\lambda \overset{\text{def}}{=} \left\{ j \in J : \lambda \leq -\frac{c_j'(b_j)}{d_j'(b_j)} \right\} \tag{15}$$

$$x_j^* : \lambda d_j'(x_j^*) = -c_j'(x_j^*), \quad j \in J^\lambda \overset{\text{def}}{=} \left\{ j \in J : -\frac{c_j'(b_j)}{d_j'(b_j)} < \lambda < -\frac{c_j'(a_j)}{d_j'(a_j)} \right\}. \tag{16}$$

Proof. *Necessity.* Let $\mathbf{x}^* = (x_j^*)_{j \in J}$ be an optimal solution to Problem (C). Then there exist constants $\lambda, u_j, v_j, j \in J$, such that KKT conditions (5) – (11) are satisfied. Consider both possible cases for λ.

1) Let $\lambda > 0$. Then system (5) – (11) becomes (5), (6), (7), (10), (11) and

$$\sum_{j \in J} d_j(x_j^*) = \alpha, \tag{17}$$

that is, the inequality constraint (2) is satisfied as an equality for $x_j^*, j \in J$, in this case.

a) If $x_j^* = a_j$, then $u_j \geq 0, v_j = 0$ according to (7) and (11). Therefore, (5) implies $c_j'(x_j^*) = u_j - \lambda d_j'(x_j^*) \geq -\lambda d_j'(x_j^*)$. Using that $d_j'(x_j^*) > 0$, we get

$$\lambda \geq -\frac{c_j'(x_j^*)}{d_j'(x_j^*)} \equiv -\frac{c_j'(a_j)}{d_j'(a_j)}.$$

b) If $x_j^* = b_j$, then $u_j = 0, v_j \geq 0$ according to (6) and (11). Therefore, (5) implies $c_j'(x_j^*) = -v_j - \lambda d_j'(x_j^*) \leq -\lambda d_j'(x_j^*)$. Hence

$$\lambda \leq -\frac{c_j'(x_j^*)}{d_j'(x_j^*)} \equiv -\frac{c_j'(b_j)}{d_j'(b_j)}.$$

c) If $a_j < x_j^* < b_j$, then $u_j = v_j = 0$ according to (6) and (7). Therefore, (5) implies $-c_j'(x_j^*) = \lambda d_j'(x_j^*)$. Since $d_j'(x_j^*) > 0, j \in J$, $\lambda > 0$ by the assumptions, then $-c_j'(x_j^*) > 0$. Using that $b_j > x_j^* > a_j$, from (12), (13) it follows that $c_j'(b_j) > c_j'(x_j^*) > c_j'(a_j)$, $d_j'(b_j) \geq d_j'(x_j^*) \geq d_j'(a_j), j \in J$. Using that $-c_j'(x_j^*) > 0$, $d_j'(x_j^*) > 0$, $\lambda = -\frac{c_j'(x_j^*)}{d_j'(x_j^*)}$, $\frac{1}{d_j'(x_j)}$ is a monotone nonincreasing function as a reciprocal of the derivative of the convex function $d_j(x_j)$ with $d_j'(x_j) > 0$, we obtain

$$\lambda = -\frac{c_j'(x_j^*)}{d_j'(x_j^*)} \leq -\frac{c_j'(x_j^*)}{d_j'(a_j)} < -\frac{c_j'(a_j)}{d_j'(a_j)}, \quad \lambda = -\frac{c_j'(x_j^*)}{d_j'(x_j^*)} \geq -\frac{c_j'(x_j^*)}{d_j'(b_j)} > -\frac{c_j'(b_j)}{d_j'(b_j)},$$

that is,

$$-\frac{c_j'(b_j)}{d_j'(b_j)} < \lambda < -\frac{c_j'(a_j)}{d_j'(a_j)}. \tag{18}$$

2) Let $\lambda = 0$. Then system (5) – (11) becomes: $c_j'(x_j^*) - u_j + v_j = 0$, $j \in J$, and (6), (7), (9), (10), (11).

a) If $x_j^* = a_j$, then $u_j \geq 0, v_j = 0$. Therefore $c_j'(a_j) \equiv c_j'(x_j^*) = u_j \geq 0$. Multiplying both sides of this inequality by $-\frac{1}{d_j'(a_j)}$ (< 0 by the assumption), we obtain

$$-\frac{c_j'(a_j)}{d_j'(a_j)} \leq 0 \equiv \lambda.$$

b) If $x_j^* = b_j$, then $u_j = 0, v_j \geq 0$. Therefore $c_j'(b_j) \equiv c_j'(x_j^*) = -v_j \leq 0$. Multiplying this inequality by $-\frac{1}{d_j'(b_j)} < 0$, we get

$$-\frac{c_j'(b_j)}{d_j'(b_j)} \geq 0 \equiv \lambda.$$

c) If $a_j < x_j^* < b_j$, then $u_j = v_j = 0$. Therefore $c_j'(x_j^*) = 0$, that is, $x_j^* = h_j^{\leq}$. Since $b_j > x_j^* > a_j$, $j \in J$, by the assumption, from (12) it follows that $c_j'(b_j) > c_j'(x_j^*) = 0$, $0 = c_j'(x_j^*) > c_j'(a_j)$. Multiplying the first inequality by $-\frac{1}{d_j'(b_j)} < 0$ and the second inequality by $-\frac{1}{d_j'(a_j)} < 0$, we obtain $-\frac{c_j'(b_j)}{d_j'(b_j)} < 0 \equiv \lambda$, $\quad \lambda \equiv 0 < -\frac{c_j'(a_j)}{d_j'(a_j)}$, that is,

$$-\frac{c_j'(b_j)}{d_j'(b_j)} < \lambda < -\frac{c_j'(a_j)}{d_j'(a_j)}.$$

In order to describe cases a), b), c) for both 1) and 2), it is convenient to introduce the index sets $J_a^\lambda, J_b^\lambda, J^\lambda$ defined by (14), (15), (16), respectively. It is obvious that $J_a^\lambda \cup J_b^\lambda \cup J^\lambda = J$. The "necessity" part of Theorem 1 is proved.

Sufficiency. Conversely, let $\mathbf{x}^* \in X$ and components of $\mathbf{x}^*$ satisfy (14), (15), (16), where $\lambda \geq 0$.

1) Let $\lambda > 0$. Then $c_j'(x_j^*) < 0$, $j \in J^\lambda$, according to (16) and $d_j'(x_j^*) > 0$. Set:

$$\lambda = -\frac{c_j'(x_j^*)}{d_j'(x_j^*)} = \lambda(\mathbf{x}^*)\ (>0), \text{ obtained from } \sum_{j\in J_a^\lambda} d_j(a_j) + \sum_{j\in J_b^\lambda} d_j(b_j) + \sum_{j\in J^\lambda} d_j(x_j^*) = \alpha;$$

$$u_j = v_j = 0 \qquad \text{for } j \in J^\lambda;$$

$$u_j = c_j'(a_j) + \lambda d_j'(a_j) \quad (\geq 0 \text{ according to the definition of } J_a^\lambda), \quad v_j = 0 \qquad \text{for } j \in J_a^\lambda;$$

$$u_j = 0, \quad v_j = -c_j'(b_j) - \lambda d_j'(b_j) \quad (\geq 0 \text{ according to the definition of } J_b^\lambda) \quad \text{for } j \in J_b^\lambda.$$

By using these expressions, it is easy to check that conditions (5), (6), (7), (8), (11) are satisfied. Conditions (9), (10) are also satisfied according to the assumption that $\mathbf{x}^* \in X$.

2) Let $\lambda = 0$. Then $c_j'(x_j^*) = 0$, $j \in J^{\lambda=0}$, in accordance with (16). Since $d_j'(x_j) > 0$ for each x_j then $c_j'(b_j) > 0$, $c_j'(a_j) < 0$, $j \in J^0 \equiv \left\{ j \in J : -\frac{c_j'(b_j)}{d_j'(b_j)} < 0 < -\frac{c_j'(a_j)}{d_j'(a_j)} \right\}$. Therefore there exists an $x_j^* = h_j^{\leq} \in (a_j, b_j)$ such that $c_j'(x_j^*) = 0$ according to the Darboux theorem. Set:

$$\lambda = -\frac{c_j'(x_j^*)}{d_j'(x_j^*)} \quad (=0); \quad u_j = v_j = 0 \qquad \text{for} \quad j \in J^{\lambda=0};$$

$$u_j = c_j'(a_j) + \lambda d_j'(a_j) = c_j'(a_j) \quad (\geq 0), \quad v_j = 0 \qquad \text{for } j \in J_a^{\lambda=0};$$

$$u_j = 0, \quad v_j = -c_j'(b_j) - \lambda d_j'(b_j) = -c_j'(b_j) \quad (\geq 0) \qquad \text{for } j \in J_b^{\lambda=0}.$$

Clearly, conditions (5), (6), (7), (11) are satisfied; conditions (9), (10) are also satisfied according to the assumption $\mathbf{x}^* \in X$, and condition (8) obviously is satisfied for $\lambda = 0$.

In both cases 1), 2) of the "sufficiency" part, $x_j^*, \lambda, u_j, v_j, j \in J$, satisfy KKT conditions (5) – (11) which are necessary and sufficient conditions for a feasible solution to be an optimal solution to a convex minimization problem. Therefore $\mathbf{x}^*$ is an optimal solution to Problem (C). Since $c_j(x_j), j \in J$, are strictly convex functions, this optimal solution is unique. ■

In view of the discussion above, the importance of Theorem 1 consists in the fact that it describes components of the optimal solution to Problem (C) only through the Lagrange multiplier λ associated with the inequality constraint (2).

Since we do not know the optimal value of the Lagrange multiplier λ from the statement of Theorem 1, in order to obtain the optimal solution to Problem (C) (or to establish that Problem (C) does not have an optimal solution), in Section 3 we define an iterative algorithm with respect to the Lagrange multiplier λ and we prove convergence and discuss computational complexity of this algorithms.

Using that $d_j'(x_j) > 0, j \in J$, from strict monotonicity of $c_j'(x_j)$ and from $a_j \leq b_j$, $j \in J$, it follows that $ub_j \overset{\text{def}}{=} -\frac{c_j'(b_j)}{d_j'(b_j)} < -\frac{c_j'(a_j)}{d_j'(a_j)} \overset{\text{def}}{=} la_j$, $j \in J$, for the expressions by means of which we define the index sets J_a^λ, J_b^λ, and J^λ.

The problem how to ensure a feasible solution to Problem (C) (1) – (3), which is an assumption of Theorem 1, is also discussed in Section 3.

3 The Algorithm for Problem (C)

3.1 Analysis of the optimal solution to Problem (C)

Before the formal statement of the algorithm for Problem (C), we discuss some properties of the optimal solution to this problem.

By using (14), (15) and (16) of Theorem 1, condition (8) can be written in the form

$$\lambda\left(\sum_{j\in J_a^\lambda} d_j(a_j) + \sum_{j\in J_b^\lambda} d_j(b_j) + \sum_{j\in J^\lambda} d_j(x_j^*) - \alpha\right) = 0, \quad \lambda \geq 0. \tag{8$'$}$$

Since the optimal solution $\mathbf{x}^*$ to Problem (C) depends on λ (Theorem 1), we consider the components of $\mathbf{x}^*$ as functions of λ for different $\lambda \in \mathbb{R}^1_+$:

$$x_j(\lambda) = \begin{cases} a_j, & \lambda \geq -\dfrac{c_j'(a_j)}{d_j'(a_j)} \\ b_j, & \lambda \leq -\dfrac{c_j'(b_j)}{d_j'(b_j)} \\ x_j^* : \ c_j'(x_j^*) + \lambda d_j'(x_j^*) = 0, & -\dfrac{c_j'(b_j)}{d_j'(b_j)} \leq \lambda \leq -\dfrac{c_j'(a_j)}{d_j'(a_j)}. \end{cases} \tag{19}$$

Functions $x_j(\lambda), j \in J$, are piecewise, monotone, piecewise differentiable functions of λ with two breakpoints at $\lambda = -\frac{c_j'(a_j)}{d_j'(a_j)}$ and $\lambda = -\frac{c_j'(b_j)}{d_j'(b_j)}, j \in J$.

Let

$$\delta(\lambda) \overset{\text{def}}{=} \sum_{j\in J_a^\lambda} d_j(a_j) + \sum_{j\in J_b^\lambda} d_j(b_j) + \sum_{j\in J^\lambda} d_j(x_j(\lambda)) - \alpha. \tag{20}$$

According to (19) and $u_j = v_j = 0,\ j \in J^\lambda$, condition (5) becomes

$$c_j'(x_j(\lambda)) + \lambda d_j'(x_j(\lambda)) = 0, \quad j \in J^\lambda. \tag{21}$$

Differentiating both sides of these expressions with respect to λ (using that $c_j''(x_j), d_j''(x_j), j \in J$, exist by assumption, $x_j'(\lambda)$ exist for all $j \in J^\lambda$ because $x_j(\lambda)$ are defined by $x_j(\lambda) = x_j^*$ such that $c_j'(x_j^*) + \lambda d_j'(x_j^*) = 0$ for $j \in J^\lambda$), we obtain

$$c_j''(x_j(\lambda))x_j'(\lambda) + d_j'(x_j(\lambda)) + \lambda d_j''(x_j(\lambda))x_j'(\lambda) = 0, \quad j \in J^\lambda. \tag{22}$$

Therefore

$$x_j'(\lambda) = -\frac{d_j'(x_j(\lambda))}{\lambda d_j''(x_j(\lambda)) + c_j''(x_j(\lambda))}, \quad j \in J^\lambda, \tag{23}$$

and since $c_j''(x_j) > 0,\ d_j''(x_j) \geq 0, j \in J$, as the second derivatives of strictly convex and convex differentiable functions, respectively, $d_j'(x_j) > 0$ by assumption, $\lambda \geq 0$, then $x_j'(\lambda) < 0,\ j \in J^\lambda$. (If we assume that $\lambda d_j''(x_j(\lambda)) + c_j''(x_j(\lambda)) = 0$, then $d_j'(x_j(\lambda)) = 0, j \in J^\lambda$, according to (22). However, $d_j'(x_j) > 0,\ j \in J$, by the assumption, a contradiction.) Consequently

$$\delta'(\lambda) \equiv \sum_{j\in J^\lambda} d_j'(x_j(\lambda))x_j'(\lambda) < 0 \tag{24}$$

when $J^\lambda \neq \emptyset$, and $\delta'(\lambda) = 0$ when $J^\lambda = \emptyset$. Hence, $\delta(\lambda)$ is *a monotone nonincreasing function* of $\lambda \in \mathbb{R}^1_+$, and $\max_{\lambda \geq 0} \delta(\lambda)$ is attained at the minimum admissible value of λ, that is, at $\lambda = 0$.

Case 1. If $\delta(0) > 0$, in order that ($8'$) and (9) $\equiv$ (2) be satisfied, there exists some $\lambda^* > 0$ such that $\delta(\lambda^*) = 0$, that is,

$$\sum_{j \in J} d_j(x_j^*) = \alpha, \tag{25}$$

which means that the inequality constraint (2) is satisfied with an equality for λ^* in this case.

Case 2. If $\delta(0) < 0$ then $\delta(\lambda) < 0$ for all $\lambda \geq 0$, and the maximum of $\delta(\lambda)$ with $\lambda \geq 0$ is $\delta(0) = \max_{\lambda \geq 0} \delta(\lambda)$ and it is attained at $\lambda = 0$ in this case. In order that ($8'$) be satisfied, λ must be equal to 0. Therefore $x_j^* = h_j^{\leq}$, $j \in J^{\lambda=0}$, according to (16) and definition of $h_j^{\leq}$.

Case 3. In the special case when $\delta(0) = 0$, the maximum value $\delta(0) = \max_{\lambda \geq 0} \delta(\lambda)$ of $\delta(\lambda)$ is also attained at the minimum admissible value of λ, that is, at $\lambda = 0$, because $\delta(\lambda)$ is a monotone nonincreasing function in accordance with the above discussion.

As we have seen, for the optimal value of λ we have $\lambda \geq 0$ in all possible cases, as the KKT condition (8) requires. We have shown that in Case 1 we need an algorithm for calculating λ^* which satisfies the KKT conditions (5) – (11) and such that λ^* satisfies (9) with an equality. In order that this be satisfied, the set

$$X^= \overset{\text{def}}{=} \{\mathbf{x} \in \mathbb{R}^n : \sum_{j \in J} d_j(x_j) = \alpha,\ a_j \leq x_j \leq b_j, j \in J\}$$

must be nonempty. That is why we have required $\alpha \leq \sum_{j \in J} d_j(b_j)$ in some cases in addition to the assumption $\sum_{j \in J} d_j(a_j) \leq \alpha$ (assumption (II), Section 2). We have also used this assumption in the proof of Theorem 1, "sufficiency" part, when $\lambda > 0$.

Using equation $\delta(\lambda) = 0$, where $\delta(\lambda)$ is defined by (20), we can determine λ as an implicit function of $\mathbf{x}$

$$\lambda = \lambda(\mathbf{x}), \tag{26}$$

because $\delta'(\lambda) < 0$ according to (24) when $J^\lambda \neq \emptyset$ (it is important that $\delta'(\lambda) \neq 0$). When it is possible to obtain a closed form expression of λ, we use it in the algorithm suggested for Problem (C). It turns out that without loss of generality, we can assume that $\delta'(\lambda) \neq 0$, that is, $\delta(\lambda)$ depends on λ, which means that $J^\lambda \neq \emptyset$ (see also the third paragraph of Remark 2 below).

At iteration k of the implementation of the algorithm, denote by $\lambda^{(k)}$ the value of the Lagrange multiplier associated with the constraint (2), by $\alpha^{(k)}$ the right-hand side of (2), and by $J^{(k)}, J_a^{\lambda(k)}, J_b^{\lambda(k)}, J^{\lambda(k)}$ the current sets $J, J_a^\lambda, J_b^\lambda, J^\lambda$, respectively.

3.2 Statement of Algorithm 1 (for solving Problem (C))

According to Theorem 1 and the preliminary analysis, we can suggest the following algorithm for solving Problem (C) with strictly convex differentiable functions $c_j(x_j), j \in J$.

Algorithm 1

```
0. (Initialization) $J := \{1, \dots, n\}, k := 0, \alpha^{(0)} := \alpha, n^{(0)} := n, J^{(0)} := J, J_a^\lambda := \emptyset, J_b^\lambda := \emptyset$,
   initialize $h_j^{\leq}, j \in J$. If $\sum_{j \in J} d_j(a_j) \leq \alpha$ go to 1; else go to 9.
1. Construct the sets $J_a^0, J_b^0, J^0$ (for $\lambda = 0$). Calculate
       $\delta(0) := \sum_{j \in J_a^0} d_j(a_j) + \sum_{j \in J_b^0} d_j(b_j) + \sum_{j \in J^0} d_j(h_j^{\leq}) - \alpha.$
   If $\delta(0) \leq 0$ then $\lambda := 0$, go to 8
   else if $\delta(0) > 0$ then:
     if $\alpha \leq \sum_{j \in J} d_j(b_j)$ then go to 2
     else if $\alpha > \sum_{j \in J} d_j(b_j)$ go to 9 (there does not exist $\lambda^* > 0$ such that $\delta(\lambda^*) = 0$) .
2. $J^{\lambda(k)} := J^{(k)}$. Calculate $\lambda^{(k)}$ by using the explicit expression of $\lambda$, determined from
   the equality $\sum_{j \in J^{\lambda(k)}} d_j(x_j) = \alpha^{(k)}$, where $x_j, j \in J^{\lambda(k)}$, are given by (16). Go to 3.
3. Construct the sets $J_a^{\lambda(k)}, J_b^{\lambda(k)}, J^{\lambda(k)}$ through (14), (15), (16) (with $j \in J^{(k)}$ instead of
   $j \in J$) and find their cardinal numbers $|J_a^{\lambda(k)}|, |J_b^{\lambda(k)}|, |J^{\lambda(k)}|$, respectively. Go to 4.
4. Calculate
       $\delta(\lambda^{(k)}) := \sum_{j \in J_a^{\lambda(k)}} d_j(a_j) + \sum_{j \in J_b^{\lambda(k)}} d_j(b_j) + \sum_{j \in J^{\lambda(k)}} d_j(x_j^*) - \alpha^{(k)}$
   where $x_j^*, j \in J^{\lambda(k)}$, are determined from (16) with $\lambda = \lambda^{(k)}$. Go to 5.
5. If $\delta(\lambda^{(k)}) = 0$ or $J^{\lambda(k)} = \emptyset$ then $\lambda := \lambda^{(k)}, J_a^\lambda := J_a^\lambda \cup J_a^{\lambda(k)}, J_b^\lambda := J_b^\lambda \cup J_b^{\lambda(k)}, J^\lambda := J^{\lambda(k)}$,
     go to 8
   else if $\delta(\lambda^{(k)}) > 0$ go to 6
   else if $\delta(\lambda^{(k)}) < 0$ go to 7.
6. $x_j^* := a_j$ for $j \in J_a^{\lambda(k)}$, $\alpha^{(k+1)} := \alpha^{(k)} - \sum_{j \in J_a^{\lambda(k)}} d_j(a_j)$, $J^{(k+1)} := J^{(k)} \setminus J_a^{\lambda(k)}$,
   $n^{(k+1)} := n^{(k)} - |J_a^{\lambda(k)}|, J_a^\lambda := J_a^\lambda \cup J_a^{\lambda(k)}, k := k + 1$. Go to 2.
7. $x_j^* := b_j$ for $j \in J_b^{\lambda(k)}$, $\alpha^{(k+1)} := \alpha^{(k)} - \sum_{j \in J_b^{\lambda(k)}} d_j(b_j)$, $J^{(k+1)} := J^{(k)} \setminus J_b^{\lambda(k)}$,
   $n^{(k+1)} := n^{(k)} - |J_b^{\lambda(k)}|, J_b^\lambda := J_b^\lambda \cup J_b^{\lambda(k)}, k := k + 1$. Go to 2.
8. $x_j^* := a_j$ for $j \in J_a^\lambda$; $x_j^* := b_j$ for $j \in J_b^\lambda$; assign $x_j^*$ the value determined from (16) for
   $j \in J^\lambda$. Go to 10.
9. Problem $(C)$ has no optimal solution because $X = \emptyset$ or there does not exist $\lambda > 0$
   satisfying Theorem 1.
10. End.
```

Remark 1 *To avoid a possible "endless loop" in programming the algorithm, the criterion of step 5 to go to step 8 at iteration k usually is not $\delta(\lambda^{(k)}) = 0$ but $\delta(\lambda^{(k)}) \in [-\varepsilon, \varepsilon]$ where $\varepsilon > 0$ is some given tolerance value up to which the equality $\delta(\lambda^*) = 0$ may be satisfied.*

3.3 Convergence and computational complexity of Algorithm 1

The following Theorem 2 states convergence of Algorithm 1, that is, "convergence" of $\lambda^{(k)}$, $J^{\lambda(k)}$, $J_a^{\lambda(k)}$, $J_b^{\lambda(k)}$, generated by Algorithm 1, to the optimal $\lambda, J^\lambda, J_a^\lambda, J_b^\lambda$ from Theorem 1, respectively.

Theorem 2 *Let $\{\lambda^{(k)}\}$ be the sequence generated by Algorithm 1. Then*
i) if $\delta(\lambda^{(k)}) > 0$ then $\lambda^{(k)} \leq \lambda^{(k+1)}$;
ii) if $\delta(\lambda^{(k)}) < 0$ then $\lambda^{(k)} \geq \lambda^{(k+1)}$.

Proof. Denote by $x_j^{(k)}$ the components of $\mathbf{x}^{(k)} = (x_j)_{j \in J^{(k)}}$ at iteration k of implementation of Algorithm 1.

Taking into consideration (24), Case 1, Case 2, Case 3 of subsection 3.1, and step 1 (sign of $\delta(0)$) and step 2 of Algorithm 1, it follows that $\lambda^{(k)} \geq 0$ for every k. Since $x_j^{(k)}$ are determined from (16): $\lambda^{(k)} d_j'(x_j^{(k)}) + c_j'(x_j^{(k)}) = 0$, substituted in $\sum_{j \in J^{\lambda(k)}} d_j(x_j^{(k)}) = \alpha^{(k)}$ at step 2 of Algorithm 1, and since $\lambda^{(k)} \geq 0, d_j'(x_j^{(k)}) > 0$, then $c_j'(x_j^{(k)}) \leq 0$, that is, $-c_j'(x_j^{(k)}) \geq 0$.

i) Let $\delta(\lambda^{(k)}) > 0$. Using step 6 of Algorithm 1, which is performed when $\delta(\lambda^{(k)}) > 0$, we obtain

$$\sum_{j \in J^{\lambda(k+1)}} d_j(x_j^{(k)}) \equiv \sum_{j \in J^{(k+1)}} d_j(x_j^{(k)}) = \sum_{j \in J^{(k)} \backslash J_a^{\lambda(k)}} d_j(x_j^{(k)}) = \alpha^{(k)} - \sum_{j \in J_a^{\lambda(k)}} d_j(x_j^{(k)}). \quad (27)$$

Our purpose is to prove that $x_j^{(k)} \leq a_j$ for $j \in J_a^{\lambda(k)}$. If we assume the contrary, that $x_j^{(k)} > a_j$ for $j \in J_a^{\lambda(k)}$, then $d_j'(x_j^{(k)}) \geq d_j'(a_j)$ according to (13), and since $d_j'(x_j^{(k)}) > 0$ then $\frac{d_j'(a_j)}{d_j'(x_j^{(k)})} \in (0, 1]$. According to the definition (14) of $J_a^{\lambda(k)}$ and relation $\lambda^{(k)} d_j'(x_j^{(k)}) = -c_j'(x_j^{(k)})$, obtained from (16) at iteration No. k, we have

$$-\frac{c_j'(a_j)}{d_j'(a_j)} \leq \lambda^{(k)} = -\frac{c_j'(x_j^{(k)})}{d_j'(x_j^{(k)})}, \quad (28)$$

and therefore $-c_j'(a_j) \leq -\frac{c_j'(x_j^{(k)}) d_j'(a_j)}{d_j'(x_j^{(k)})}$. Using that $-c_j'(x_j^{(k)}) \geq 0, d_j'(x_j^{(k)}) > 0$ and $\frac{d_j'(a_j)}{d_j'(x_j^{(k)})} \in (0, 1]$ by assumption, it follows that $-c_j'(a_j) \leq -c_j'(x_j^{(k)})$. Hence $x_j^{(k)} \leq a_j$ in accordance with (12), a contradiction with the assumption that $a_j < x_j^{(k)}$. Therefore the assumption $a_j < x_j^{(k)}, j \in J_a^{\lambda(k)}$, is wrong.

Using that $d_j(x_j)$ is an increasing function ($d_j'(x_j) > 0$) by assumption, $a_j \geq x_j^{(k)}, j \in J_a^{\lambda(k)}$, and step 6 of Algorithm 1, from (27) we get

$$\sum_{j \in J^{\lambda(k+1)}} d_j(x_j^{(k)}) = \alpha^{(k)} - \sum_{j \in J_a^{\lambda(k)}} d_j(x_j^{(k)}) \geq \alpha^{(k)} - \sum_{j \in J_a^{\lambda(k)}} d_j(a_j) = \alpha^{(k+1)} = \sum_{j \in J^{\lambda(k+1)}} d_j(x_j^{(k+1)}).$$

Therefore there exists at least one $j_0 \in J^{\lambda(k+1)}$ such that $d_{j_0}(x_{j_0}^{(k)}) \geq d_{j_0}(x_{j_0}^{(k+1)})$, and since $d_j(x_j)$ is an increasing function, then $x_{j_0}^{(k)} \geq x_{j_0}^{(k+1)}$. Hence

$$\lambda^{(k)} = -\frac{c_{j_0}'(x_{j_0}^{(k)})}{d_{j_0}'(x_{j_0}^{(k)})} \leq -\frac{c_{j_0}'(x_{j_0}^{(k+1)})}{d_{j_0}'(x_{j_0}^{(k)})} \leq -\frac{c_{j_0}'(x_{j_0}^{(k+1)})}{d_{j_0}'(x_{j_0}^{(k+1)})} = \lambda^{(k+1)}. \quad (29)$$

We have used the relationship (16) between $\lambda^{(k)}$ and $x_j^{(k)}$ for $j \in J^{\lambda(k)}$ according to step 2 of Algorithm 1, the fact that $-c_{j_0}'(x_{j_0})$ is a monotone decreasing function according to

(12), $-c'_j(x_j^{(k)}) \ge 0$, $j \in J^{\lambda(k)}$, according to (16) with $\lambda^{(k)} \ge 0$ and $d'_j(x_j^{(k)}) > 0$, and that $\frac{1}{d'_j(x_j)}$ is a monotone nonincreasing function as a reciprocal of the derivative of a convex function $d_j(x_j)$ with $d'_j(x_j) > 0$.

The proof of part ii) is omitted because it is similar to that of part i). ■

Remark 2 *Since we do not know the optimal value of λ which is involved in the statement of Theorem 1, we approximate the value of λ until we obtain its optimal value at the final iteration of algorithm performance. In order to determine the current value $\lambda^{(k)}$ of λ at each iteration of Algorithm 1, including the initial value, we assume that $J^{\lambda(k)} = J^{(k)}$ at the beginning of the corresponding iteration (step 2).*

Theorem 2, definitions of J_a^λ (14), J_b^λ (15), J^λ (16) and steps 6 and 7 of Algorithm 1 allow us to assert that the values of $\lambda^{(k)}, k = 0, 1, \ldots$, calculated at step 2, are such that $j \in J_a^{\lambda(k)}$ implies $j \in J_a^{\lambda(k+1)}$, $j \in J_b^{\lambda(k)}$ implies $j \in J_b^{\lambda(k+1)}$, and since $J^{\lambda(k)}$ is reduced (steps 6 and 7 of Algorithm 1), then $j \in J^{\lambda(k+1)}$ implies $j \in J^{\lambda(k)}$. That is, we have $J_a^{\lambda(k)} \subseteq J_a^{\lambda(k+1)}$, $J_b^{\lambda(k)} \subseteq J_b^{\lambda(k+1)}$, and $J^{\lambda(k)} \supseteq J^{\lambda(k+1)}$. This means that if j belongs to current index set $J_a^{\lambda(k)}$, then j belongs to the next index set $J_a^{\lambda(k+1)}$ and so on, this j belongs to the "optimal" index set J_a^λ according to Theorem 2 and definition (14); the same holds true about the index sets $J_b^{\lambda(k)}$ and J_b^λ (15). Therefore $\lambda^{(k)}$ converges to the optimal value λ from the statement of Theorem 1, and $J_a^{\lambda(k)}, J_b^{\lambda(k)}, J^{\lambda(k)}$ "converge" to $J_a^\lambda, J_b^\lambda, J^\lambda$, respectively. This means that the calculation of λ, operations $x_j^* := a_j$, $j \in J_a^{\lambda(k)}$ (step 6), $x_j^* := b_j$, $j \in J_b^{\lambda(k)}$ (step 7), and the construction of $J_a^\lambda, J_b^\lambda, J^\lambda$ are in accordance with Theorem 1. The final sets $J_a^\lambda, J_b^\lambda, J^\lambda$ are constructed at step 1 or at step 5 (when $\delta(\lambda^{(k)}) = 0$ or $J^{\lambda(k)} = \emptyset$) of iteration k, where k is the number of the last iteration of algorithm performance.

Since at the beginning of Algorithm 1 we have $J^{\lambda(0)} := J$ (steps 0 and 2) and since $J^{\lambda(k)} \supseteq J^{\lambda(k+1)}$, then $J^{\lambda(k)} \ne \emptyset$ for all $k \le K$, where K is some nonnegative integer. If we obtain $J^{\lambda(K)} = \emptyset$, this would mean that $J_a^{\lambda(K)} \cup J_b^{\lambda(K)} = J$, that is, Problem (C) has been already solved at iteration K, and $\delta(\lambda^{(K)}) = const$.

As we have seen in the proof of Theorem 2, Algorithm 1 guarantees that $\lambda \ge 0$ for Problem (C) as Theorem 1 requires. In the proof of Theorem 2 we have essentially used that $\lambda^{(k)} \ge 0$ in order to deduce that $-c'_j(x_j^{(k)}) \ge 0$.

Consider the *feasibility* of $\mathbf{x}^* = (x_j^*)_{j \in J}$, generated by Algorithm 1.

Components $x_j^* = a_j, j \in J_a^\lambda$, and $x_j^* = b_j, j \in J_b^\lambda$, obviously satisfy (3). Let $j \in J^\lambda$. Suppose that $x_j^* < a_j$ or $x_j^* > b_j$ for $j \in J^\lambda$. By monotonicity of $c'_j(x_j)$ and $d'_j(x_j)$ and strict convexity of $c_j(x_j)$ it follows that $c'_j(x_j^*) < c'_j(a_j)$, $d'_j(x_j^*) \le d'_j(a_j)$ or $c'_j(x_j^*) > c'_j(b_j)$, $d'_j(x_j^*) \ge d'_j(b_j)$, respectively. Hence, using that $d'_j(.) > 0$ by assumption, $-c'_j(.) \ge 0$, $j \in J^\lambda$ (see proof of Theorem 2), we get

$$\lambda \equiv \frac{-c'_j(x_j^*)}{d'_j(x_j^*)} > \frac{-c'_j(a_j)}{d'_j(x_j^*)} \ge \frac{-c'_j(a_j)}{d'_j(a_j)} \quad \text{or} \quad \lambda \equiv \frac{-c'_j(x_j^*)}{d'_j(x_j^*)} < \frac{-c'_j(b_j)}{d'_j(x_j^*)} \le \frac{-c'_j(b_j)}{d'_j(b_j)},$$

that is, $j \notin J^\lambda$, a contradiction. Therefore the assumption is wrong, and $x_j^* \in (a_j, b_j), j \in J^\lambda$. Consequently, all $x_j^*, j \in J$, satisfy (3).

We have proved in subsection 3.1 that if $\delta(0) > 0$ and $X \ne \emptyset$, where X is defined by (2) – (3), then there exists a $\lambda^* \ge 0$ such that $\delta(\lambda^*) = 0$ (Case 1). Since at step 2 of

Algorithm 1 we determine $\lambda^{(k)}$ from the equality $\sum_{j\in J^{\lambda(k)}} d_j(x_j^{(k)}) = \alpha^{(k)}$ for each iteration k, then (2) is satisfied with an equality in this case. Otherwise, if $\delta(0) < 0$ then we set $\lambda = 0$ (step 1) and we have $\sum_{j\in J} d_j(x_j(0)) - \alpha \equiv \delta(0) < 0$, that is, (2) is also satisfied in this case but as a strict inequality. When $\delta(0) = 0$, since $\delta(\lambda)$ is monotone nonincreasing, then (2) is also satisfied as a strict inequality.

Therefore $\mathbf{x}^*$, obtained by Algorithm 1, is feasible for Problem (C), which is an assumption of Theorem 1.

At each iteration, Algorithm 1 calculates the value of at least one variable (steps 6, 7, 8) and at each iteration we solve a problem of the type (C) but *of less dimension* (steps 2 – 7). Therefore Algorithm 1 is finite and it converges with at most $n = |J|$ iterations, that is, the iteration complexity of Algorithm 1 is $\mathcal{O}(n)$.

Step 0 (initialization and checking whether the feasible region X is empty) takes time $\mathcal{O}(n)$. Step 1 (construction of sets J_a^0, J_b^0, J^0 and calculation of $\delta(0)$) also takes time $\mathcal{O}(n)$. The calculation of $x_j^{(k)}, j \in J$, and $\lambda^{(k)}$ requires $\mathcal{O}(n)$ time (step 2). Step 3 takes $\mathcal{O}(n)$ time because of the construction of index sets $J_a^{\lambda(k)}, J_b^{\lambda(k)}, J^{\lambda(k)}$. Steps 4 also requires $\mathcal{O}(n)$ time, and step 5 requires constant time. Each of steps 6, 7 and 8 takes time which is bounded by $\mathcal{O}(n)$ because at these steps we assign some of x_j's the optimal value, and since the number of all x_j's is n, then steps 6, 7 and 8 take time $\mathcal{O}(n)$. Hence, Algorithm 1 has $\mathcal{O}(n^2)$ running time and it belongs to the class of strongly polynomially bounded algorithms.

As the computational experiments show (those presented in Section 5 as well as many other experiments), the number of iterations of the algorithm performance is not only at most n but it is much less than n for large n. In fact, this number does not depend on n but only on the three index sets $J_a^\lambda, J_b^\lambda, J^\lambda$ defined by (14), (15), (16), respectively. In practice, Algorithm 1 has $\mathcal{O}(n)$ running time.

3.4 Commentary

Some of the main characteristics of the approach suggested in this paper are the following.

Since the method uses values of the first derivatives of the objective function $c(\mathbf{x})$, we can consider it as a *first-order method*. Also, this method is a *saddle-point method* or, more precisely, a *dual variables saddle-point method* because it is based on convergence with respect to the Lagrange multiplier (dual variable) λ associated with the inequality constraint (2).

At step 2 of Algorithm 1 we use the expression of $\lambda^{(k)}$, determined from the equality $\delta(\lambda^{(k)}) = 0$, where x_j^* are from (16), $j \in J^{\lambda(k)} = J^{(k)}$. As it has been proved, under the assumptions we can always determine $\lambda = \lambda(\mathbf{x}^*)$ from $\delta(\lambda) = 0$ as an implicit function of $\mathbf{x}^*$ (see (26)). For example, when functions $d_j(x_j), j \in J$, are linear, the explicit expression of λ is always available. Other examples of functions for which it is possible to obtain a closed form expression of λ are given in Section 5. Of course, there are also other functions $c_j(x_j)$, $d_j(x_j)$, $j \in J$, for which the approach, suggested in this paper, is applicable and gives good results.

When the optimal Lagrange multiplier λ^* associated with (2) is known, then Problem (C) (1) – (3) can be replaced by the following convex separable optimization problem

$$\min \left\{ \sum_{j \in J} [c_j(x_j) + \lambda^* d_j(x_j)] - \lambda^* \alpha \right\}$$

subject to

$$\mathbf{x} \in A,$$

where

$$A = \{\mathbf{x} \in \mathbb{R}^n : a_j \leq x_j \leq b_j,\ j \in J\}.$$

The problem dual to (C) is

$$\max\ \Psi(\lambda)$$

subject to

$$\lambda \in \mathbb{R}^1_+,$$

where

$$\Psi(\lambda) = \min_{\mathbf{x} \in A} \left\{ \sum_{j \in J} [c_j(x_j) + \lambda d_j(x_j)] - \lambda \alpha \right\}.$$

Thus, using the Lagrangian duality and Theorem 1, we have replaced the *multivariable* Problem (C) of $\mathbf{x} \in \mathbb{R}^n$ by the *single-variable* optimization problem of the above type for finding $\lambda \in \mathbb{R}^1_+$.

4 Extensions

4.1 Theoretical aspects

In the above discussion, we have required $d'_j(.) > 0, j \in J$, in constraint (2) of Problem (C). However, if it is allowed: i) $d'_j(x_j) \equiv 0$; or ii) $d'_j(x_j) \not\equiv 0$ but $d'_j(a_j) = 0$ and/or $d'_j(b_j) = 0$ for some $j \in J$ in (2), then for such indices j we cannot construct the expressions $-\frac{c'_j(a_j)}{d'_j(a_j)}$ and/or $-\frac{c'_j(b_j)}{d'_j(b_j)}$, by means of which we define index sets J_a^λ (14), J_b^λ (15), and J^λ (16). In case i) we have $d_j(x_j) =: d_j = const$ and x_j's are not involved in (2) for such indices j.

It turns out that we can avoid this difficulty and solve Problem (C) with $d'_j(x_j) \equiv 0$ or $d'_j(x_j) \not\equiv 0$ but $d'_j(a_j) = 0$ and/or $d'_j(b_j) = 0$ for some $j \in J$.

Denote

$$Z0 = \{j \in J : d'_j(x_j) \equiv 0\},$$

$$ZA = \{j \in J \setminus Z0 : d'_j(a_j) = 0\},$$

$$ZB = \{j \in J \setminus Z0 : d'_j(b_j) = 0\}.$$

Here "0" is the "computer zero". In particular, when $J = Z0$ and $d_j(x_j) =: d_j = 0, j \in J$, $\alpha = 0$, then feasible region X (2) – (3) is defined only by (3).

Theorem 3 (Characterization of the optimal solution to Problem (C): an extended version)

Problem (C) can be decomposed into two subproblems: $(C1)$ for $j \in Z0$ and $(C2)$ for $j \in J \setminus Z0$ with $\alpha := \alpha - \sum_{j\in Z0} d_j(x_j^*) \equiv \alpha - \sum_{j\in Z0} d_j$.

The optimal solution to $(C1)$ is

$$x_j^* = \begin{cases} a_j, & j \in Z0,\ h_j^{\leq} \leq a_j \\ b_j, & j \in Z0,\ h_j^{\leq} \geq b_j \\ h_j^{\leq}, & j \in Z0,\ a_j < h_j^{\leq} < b_j, \end{cases} \tag{30}$$

that is, subproblem $(C1)$ itself is decomposed into $n_0 \equiv |Z0|$ independent problems.

The optimal solution to $(C2)$ is given by (14), (15), (16) with $J := J \setminus Z0$, $\alpha := \alpha - \sum_{j\in Z0} d_j(x_j^*) \equiv \alpha - \sum_{j\in Z0} d_j$, where we adopt $\frac{c_j'(a_j)}{d_j'(a_j)} = \lim_{t\to a_j} \frac{c_j'(t)}{d_j'(t)}$ if $j \in ZA$, and we adopt $\frac{c_j'(b_j)}{d_j'(b_j)} = \lim_{t\to b_j} \frac{c_j'(t)}{d_j'(t)}$ when $j \in ZB$.

It is permissible some of the limits above be equal to $-\infty$ or $+\infty$.

Proof of Theorem 3 repeats in part the proof of Theorem 1.

Proof. *Necessity.* Let $\mathbf{x}^* = (x_j^*)_{j\in J}$ be an optimal solution to Problem (C).

1) Let $j \in Z0$, that is, $d_j'(x_j) \equiv 0$. The KKT conditions for Problem (C) are

$$c_j'(x_j^*) - u_j + v_j = 0,\ j \in Z0 \quad \text{(from (5))} \quad \text{and (6)} - \text{(11)}.$$

a) If $x_j^* = a_j$, then $u_j \geq 0, v_j = 0$ according to (7) and (11). KKT condition (5) and definition of $h_j^{\leq}$ imply that $c_j'(x_j^*) = u_j \geq 0 \equiv c_j'(h_j^{\leq})$. Since $c_j'(x_j)$ is a monotone increasing function of x_j for each $j \in J$, then $x_j^* \equiv a_j \geq h_j^{\leq}$.

b) If $x_j^* = b_j$, then $u_j = 0, v_j \geq 0$ according to (6) and (11). Therefore (5) implies that $c_j'(x_j^*) = -v_j \leq 0 \equiv c_j'(h_j^{\leq})$. Using that $c_j'(x_j)$ is a monotone increasing function of x_j for each $j \in J$, we obtain $x_j^* \equiv b_j \leq h_j^{\leq}$.

c) If $a_j < x_j^* < b_j$, then $u_j = v_j = 0$ according to (6) and (7). Therefore (5) implies that $-c_j'(x_j^*) = 0$, that is, $x_j^* = h_j^{\leq}$ according to definition of $h_j^{\leq}$.

2) Components of the optimal solution to $(C2)$ are obtained by using the same approach as that of the "necessity" part of the proof of Theorem 1 but with the reduced index set $J := J \setminus Z0$ and reduced right-hand side of (2) $\alpha := \alpha - \sum_{j\in Z0} d_j(x_j^*) \equiv \alpha - \sum_{j\in Z0} d_j$.

Sufficiency. Conversely, let $\mathbf{x}^* = (x_j^*)_{j\in J} \in X$ and the components of $\mathbf{x}^*$ satisfy (30) for $j \in Z0$, and (14), (15), (16) with $J := J\setminus Z0$ and $\alpha := \alpha - \sum_{j\in Z0} d_j(x_j^*) \equiv \alpha - \sum_{j\in Z0} d_j$. Set:

$$\begin{array}{ll} \lambda = 0;\ u_j = v_j = 0 & \text{for } a_j < x_j^* < b_j,\ j \in Z0; \\ u_j = c_j'(a_j),\ v_j = 0 & \text{for } x_j^* = a_j,\ j \in Z0; \\ u_j = 0,\ v_j = -c_j'(b_j) & \text{for } x_j^* = b_j,\ j \in Z0. \end{array}$$

If $\lambda > 0$ set:

$$\begin{array}{ll} \lambda = -\dfrac{c_j'(x_j^*)}{d_j'(x_j^*)} = \lambda(\mathbf{x}^*) \quad (> 0) & \text{from (16)}; \\ u_j = v_j = 0 & \text{for } a_j < x_j^* < b_j,\ j \in J \setminus Z0; \\ u_j = c_j'(a_j) + \lambda d_j'(a_j) \quad (\geq 0), \quad v_j = 0 & \text{for } x_j^* = a_j,\ j \in J \setminus Z0; \\ u_j = 0, \quad v_j = -c_j'(b_j) - \lambda d_j'(b_j) \quad (\geq 0) & \text{for } x_j^* = b_j,\ j \in J \setminus Z0. \end{array}$$

If $\lambda = 0$ set:

$$\begin{array}{ll} \lambda = 0; \quad u_j = v_j = 0 & \text{for } a_j < x_j^* < b_j,\ j \in J \setminus Z0; \\ u_j = c_j'(a_j) \quad (\geq 0), \quad v_j = 0 & \text{for } x_j^* = a_j,\ j \in J \setminus Z0; \\ u_j = 0, \quad v_j = -c_j'(b_j) \quad (\geq 0) & \text{for } x_j^* = b_j,\ j \in J \setminus Z0. \end{array}$$

It can be verified that $\mathbf{x}^*, \lambda, u_j, v_j, j \in J$, satisfy the KKT conditions (5) – (11). Then $\mathbf{x}^*$ with components (30) for $j \in Z0$, and (14), (15), (16) with $\alpha := \alpha - \sum_{j \in Z0} d_j(x_j^*) \equiv \alpha - \sum_{j \in Z0} d_j$ for $j \in J \setminus Z0$ is an optimal solution to Problem $(C) = (C1) \cup (C2)$. ■

Thus, with the use of Theorem 3, we can express components $x_j^*, j \in Z0$, of the optimal solution to Problem (C) *without* the necessity of calculating expressions $-\frac{c_j'(a_j)}{d_j'(a_j)}$ with $d_j'(a_j) = 0$ and $-\frac{c_j'(b_j)}{d_j'(b_j)}$ with $d_j'(b_j) = 0$.

4.2 Computational aspects

Algorithm 1 can also be applied in cases when $a_j = -\infty$ for some $j \in J$ and/or $b_j = \infty$ for some $j \in J$. However, if we use the computer values of $-\infty$ and $+\infty$ at steps 0 and 1 of Algorithm 1 to check whether the feasible region X (2) – (3) is empty or nonempty, and at step 3 in the expressions $-\frac{c_j'(x_j)}{d_j'(x_j)}$ with $x_j = -\infty$ and/or $x_j = +\infty$, by means of which we construct index sets $J_a^\lambda, J_b^\lambda, J^\lambda$, this could sometimes lead to arithmetic overflow. If we use other values of $-\infty$ and $+\infty$ with smaller absolute values than those of the computer values of $-\infty$ and $+\infty$, it would lead to inconvenience and dependence on the data of the particular problems. To avoid these difficulties and to take into account the above discussion, it is convenient to do the following.

Construct the sets of indices:

$$\begin{aligned} SVN &= \{j \in J \setminus Z0 : a_j > -\infty, \quad b_j < +\infty\} \\ SV1 &= \{j \in J \setminus Z0 : a_j > -\infty, \quad b_j = +\infty\} \\ SV2 &= \{j \in J \setminus Z0 : a_j = -\infty, \quad b_j < +\infty\} \\ SV &= \{j \in J \setminus Z0 : a_j = -\infty, \quad b_j = +\infty\}. \end{aligned} \tag{31}$$

It is obvious that $Z0 \cup SV \cup SV1 \cup SV2 \cup SVN = J$, that is, the set $J \setminus Z0$ is partitioned into the four subsets $SVN, SV1, SV2, SV$, defined above.

In computer programming of Algorithm 1, we use computer values of $-\infty$ and $+\infty$ for constructing the sets $SVN, SV1, SV2, SV$.

In order to construct the sets $J_a^\lambda, J_b^\lambda, J^\lambda$ without the necessity of calculating the values $-\frac{c_j'(x_j)}{d_j'(x_j)}$ with $x_j = -\infty$ or $x_j = +\infty$, except for the sets $J, Z0, SV, SV1, SV2, SVN$, we need some subsidiary sets defined as follows.

For SVN:

$$J^{\lambda\, SVN} = \left\{ j \in SVN : -\frac{c_j'(b_j)}{d_j'(b_j)} < \lambda < -\frac{c_j'(a_j)}{d_j'(a_j)} \right\},$$

$$J_a^{\lambda\, SVN} = \left\{ j \in SVN : \lambda \geq -\frac{c_j'(a_j)}{d_j'(a_j)} \right\},$$

$$J_b^{\lambda\, SVN} = \left\{ j \in SVN : \lambda \leq -\frac{c_j'(b_j)}{d_j'(b_j)} \right\};$$

for $SV1$:

$$J^{\lambda\, SV1} = \left\{ j \in SV1 : \lambda < -\frac{c_j'(a_j)}{d_j'(a_j)} \right\}, \tag{32}$$

$$J_a^{\lambda\, SV1} = \left\{ j \in SV1 : \lambda \geq -\frac{c_j'(a_j)}{d_j'(a_j)} \right\};$$

for $SV2$:

$$J^{\lambda\, SV2} = \left\{ j \in SV2 : \lambda > -\frac{c_j'(b_j)}{d_j'(b_j)} \right\},$$

$$J_b^{\lambda\, SV2} = \left\{ j \in SV2 : \lambda \leq -\frac{c_j'(b_j)}{d_j'(b_j)} \right\};$$

for SV:

$$J^{\lambda\, SV} = SV.$$

Then

$$\begin{aligned} J^{\lambda} &:= J^{\lambda\, SVN} \cup J^{\lambda\, SV1} \cup J^{\lambda\, SV2} \cup J^{\lambda\, SV} \\ J_a^{\lambda} &:= J_a^{\lambda\, SVN} \cup J_a^{\lambda\, SV1} \\ J_b^{\lambda} &:= J_b^{\lambda\, SVN} \cup J_b^{\lambda\, SV2}. \end{aligned} \tag{33}$$

We use the sets $J^{\lambda}, J_a^{\lambda}, J_b^{\lambda}$, defined by (33), as the corresponding sets with the same names in Algorithm 1.

Using these index sets, the check whether feasible region X (2) – (3) is empty or nonempty is modified as follows.

i) If $SVN \cup SV1 = J \setminus Z0$, that is, if all a_j's are finite but some of (or all) b_j's are equal to $+\infty$ for the variables which are involved in (2), then it is sufficient to check whether

$$\sum_{j \in J \setminus Z0} d_j(a_j) \leq \alpha$$

and it is not necessary to check whether $\alpha \leq \sum_{j \in J \setminus Z0} d_j(b_j)$ at step 1 of Algorithm 1 in this case;

ii) Else if $SV2 \cup SV \neq \emptyset$, that is, if there exists at least one variable x_j which is involved in (2) with $a_j = -\infty$ then:

if $c_j(x_j) \geq 0, j \in J$, and $SVN \cup SV1 \neq \emptyset$, then it is sufficient to check whether

$$\sum_{j \in SVN \cup SV1} d_j(a_j) \leq \alpha$$

else $X \neq \emptyset$ and it is not necessary to check anything else in this case.

With the use of results of this section, steps 0, 1 and 3 of Algorithm 1 can be modified as follows.

```
Step 0^1. (Initialization) $J := \{1, \dots, n\}$, $k := 0, \alpha^{(0)} := \alpha, n^{(0)} := n$, $J^{(0)} := J$,
    $J_a^\lambda := \emptyset, J_b^\lambda := \emptyset$, initialize $h_j^{\leq}, j \in J$.
    Construct the set $Z0$. If $j \in Z0$ then:
      if $h_j^{\leq} \leq a_j$ then $x_j^* := a_j$
      else if $h_j^{\leq} \geq b_j$ then $x_j^* := b_j$
      else if $a_j < h_j^{\leq} < b_j$ then $x_j^* := h_j^{\leq}$.
    If $J = Z0$ and $\sum_{j \in J} d_j \leq \alpha$ go to step 10
      else if $J = Z0$ and $\sum_{j \in J} d_j > \alpha$ go to step 9.
    Set $J := J \setminus Z0, J^{(0)} := J, n^{(0)} := n - |Z0|$, $\alpha^{(0)} := \alpha - \sum\limits_{j \in Z0} d_j$.
    Construct the sets $SVN, SV1, SV2, SV$.
    If $SVN \cup SV1 = J$ then
      if $\sum_{j \in J} d_j(a_j) \leq \alpha$ go to step 1
      else go to step 9 (feasible region $X$ is empty)
    else if $SV2 \cup SV \neq \emptyset$ then
      if $c_j(x_j) \geq 0, j \in J$, and $SVN \cup SV1 \neq \emptyset$ then
        if $\sum_{j \in SVN \cup SV1} d_j(a_j) \leq \alpha$ go to step 1
        else go to step 9 (feasible region $X$ is empty)
      else go to step 1 (feasible region $X$ is nonempty).
Step 1^1. Construct the sets $J^{0\,SVN}, J_a^{0\,SVN}, J_b^{0\,SVN}, J^{0\,SV1}, J_a^{0\,SV1}$, $J^{0\,SV2}, J_b^{0\,SV2}, J^{0\,SV}$
    (for $\lambda = 0$).
    Construct the sets $J_a^0, J_b^0, J^0$ through (33) with $\lambda = 0$. Calculate
        $$\delta(0) := \sum_{j \in J_a^0} d_j(a_j) + \sum_{j \in J_b^0} d_j(b_j) + \sum_{j \in J^0} d_j(h_j^{\leq}) - \alpha.$$
    If $\delta(0) \leq 0$ then $\lambda := 0$, go to step 8
    else if $\delta(0) > 0$ then:
      if $SV2 \cup SVN = J$ then
        if $\alpha \leq \sum_{j \in J} d_j(b_j)$ go to step 2
        else go to step 9 (there does not exist a $\lambda^* > 0$ such that $\delta(\lambda^*) = 0$)
      else if $SV1 \cup SV \neq \emptyset$ go to step 2 (there exists a $\lambda^* > 0$ such that $\delta(\lambda^*) = 0$).
Step 3^1. Construct the sets $J^{\lambda\,SVN}, J_a^{\lambda\,SVN}, J_b^{\lambda\,SVN}, J^{\lambda\,SV1}, J_a^{\lambda\,SV1}$, $J^{\lambda\,SV2}$, $J_b^{\lambda\,SV2}, J^{\lambda\,SV}$
    (with $J^{(k)}$ instead of $J$).
    Construct the sets $J_a^{\lambda(k)}, J_b^{\lambda(k)}, J^{\lambda(k)}$ by using (33) and find their cardinalities.
    Go to step 4.
```

Modifications of Algorithm 1, connected with theoretical and computational aspects, do not influence upon its computational complexity, discussed in Section 3, because these modifications do not affect the "iterative" steps of Algorithm 1.

5 Computational Experiments

In this section, we present results of some numerical experiments, obtained by applying Algorithm 1, suggested in this paper, for solving Problem (C) with particular functions

$c_j(x_j)$ and $d_j(x_j), j \in J$. The computations have been performed on an Intel Pentium IV Celeron Processor 2.66 GHz/480MB SDRAM IBM PC compatible. Each type of problems was run 30 times. Parameters and data were randomly generated in intervals where the functions $c_j(x_j)$ are strictly convex. Only because of the limitations of the interactive system used, large-scale problems of more than 1500 variables are not tested.

1.
$$c_j(x_j) = \frac{s_j}{x_j}, \quad x_j \neq 0,\ s_j > 0; \quad d_j(x_j) = d_j x_j^p, \quad d_j \geq 0, p \geq 1, x_j > 0.$$

Number of variables	n=1200	n=1500
(Average) Number of iterations	2.10	2.13
Average run time (in seconds)	0.001	0.0012

2.

$$c_j(x_j) = -s_j \ \ln\ m_j x_j, \quad s_j > 0, m_j > 0, x_j > 0; \quad d_j(x_j) = d_j x_j^p, \quad d_j \geq 0, p \geq 1, x_j > 0.$$

Number of variables	n=1200	n=1500
(Average) Number of iterations	2.07	3.03
Average run time (in seconds)	0.0012	0.0018

3.

$$c_j(x_j) = c_j x_j^q, \quad c_j > 0, q > 1, x_j \geq 0, \quad d_j(x_j) = d_j x_j^p; \quad d_j \geq 0, p > 1, x_j > 0.$$

Number of variables	n=1200	n=1500
(Average) Number of iterations	3.03	3.07
Average run time (in seconds)	0.0012	0.0019

Similarly, we can consider other strictly convex objective functions $c(\mathbf{x}) = \sum_{j \in J} c_j(x_j)$ and convex constraint functions $d_j(x_j), j \in J$.

Effectiveness of Algorithm 1 for Problem (C) has been tested by many other examples. As we can observe, the (average) number of iterations is much less than the number of variables n for large n.

References

[1] P. Berman, N. Kovoor, and P.M. Pardalos, Algorithms for the least distance problem, in: *Complexity in Numerical Optimization*, P.M. Pardalos (Ed.), World Scientific, New Jersey, 1993, pp. 33-56.

[2] D.P. Bertsekas, Projected Newton methods for optimization problems with simple constraints, *SIAM J. Control Optim.* **20** (1982) 221-246.

[3] G.R. Bitran, A.C. Hax, Disaggregation and resource allocation using convex knapsack problems with bounded variables, *Management Sci.* **27** (1981), No. 4, 431-441.

[4] J.R. Brown, Bounded knapsack sharing, *Math. Program.* **67** (1994), No. 3, 343 - 382.

[5] P. Brucker, An $\mathcal{O}(n)$ algorithm for quadratic knapsack problems, *Oper. Res. Lett.* **3** (1984), No. 3, 163-166.

[6] P.H. Calamai, J.J. Moré, Quasi-Newton updates with bounds, *SIAM J. Numer. Anal.* **24** (1987), No. 6, 1434-1441.

[7] A. Charnes, W.W. Cooper, The theory of search: optimum distribution of search effort, *Management Sci.* **5** (1958), 44-50.

[8] R.W. Cottle, S.G. Duval, K. Zikan, A Lagrangean relaxation algorithm for the constrained matrix problem, *Naval Res. Logist. Quart.* **33** (1986), No. 1, 55-76.

[9] R.S. Dembo, U. Tulowitzki, On the minimization of quadratic functions subject to box constraints, Working Paper Series B 71, School of Organization and Management, Yale University, New Haven, 1983.

[10] J.-P. Dussault, J.A. Ferland, B. Lemaire, Convex quadratic programming with one constraint and bounded variables, *Math. Program.* **36** (1986), No. 1, 90-104.

[11] J.A. Ferland, B. Lemaire, P. Robert, Analytic solutions for nonlinear programs with one or two equality constraints, Publication 285, Departement d'informatique et de recherche operationnelle, Université de Montréal, Montréal, 1978.

[12] S.M. Grzegorski, Orthogonal projections on convex sets for Newton-like methods, *SIAM J. Numer. Anal.* **22** (1985), 1208-1219.

[13] M. Held, P. Wolfe, H.P. Crowder, Validation of subgradient optimization, *Math. Program.* **6** (1974), 62-88.

[14] R. Helgason, J. Kennington, H. Lall, A polynomially bounded algorithm for a singly constrained quadratic program, *Math. Program.* **18** (1980), No. 3, 338-343.

[15] G.T.Herman, A.Lent, A family of iterative quadratic optimization algorithms for pairs of inequalities, with application in diagnostic radiology, *Math. Program. Study* **9** (1978), 15-29.

[16] N. Katoh, T. Ibaraki, H. Mine, A polynomial time algorithm for the resource allocation problem with a convex objective function, *J. Oper. Res. Soc.* **30** (1979), No. 5, 449-455.

[17] H. Luss, S.K. Gupta, Allocation of effort resources among competing activities, *Oper. Res.* **23** (1975), No. 2, 360-366.

[18] R.K. McCord, Minimization with one linear equality constraint and bounds on the variables, Technical Report SOL 79-20, System Optimization Laboratory, Department of Operations Research, Stanford University, Stanford, 1979.

[19] C. Michelot, A finite algorithm for finding the projection of a point onto the canonical simplex of $\mathbb{R}^n$, *J. Optim. Theory Appl.* **50** (1986), No. 1, 195-200.

[20] J.J. Moré, G. Toraldo, Algorithms for bound constrained quadratic programming problems, *Numer. Math.* **55** (1989), No. 4, 377-400.

[21] J.J. Moré, S.A. Vavasis, On the solution of concave knapsack problems, *Math. Program.* **49** (1991), No. 3, 397-411.

[22] P.M. Pardalos, N. Kovoor, An algorithm for a singly constrained class of quadratic programs subject to upper and lower bounds, *Math. Program.* **46** (1990), No. 3, 321-328.

[23] P.M.Pardalos, Y. Ye, C.-G. Han, Algorithms for the solution of quadratic knapsack problems, *Linear Algebra Appl.* **152** (1991), 69-91.

[24] A.G. Robinson, N. Jiang, C.S. Lerme, On the continuous quadratic knapsack problem, *Math. Program.* **55** (1992), No. 1, 99-108.

[25] R.T. Rockafellar, R.J.-B. Wets, A note about projections in the implementation of stochastic quasigradient methods, in: *Numerical Techniques for Stochastic Optimization*, Yu. Ermoliev, R.J.- B. Wets (Eds.), Springer Verlag, Berlin, 1988, pp. 385-392.

[26] S.M. Stefanov, On the implementation of stochastic quasigradient methods to some facility location problems, *Yugosl. J. Oper. Res.*, **10** (2000), No. 2, 235-256.

[27] S.M. Stefanov, Convex separable minimization subject to bounded variables, *Comput. Optim. Appl.* **18** (2001), No. 1, 27-48.

[28] S.M. Stefanov, Convex quadratic minimization subject to a linear constraint and box constraints, *Appl. Math. Res. Express* **2004** (2004), No. 1, 17-42.

[29] S.M. Stefanov, Polynomial algorithms for projecting a point onto a region defined by a linear constraint and box constraints in $\mathbb{R}^n$, *J. Appl. Math.* **2004** (2004), No. 5, 409-431.

[30] S.M. Stefanov, Convex separable minimization problems with a linear constraint and bounded variables, *Int. J. Math. Math. Sci.* **2005** (2005), No. 9, 1339-1363.

[31] S.M. Stefanov, An efficient method for minimizing a convex separable logarithmic function subject to a convex inequality constraint or linear equality constraint, *J. Appl. Math. Decis. Sci.* **2006** (2006), No. 1, Article ID 89307, 1-19.

[32] S.M. Stefanov, Minimizing a convex separable exponential function subject to linear equality constraint and bounded variables, *J. Interdiscip. Math.* **9** (2006), No. 1, 207-226.

[33] S.M. Stefanov, Minimization of a convex linear-fractional separable function subject to a convex inequality constraint or linear inequality constraint and bounds on the variables, *Appl. Math. Res. Express* **2006** (2006), No. 2, Article ID 36581, 1-24.

[34] S.M. Stefanov, Minimization of a strictly convex separable function subject to a convex inequality constraint or linear equality constraints and bounds on the variables, *Sci. Res.* **5** (2007) 1-10.

[35] S.A. Vavasis, Local minima for indefinite quadratic knapsack problems, *Math. Program.* **54** (1992), No. 2, 127-153.

[36] P. Wolfe, Algorithm for a least-distance programming problem, *Math. Program. Study* **1** (1974), 190-205.

[37] P.H. Zipkin, Simple ranking methods for allocation of one resource, *Management Sci.* **26** (1980), No. 1, 34-43.