AN EFFICIENT METHOD FOR MINIMIZING A CONVEX SEPARABLE LOGARITHMIC FUNCTION SUBJECT TO A CONVEX INEQUALITY CONSTRAINT OR LINEAR EQUALITY CONSTRAINT

STEFAN M. STEFANOV

Received 24 May 2005; Accepted 4 October 2005

We consider the problem of minimizing a convex separable logarithmic function over a region defined by a convex inequality constraint or linear equality constraint, and two-sided bounds on the variables (box constraints). Such problems are interesting from both theoretical and practical point of view because they arise in some mathematical programming problems as well as in various practical problems such as problems of production planning and scheduling, allocation of resources, decision making, facility location problems, and so forth. Polynomial algorithms are proposed for solving problems of this form and their convergence is proved. Some examples and results of numerical experiments are also presented.

1. Introduction

Consider the following problem of minimizing a convex separable logarithmic function subject to a strictly convex inequality constraint and bounded variables:
(CSL)

$$\min\left\{c(\mathbf{x}) \equiv \sum_{j\in J} c_j(x_j) \equiv \sum_{j\in J}(-s_j \ln m_j x_j)\right\} \tag{1.1}$$

subject to

$$\sum_{j\in J} d_j(x_j) \equiv \sum_{j\in J} d_j x_j^p \le \alpha, \tag{1.2}$$

$$a_j \le x_j \le b_j, \quad j \in J, \tag{1.3}$$

where $s_j > 0, m_j > 0, x_j > 0, d_j'(x_j) > 0, j \in J, p \ge 1, \mathbf{x} = (x_j)_{j\in J}$, and $J \overset{\text{def}}{=} \{1,\dots,n\}$.

Since $c''_j(x_j) = s_j/x_j^2 > 0$, then $c_j(x_j)$, $j \in J$, are convex functions defined for $x_j > 0$, $m_j > 0$, $j \in J$, and since $c'_j(x_j) = -s_j/x_j < 0$ under the assumptions, then functions $c_j(x_j)$, $j \in J$, are decreasing.

As it is known, $d_j(x_j) = d_j x_j^p$, $j \in J$, are convex functions with $d_j \geq 0$, $p \geq 1$ and $x_j \geq 0$, and strictly convex functions for $d_j > 0$, $p > 1$ and $x_j > 0$. Since $d'_j(x_j) = p d_j x_j^{p-1}$ and since $d_j \geq 0$, $p \geq 1$, then the requirement $d'_j(x_j) > 0$ becomes $d_j > 0$, $x_j > 0$, $j \in J$. In particular, since $x_j > 0$, $j \in J$, then $a_j > 0$, $b_j > 0$, $j \in J$.

Similarly, consider the problem of minimizing a convex separable logarithmic function subject to a linear equality constraint and bounded variables:
(CSLE)

$$\min\left\{c(\mathbf{x}) \equiv \sum_{j\in J} c_j(x_j) \equiv \sum_{j\in J}\Big(-s_j \ln\left(1+m_j x_j\right)\Big)\right\} \tag{1.4}$$

subject to

$$\sum_{j\in J} d_j x_j = \alpha, \tag{1.5}$$

$$a_j \leq x_j \leq b_j, \quad j \in J, \tag{1.6}$$

where $s_j > 0$, $m_j > 0$, $d_j > 0$, $x_j > -1/m_j$, $j \in J$. Using that $c''_j(x_j) = s_j m_j^2/(1 + m_j x_j)^2 > 0$, it follows that $c_j(x_j)$ are strictly convex functions. Since $c'_j(x_j) = -s_j m_j/(1 + m_j x_j) < 0$ under the assumptions, then functions $c_j(x_j)$, $j \in J$, are decreasing.

Problems (CSL) and (CSLE) are convex separable programming problems because the objective functions and constraint functions are convex (or strictly convex), and separable, that is, these functions can be expressed as the sums of single-variable functions.

It turns out that some problems, arising in production planning and scheduling, in allocation of resources [2, 6, 7, 14], in decision making [2, 7, 10, 12, 14], in the theory of search, in subgradient optimization, in facility location [10, 12, 13], and so forth, can be described mathematically by using problems like (CSL) and (CSLE), defined by (1.1)–(1.3) and (1.4)–(1.6), respectively. That is why, in order to solve such practical problems, we need some results and methods for solving (CSL) and (CSLE).

Problems like (CSL) and (CSLE) are subject of intensive study. Related problems and methods for them are considered, for example, in [1–14].

Algorithms for resource allocation problems are proposed in [2, 6, 7, 14]. Algorithms for facility location problems are suggested in [10, 12], and so forth. Singly constrained quadratic programs with bounded variables are considered in [3, 5]. Some separable programs are considered and methods for solving them are suggested in [11–13], and so forth.

Theory of nonlinear programming and, in particular, convex programming, is considered, for example, in [8, 9]. Numerical methods for solving optimization problems are widely discussed, for example, in [1, 4].

This paper is devoted to development of new efficient polynomial algorithms for solving problems (CSL) and (CSLE). The paper is organized as follows. In Section 2, characterization theorems (necessary and sufficient conditions) for the optimal solutions to the considered problems are proved. In Section 3, new algorithms of polynomial complexity are suggested and their convergence is proved. In Section 4, we consider some theoretical and numerical aspects of implementation of the algorithms and give some extensions of both characterization theorems and algorithms. In Section 5, we present results of some numerical experiments.

2. Characterization theorems

2.1. Problem (CSL). First consider problem (CSL) defined by (1.1)–(1.3).

Suppose that following assumptions are satisfied.

(1.a) $a_j \le b_j$ 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*.

(1.b) $\sum_{j\in J} d_j a_j^p \le \alpha$. Otherwise the constraints (1.2)-(1.3) are inconsistent and $X^{\le} = \emptyset$ where $X^{\le}$ is defined by (1.2)-(1.3). In addition to this assumption we suppose that $\alpha \le \sum_{j\in J} d_j b_j^p$ in some cases which are specified below.

The Lagrangian for problem (CSL) is

$$L(\mathbf{x},\mathbf{u},\mathbf{v},\lambda) = -\sum_{j\in J} s_j \ln m_j x_j + \lambda\left(\sum_{j\in J} d_j x_j^p - \alpha\right) + \sum_{j\in J} u_j(a_j - x_j) + \sum_{j\in J} v_j(x_j - b_j), \tag{2.1}$$

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) necessary and sufficient optimality conditions for the minimum solution $\mathbf{x}^* = (x_j^*)_{j\in J}$ are

$$-\frac{s_j}{x_j^*} + \lambda p d_j (x_j^*)^{p-1} - u_j + v_j = 0, \quad j \in J, \tag{2.2}$$

$$u_j(a_j - x_j^*) = 0, \quad j \in J, \tag{2.3}$$

$$v_j(x_j^* - b_j) = 0, \quad j \in J, \tag{2.4}$$

$$\lambda\left(\sum_{j\in J} d_j (x_j^*)^p - \alpha\right) = 0, \quad \lambda \in \mathbb{R}^1_+, \tag{2.5}$$

$$\sum_{j\in J} d_j (x_j^*)^p \le \alpha, \tag{2.6}$$

$$a_j \le x_j^* \le b_j, \quad j \in J, \tag{2.7}$$

$$u_j \in \mathbb{R}^1_+, \quad v_j \in \mathbb{R}^1_+, \quad j \in J, \tag{2.8}$$

where λ, u_j, v_j, $j \in J$, are the Lagrange multipliers associated with the constraints (1.2), $a_j \le x_j$, $x_j \le b_j$, $j \in J$, respectively. If $a_j = -\infty$ or $b_j = +\infty$ for some j, we do not consider the corresponding condition (2.3) [(2.4)] and Lagrange multiplier u_j [v_j, resp.]

According to conditions (2.2)–(2.8), $\lambda \geq 0$, $u_j \geq 0$, $v_j \geq 0$, $j \in J$, and complementary conditions (2.3), (2.4), (2.5) must be satisfied. In order to find x_j^*, $j \in J$, from system (2.2)–(2.8), 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^{2n+1}, where $2n+1$ is the number of all λ, u_j, v_j, $j \in J$, $|J| = n$. This is an enormous number of cases, especially for large-scale problems. For example, when $n = 1500$, we have $2^{3001} \approx 10^{900}$ cases. Moreover, in each case we have to solve a large-scale system of (nonlinear) equations in x_j^*, λ, u_j, v_j, $j \in J$. Therefore the *direct* application of 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 problems under consideration.

The following Theorem 2.1 gives a characterization of the optimal solution to problem (CSL). Its proof, of course, is based on the KKT theorem. As we will see in Section 5, by using Theorem 2.1 and the algorithm based on it, we can solve problem (CSL) with $n = 1500$ variables for a hundredth of a second on a personal computer.

Theorem 2.1 (characterization of the optimal solution to problem (CSL)). *A feasible solution $\mathbf{x}^* = (x_j^*)_{j\in J} \in X^{\leq}$, where $X^{\leq}$ is defined by (1.2)-(1.3), is the optimal solution to problem (CSL) if and only if there exists some $\lambda \in \mathbb{R}_+^1$ such that*

$$x_j^* = a_j, \quad j \in J_a^\lambda \overset{\text{def}}{=} \left\{ j \in J : \lambda \geq \frac{s_j}{pd_j a_j^p} \right\}, \tag{2.9}$$

$$x_j^* = b_j, \quad j \in J_b^\lambda \overset{\text{def}}{=} \left\{ j \in J : \lambda \leq \frac{s_j}{pd_j b_j^p} \right\}, \tag{2.10}$$

$$x_j^* = \sqrt[p]{\frac{s_j}{p\lambda d_j}}, \quad j \in J^\lambda \overset{\text{def}}{=} \left\{ j \in J : \frac{s_j}{pd_j b_j^p} < \lambda < \frac{s_j}{pd_j a_j^p} \right\}. \tag{2.11}$$

We will show below that $\lambda > 0$ strictly, so that the expressions of x_j^*, $j \in J^\lambda$, in (2.11) (especially expressions under the radical sign) are correct.

Proof

Necessity. Let $\mathbf{x}^* = (x_j^*)_{j\in J}$ be the optimal solution to (CSL). Then there exist constants λ, u_j, v_j, $j \in J$, such that KKT conditions (2.2)–(2.8) are satisfied. Consider both possible cases for λ.

(1) Let $\lambda > 0$. Then system (2.2)–(2.8) becomes (2.2), (2.3), (2.4), (2.7), (2.8) and

$$\sum_{j\in J} d_j (x_j^*)^p = \alpha, \tag{2.12}$$

that is, the inequality constraint (1.2) is satisfied with an equality for x_j^*, $j \in J$, in this case.

(a) If $x_j^* = a_j$ then $u_j \geq 0$, and $v_j = 0$ according to (2.4). Therefore (2.2) implies $-s_j/x_j^* = u_j - \lambda p d_j (x_j^*)^{p-1} \geq -\lambda p d_j (x_j^*)^{p-1}$. Since $d_j > 0, x_j^* > 0, j \in J, p \geq 1$

then

$$\lambda \geq \frac{s_j}{pd_j(x_j^*)^p} \equiv \frac{s_j}{pd_j a_j^p}. \tag{2.13}$$

(b) If $x_j^* = b_j$ then $u_j = 0$ according to (2.3), and $v_j \geq 0$. Therefore (2.2) implies $-s_j/x_j^* = -v_j - \lambda p d_j (x_j^*)^{p-1} \leq -\lambda p d_j (x_j^*)^{p-1}$. Hence

$$\lambda \leq \frac{s_j}{pd_j(x_j^*)^p} \equiv \frac{s_j}{pd_j b_j^p}. \tag{2.14}$$

(c) If $a_j < x_j^* < b_j$ then $u_j = v_j = 0$ according to (2.3) and (2.4). Therefore (2.2) implies $s_j/x_j^* = \lambda p d_j (x_j^*)^{p-1}$. Hence $\lambda = s_j/pd_j(x_j^*)^p$, and $x_j^* = \sqrt[p]{s_j/p\lambda d_j}$. Since $s_j > 0$, $d_j > 0$, $x_j^* > 0$, $j \in J$, $p \geq 1$, $\lambda > 0$ by the assumption and $b_j > x_j^*$, $x_j^* > a_j$, it follows that $\lambda = s_j/pd_j(x_j^*)^p < s_j/pd_j a_j^p$, $\lambda = s_j/pd_j(x_j^*)^p > s_j/pd_j b_j^p$, that is,

$$\frac{s_j}{pd_j b_j^p} < \lambda < \frac{s_j}{pd_j a_j^p}. \tag{2.15}$$

(2) Let $\lambda = 0$. Then system (2.2)–(2.8) becomes

$$-\frac{s_j}{x_j^*} - u_j + v_j = 0, \quad j \in J, \quad (2.3),\ (2.4),\ (2.6),\ (2.7),\ (2.8). \tag{2.16}$$

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

$$\frac{s_j}{pd_j a_j^p} \leq 0 \equiv \lambda. \tag{2.17}$$

Since $s_j > 0$, $d_j > 0$, $a_j > 0$, $j \in J$, $p \geq 1$, this case *is impossible.*

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

$$\frac{s_j}{pd_j b_j^p} \geq 0 \equiv \lambda. \tag{2.18}$$

(c) If $a_j < x_j^* < b_j$ then $u_j = v_j = 0$. Therefore $-s_j/x_j^* = 0$, and since $s_j > 0$, $x_j^* > 0$, $j \in J$, then this case *is impossible.*

To describe cases (a), (b), (c) for both (1) and (2), it is convenient to introduce the index sets J_a^λ, J_b^λ, J^λ defined by (2.9), (2.10), and (2.11), respectively. It is obvious that $J_a^\lambda \cup J_b^\lambda \cup J^\lambda = J$. The "necessity" part is proved.

Sufficiency. Conversely, let $\mathbf{x}^* \in X^{\leq}$ and components of $\mathbf{x}^*$ satisfy (2.9), (2.10), and (2.11), where $\lambda \in \mathbb{R}^1_+$.

(1) Let $\lambda > 0$. Set:

$$\lambda = \frac{s_j}{pd_j (x_j^*)^p} \quad (>0), \qquad \text{obtained from } \sum_{j\in J_a^\lambda} d_j a_j^p + \sum_{j\in J_b^\lambda} d_j b_j^p + \sum_{j\in J^\lambda} \frac{s_j}{p\lambda} = \alpha;$$

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

$$u_j = -\frac{s_j}{a_j} + \lambda p d_j a_j^{p-1} \quad (\geq 0 \text{ according to the definition of } J_a^\lambda), \quad v_j = 0 \quad \text{for } j \in J_a^\lambda;$$

$$u_j = 0, \quad v_j = \frac{s_j}{b_j} - \lambda p d_j b_j^{p-1} \quad (\geq 0 \text{ according to the definition of } J_b^\lambda) \quad \text{for } j \in J_b^\lambda. \tag{2.19}$$

By using these expressions, it is easy to check that conditions (2.2), (2.3), (2.4), (2.5), (2.8) are satisfied; conditions (2.6) and (2.7) are also satisfied according to the assumption $\mathbf{x}^* \in X^{\leq}$.

(2) If $\lambda = 0$ then $-s_j/x_j^* - u_j + v_j = 0$, $j \in J^\lambda$ according to (2.2), and

$$J^{\lambda=0} = \left\{ j : \frac{s_j}{pd_j b_j^p} < 0 < \frac{s_j}{pd_j a_j^p} \right\}. \tag{2.20}$$

Since $s_j > 0$, $d_j > 0$, $b_j > 0$, $j \in J$, $p \geq 1$, then $s_j/pd_j b_j^p > 0$, and $J^{\lambda=0} = \varnothing$. Similarly, $J_a^{\lambda=0} = \varnothing$. Therefore $J = J_b^{\lambda=0}$ for $\lambda = 0$.

Set:

$$u_j = 0, \quad v_j = \frac{s_j}{b_j} \quad (>0) \quad \text{for } j \in J = J_b^{\lambda=0}. \tag{2.21}$$

Obviously conditions (2.2), (2.3), (2.4), (2.8) are satisfied; conditions (2.6), (2.7) are also satisfied according to the assumption $\mathbf{x}^* \in X^{\leq}$, and condition (2.5) obviously is satisfied for $\lambda = 0$.

In both cases (1) and (2) of the "sufficiency" part, x_j^*, λ, u_j, v_j, $j \in J$, satisfy KKT conditions (2.2)–(2.8) 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 (CSL), and since $c(\mathbf{x})$ is a strictly convex function, then this optimal solution is unique. □

In view of the discussion above, the importance of Theorem 2.1 consists in the fact that it describes components of the optimal solution to problem (CSL) only through the Lagrange multiplier λ associated with the inequality constraint (1.2).

Since we do not know the optimal value of λ from Theorem 2.1, we define an iterative process with respect to the Lagrange multiplier λ and we prove convergence of this process in Section 3 *(The algorithms)*.

According to Theorem 2.1, $\lambda \geq 0$. In fact, since $p\lambda d_j(x_j^*)^p = s_j$, where $d_j > 0$, $x_j^* > 0$, $s_j > 0$, $j \in J$, $p \geq 1$, then $\lambda > 0$. Since $s_j/pd_ja_j^p > 0$, $s_j/pd_jb_j^p > 0$ under the assumptions, then if $\lambda := 0$ at step (2) of Algorithm 3.1 below, obviously $J^{\lambda=0} = J_a^{\lambda=0} = \varnothing$ and *it would not be necessary* to compute x_j^*, $j \in J^0$, using (2.11) for $\lambda = 0$, where λ is involved in the denominator of expression (2.11) for x_j^*. An analogous remark is also valid for problem (CSLE).

Using $d_j > 0$, $s_j > 0$, $0 < a_j \leq b_j$, $j \in J$, and $p \geq 1$, it follows that

$$ub_j \overset{\text{def}}{=} \frac{s_j}{pd_jb_j^p} \leq \frac{s_j}{pd_ja_j^p} \overset{\text{def}}{=} la_j, \quad j \in J \tag{2.22}$$

for the expressions by which we define the sets J_a^λ, J_b^λ, J^λ.

The problem how to ensure a feasible solution to problem (CSL), which is an assumption of Theorem 2.1, is discussed in Section 3.3.

2.2. Problem (CSLE).

Consider the problem of minimizing a convex separable logarithmic function subject to a linear equality constraint and box constraints (CSLE) (1.4)–(1.6).

Assumptions:

(2.a) $a_j \leq b_j$ for all $j \in J$.

(2.b) $\sum_{j\in J} d_ja_j \leq \alpha \leq \sum_{j\in J} d_jb_j$. Otherwise the constraints (1.5)-(1.6) are inconsistent and the feasible region X^L, defined by (1.5)-(1.6), is empty.

The KKT conditions for problem (CSLE) are

$$\begin{gathered}
-\frac{s_jm_j}{1+m_jx_j^*} + \lambda d_j - u_j + v_j = 0, \quad j \in J,\ \lambda \in \mathbb{R}^1, \\
u_j(a_j - x_j^*) = 0, \quad j \in J, \\
v_j(x_j^* - b_j) = 0, \quad j \in J, \\
\sum_{j\in J} d_jx_j^* = \alpha, \\
a_j \leq x_j^* \leq b_j, \quad j \in J, \\
u_j \in \mathbb{R}^1_+, \quad v_j \in \mathbb{R}^1_+, \quad j \in J.
\end{gathered} \tag{2.23}$$

In this case, the following Theorem 2.2, which is similar to Theorem 2.1, holds true.

Theorem 2.2 (characterization of the optimal solution to problem (CSLE)). *A feasible solution $\mathbf{x}^* = (x_j^*)_{j\in J} \in X^L$, where X^L is defined by (1.5)-(1.6), is the optimal solution to* *problem (CSLE) if and only if there exists some $\lambda \in \mathbb{R}^1$ such that*

$$x_j^* = a_j, \quad j \in J_a^\lambda \overset{\text{def}}{=} \left\{ j \in J : \lambda \geq \frac{s_j m_j}{d_j(1+m_j a_j)} \right\}, \tag{2.24}$$

$$x_j^* = b_j, \quad j \in J_b^\lambda \overset{\text{def}}{=} \left\{ j \in J : \lambda \leq \frac{s_j m_j}{d_j(1+m_j b_j)} \right\}, \tag{2.25}$$

$$x_j^* = \frac{s_j}{\lambda d_j} - \frac{1}{m_j}, \quad j \in J^\lambda \overset{\text{def}}{=} \left\{ j \in J : \frac{s_j m_j}{d_j(1+m_j b_j)} < \lambda < \frac{s_j m_j}{d_j(1+m_j a_j)} \right\}. \tag{2.26}$$

It can be shown that $\lambda > 0$ strictly, so that the expressions of x_j^*, $j \in J^\lambda$, in (2.26) are correct.

The proof of Theorem 2.2 is omitted because it is similar to that of Theorem 2.1.

3. The algorithms

3.1. Analysis of the optimal solution to problem (CSL). Before the formal statement of the algorithm for problem (CSL), we discuss some properties of its optimal solution.

Using (2.9), (2.10) and (2.11), condition (2.5) can be written as follows

$$\lambda \left(\sum_{j \in J_a^\lambda} d_j a_j^p + \sum_{j \in J_b^\lambda} d_j b_j^p + \frac{1}{p\lambda} \sum_{j \in J^\lambda} s_j - \alpha \right) = 0, \quad \lambda \geq 0. \tag{3.1}$$

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

$$x_j^* = x_j(\lambda) = \begin{cases} a_j, & j \in J_a^\lambda \\ b_j, & j \in J_b^\lambda \\ \sqrt[p]{\dfrac{s_j}{p\lambda d_j}}, & j \in J^\lambda. \end{cases} \tag{3.2}$$

Functions $x_j(\lambda)$, $j \in J$, are piecewise linear, monotonically nonincreasing, piecewise differentiable functions of λ with two breakpoints at $\lambda = s_j/pd_j a_j^p$ and $\lambda = s_j/pd_j b_j^p$.

Let

$$\delta(\lambda) \overset{\text{def}}{=} \sum_{j \in J_a^\lambda} d_j a_j^p + \sum_{j \in J_b^\lambda} d_j b_j^p + \frac{1}{p\lambda} \sum_{j \in J^\lambda} s_j - \alpha. \tag{3.3}$$

If we differentiate $\delta(\lambda)$ with respect to λ, we get

$$\delta'(\lambda) \equiv -\frac{1}{p\lambda^2} \sum_{j\in J^\lambda} s_j < 0, \tag{3.4}$$

when $J^\lambda \neq \varnothing$, and $\delta'(\lambda) = 0$ when $J^\lambda = \varnothing$. Hence $\delta(\lambda)$ is a *monotonically 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. Since $J^{\lambda=0} = J_a^{\lambda=0} = \varnothing$ then

$$\delta(0) = \sum_{j\in J_b^\lambda} d_j b_j^p - \alpha \tag{3.5}$$

according to (3.3), and $\delta(0) \geq 0$ in accordance with assumption (1.b). In order that (3.1) and (2.6) $\equiv$ (1.2) be satisfied, there exists some $\lambda^* > 0$ such that $\delta(\lambda^*) = 0$, that is,

$$\sum_{j\in J} d_j (x_j^*)^p = \alpha, \tag{3.6}$$

which means that the inequality constraint (1.2) is satisfied with an equality for λ^* in this case.

Case 2. The case $\delta(0) < 0$ is impossible for problem (CSL) according to above consideration.

As we have seen, for the optimal value of λ we have $\lambda \geq 0$ in all possible cases, as the KKT condition (2.5) requires. We have shown that in Case 1 we need an algorithm for finding λ^*, which satisfies the KKT conditions (2.2)–(2.8) and such that λ^* satisfies (2.6) with an equality. In order that this be fulfilled, we have required $\alpha \leq \sum_{j\in J} d_j b_j^p$ in some cases in addition to the assumption $\sum_{j\in J} d_j a_j^p \leq \alpha$ (see assumption (1.b)). We have also used this assumption in the proof of Theorem 2.1, "sufficiency" part, when $\lambda > 0$.

Using the equation $\delta(\lambda) = 0$, where $\delta(\lambda)$ is defined by (3.3), we are able to obtain a closed form expression for λ:

$$\lambda = \frac{1}{p}\left(\alpha - \sum_{j\in J_a^\lambda} d_j a_j^p - \sum_{j\in J_b^\lambda} d_j b_j^p\right)^{-1} \sum_{j\in J^\lambda} s_j, \tag{3.7}$$

because $\delta'(\lambda) < 0$ according to (3.4) when $J^\lambda \neq \varnothing$ (it is important that $\delta'(\lambda) \neq 0$). This expression of λ is used in the algorithm suggested for problem (CSL). 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 $J^\lambda \neq \varnothing$.

At iteration k of the implementation of algorithms, denote by $\lambda^{(k)}$ the value of the Lagrange multiplier associated with constraint (1.2) [(1.5), resp.], by $\alpha^{(k)}$ the right-hand side of (1.2) [(1.5), resp.]; by $J^{(k)}$, $J_a^{\lambda(k)}$, $J_b^{\lambda(k)}$, $J^{\lambda(k)}$ the current sets J, J_a^λ, J_b^λ, J^λ, respectively.

3.2. Algorithm 3.1 (for problem (CSL)). The following algorithm for solving problem (CSL) is based on Theorem 2.1, see Algorithm 3.1.

```
(1) (Initialization) $J := \{1,\dots,n\}$, $k := 0$, $\alpha^{(0)} := \alpha$, $n^{(0)} := n$, $J^{(0)} := J$, $J_a^\lambda := \emptyset$, $J_b^\lambda := \emptyset$.
    If $\sum_{j\in J} d_j a_j^p \le \alpha \le \sum_{j\in J} d_j b_j^p$, go to (2) else go to (9).
(2) $J^{\lambda(k)} := J^{(k)}$. Calculate $\lambda^{(k)}$ by using the explicit expression (3.7) of $\lambda$. Go to (3).
(3) Construct the sets $J_a^{\lambda(k)}$, $J_b^{\lambda(k)}$, $J^{\lambda(k)}$ through (2.9), (2.10), (2.11) (with $j \in J^{(k)}$
    instead of $j \in J$) and find their cardinalities $|J_a^{\lambda(k)}|$, $|J_b^{\lambda(k)}|$, $|J^{\lambda(k)}|$, respectively.
    Go to (4).
(4) Calculate
        $$\delta(\lambda^{(k)}) := \sum_{j\in J_a^{\lambda(k)}} d_j a_j^p + \sum_{j\in J_b^{\lambda(k)}} d_j b_j^p + \frac{1}{p\lambda^{(k)}} \sum_{j\in J^{\lambda(k)}} s_j - \alpha^{(k)}. \quad (3.8)$$
    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 $j \in J_a^{\lambda(k)}$, $\alpha^{(k+1)} := \alpha^{(k)} - \sum_{j\in J_a^{\lambda(k)}} d_j a_j^p$, $J^{(k+1)} := J^{(k)} \setminus J_a^{\lambda(k)}$,
    $n^{(k+1)} := n^{(k)} - |J_a^{\lambda(k)}|$, $J_a^\lambda := J_a^\lambda \cup J_a^{\lambda(k)}$, $k := k+1$. Go to (2).
(7) $x_j^* := b_j$ for $j \in J_b^{\lambda(k)}$, $\alpha^{(k+1)} := \alpha^{(k)} - \sum_{j\in J_b^{\lambda(k)}} d_j b_j^p$, $J^{(k+1)} := J^{(k)} \setminus J_b^{\lambda(k)}$,
    $n^{(k+1)} := n^{(k)} - |J_b^{\lambda(k)}|$, $J_b^\lambda := J_b^\lambda \cup J_b^{\lambda(k)}$, $k := k+1$. Go to (2).
(8) $x_j^* := a_j$ for $j \in J_a^\lambda$; $x_j^* := b_j$ for $j \in J_b^\lambda$; $x_j^* := \sqrt[p]{s_j/p\lambda d_j}$ for $j \in J^\lambda$. Go to (10).
(9) Problem (CSL) has no optimal solution because $X^{\le} = \emptyset$ or there does not exist
    a $\lambda^* > 0$ satisfying Theorem 2.1.
(10) End.
```

Algorithm 3.1. (For problem (CSL)).

3.3. Convergence and complexity of Algorithm 3.1. The following Theorem 3.1 states convergence of Algorithm 3.1.

THEOREM 3.1. *Let $\lambda^{(k)}$ be the sequence generated by Algorithm 3.1. Then*
(i) *if $\delta(\lambda^{(k)}) > 0$ then $\lambda^{(k)} \le \lambda^{(k+1)}$;*
(ii) *if $\delta(\lambda^{(k)}) < 0$ then $\lambda^{(k)} \ge \lambda^{(k+1)}$.*

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 3.1.

(i) Let $\delta(\lambda^{(k)}) > 0$. Using step (6) of Algorithm 3.1, which is performed when $\delta(\lambda^{(k)}) > 0$, we get

$$\sum_{j\in J^{\lambda(k+1)}} d_j\left(x_j^{(k)}\right)^p \equiv \sum_{j\in J^{(k+1)}} d_j\left(x_j^{(k)}\right)^p = \sum_{j\in J^{(k)}\setminus J_a^{\lambda(k)}} d_j\left(x_j^{(k)}\right)^p = \alpha^{(k)} - \sum_{j\in J_a^{\lambda(k)}} d_j\left(x_j^{(k)}\right)^p. \quad (3.9)$$

Let $j \in J_a^{\lambda(k)}$. According to definition (2.9) of $J_a^{\lambda(k)}$ we have

$$\frac{s_j}{pd_j a_j^p} \le \lambda^{(k)} = \frac{s_j}{pd_j \left(x_j^{(k)}\right)^p}. \tag{3.10}$$

Multiplying this inequality by $pd_j a_j^p / s_j > 0$ we obtain $1 \le a_j^p / (x_j^{(k)})^p$. Therefore $x_j^{(k)} \le a_j$ because $a_j > 0$, $x_j^{(k)} > 0$ and $p \ge 1$.

Using that $d_j > 0$, $a_j \ge x_j^{(k)}$, $j \in J_a^{\lambda(k)}$, and step (6), from (3.9) we get

$$\begin{aligned} \sum_{j \in J^{\lambda(k+1)}} d_j \left(x_j^{(k)}\right)^p &= \alpha^{(k)} - \sum_{j \in J_a^{\lambda(k)}} d_j \left(x_j^{(k)}\right)^p \\ &\ge \alpha^{(k)} - \sum_{j \in J_a^{\lambda(k)}} d_j a_j^p = \alpha^{(k+1)} = \sum_{j \in J^{\lambda(k+1)}} d_j \left(x_j^{(k+1)}\right)^p. \end{aligned} \tag{3.11}$$

Since $d_j > 0$, $j \in J$, then there exists at least one $j_0 \in J^{\lambda(k+1)}$ such that $x_{j_0}^{(k)} \ge x_{j_0}^{(k+1)}$. Then

$$\lambda^{(k)} = \frac{s_{j_0}}{pd_{j_0} \left(x_{j_0}^{(k)}\right)^p} \le \frac{s_{j_0}}{pd_{j_0} \left(x_{j_0}^{(k+1)}\right)^p} = \lambda^{(k+1)}. \tag{3.12}$$

We have used that the relationship between $\lambda^{(k)}$ and $x_j^{(k)}$ is given by (2.11) for $j \in J^{\lambda(k)}$ according to step (2) of Algorithm 3.1, and $\lambda^{(k)} \ge 0$, $d_j > 0$, $s_j > 0$, $j \in J$, and $p \ge 1$.

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

Consider the feasibility of $\mathbf{x}^* = (x_j^*)_{j \in J}$, generated by Algorithm 3.1.

Components $x_j^* = a_j$, $j \in J_a^\lambda$, and $x_j^* = b_j$, $j \in J_b^\lambda$, obviously satisfy (1.3). Using

$$\frac{s_j}{pd_j b_j^p} < \lambda \equiv \frac{s_j}{pd_j (x_j^*)^p} < \frac{s_j}{pd_j a_j^p}, \quad j \in J^\lambda \tag{3.13}$$

and $p \ge 1$, $d_j > 0$, $s_j > 0$, $a_j > 0$, $b_j > 0$, $j \in J$, it follows that $a_j < x_j^* < b_j$ for $j \in J^\lambda$. Hence all x_j^*, $j \in J$, satisfy (1.3).

We have proved that if $\delta(0) \ge 0$ and $X^{\le} \neq \varnothing$ where $X^{\le}$ is defined by (1.2)-(1.3), then there exists a $\lambda^* \ge 0$ such that $\delta(\lambda^*) = 0$. Since at step (2) we calculate $\lambda^{(k)}$ from the equality $\sum_{j \in J^{\lambda(k)}} d_j \left(x_j^{(k)}\right)^p = \alpha^{(k)}$ for each k, then (1.2) is satisfied with an equality in this case. Otherwise, the case $\delta(0) < 0$ is impossible (see Case 2 above).

Therefore, Algorithm 3.1 generates $\mathbf{x}^*$ which is feasible for problem (CSL), which is an assumption of Theorem 2.1.

Remark 3.2. Theorem 3.1, definitions of J_a^λ (2.9), J_b^λ (2.10) and J^λ (2.11), and steps (6), (7) and (8) of Algorithm 3.1 allow us to assert that $J_a^{\lambda(k)} \subseteq J_a^{\lambda(k+1)}$, $J_b^{\lambda(k)} \subseteq J_b^{\lambda(k+1)}$, and $J^{\lambda(k)} \supseteq J^{\lambda(k+1)}$. This means that if j belongs to current set $J_a^{\lambda(k)}$, then j belongs to the next index set $J_a^{\lambda(k+1)}$ and, therefore, to the optimal index set J_a^λ; the same holds true about the sets $J_b^{\lambda(k)}$ and J_b^λ. Therefore $\lambda^{(k)}$ converges to the optimal λ of Theorem 2.1, and $J_a^{\lambda(k)}$, $J_b^{\lambda(k)}$, $J^{\lambda(k)}$ "converge" to the optimal index sets J_a^λ, J_b^λ, J^λ, respectively. This means that 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^λ, J_b^λ, J^λ are in accordance with Theorem 2.1.

At each iteration Algorithm 3.1 determines the value of at least one variable (steps (6), (7), (8)) and at each iteration we solve a problem of the form (CSL) but of *less* dimension (steps (2), (3), (4), (5), (6), (7)). Therefore Algorithm 3.1 is finite and it converges with at most $n = |J|$ iterations, that is, the iteration complexity of Algorithm 3.1 is $\mathcal{O}(n)$.

Step (1) (initialization and checking whether $X^{\leq}$ is empty) takes time $\mathcal{O}(n)$. The calculation of $\lambda^{(k)}$ requires constant time (step (2)). Step (3) takes $\mathcal{O}(n)$ time because of the construction of $J_a^{\lambda(k)}$, $J_b^{\lambda(k)}$, $J^{\lambda(k)}$. Step (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 the final 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 3.1 has $\mathcal{O}(n^2)$ running time and it belongs to the class of strongly polynomially bounded algorithms.

Computational experiments show that the number of iterations of the algorithm performance is not only at most n, but it is much, much less than n for large n. In fact, this number does not depend on n but only on the three index sets defined by (2.9), (2.10), (2.11). In practice, Algorithm 3.1 has $\mathcal{O}(n)$ running time.

3.4. Algorithm 3.2 (for problem (CSLE)) and its convergence. After analysis of the optimal solution to problem (CSLE), similar to that to problem (CSL), we suggest the following algorithm for solving problem (CSLE).

To avoid a possible "endless loop" in programming Algorithms 3.1 and 3.2, 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 or chosen) tolerance value up to which the equality $\delta(\lambda^*) = 0$ may (for Algorithm 3.1) or must (for Algorithm 3.2) be satisfied.

A theorem analogous to Theorem 3.1 holds for Algorithm 3.2, which guarantees the "convergence" of $\lambda^{(k)}$, $J^{\lambda(k)}$, $J_a^{\lambda(k)}$, $J_b^{\lambda(k)}$ to the optimal λ, J^λ, J_a^λ, J_b^λ, respectively.

Theorem 3.3. *Let $\lambda^{(k)}$ be the sequence generated by Algorithm 3.2. Then*
(i) *if $\delta(\lambda^{(k)}) > 0$ then $\lambda^{(k)} \leq \lambda^{(k+1)}$;*
(ii) *if $\delta(\lambda^{(k)}) < 0$ then $\lambda^{(k)} \geq \lambda^{(k+1)}$.*

The proof of Theorem 3.3 is omitted because it is similar to that of Theorem 3.1.

It can be proved that Algorithm 3.2 has $\mathcal{O}(n^2)$ running time, and point $\mathbf{x}^* = (x_j^*)_{j \in J}$ generated by this algorithm is feasible for problem (CSLE), which is an assumption of Theorem 2.2.

```
(1) (Initialization) $J := \{1,\dots,n\}$, $k := 0$, $\alpha^{(0)} := \alpha$, $n^{(0)} := n$, $J^{(0)} := J$, $J_a^\lambda := \emptyset$, $J_b^\lambda := \emptyset$.
    If $\sum_{j\in J} d_j a_j \le \alpha \le \sum_{j\in J} d_j b_j$, go to (2) else go to (9).
(2) $J^{\lambda(k)} := J^{(k)}$. Calculate $\lambda^{(k)}$ by using the explicit expression
        $\lambda^{(k)} = \left(\alpha^{(k)} + \sum_{j\in J^{\lambda(k)}} \frac{d_j}{m_j}\right)^{-1} \sum_{j\in J^{\lambda(k)}} s_j.$    (3.14)
    Go to (3).
(3) Construct the sets $J_a^{\lambda(k)}$, $J_b^{\lambda(k)}$, $J^{\lambda(k)}$ through (2.24), (2.25), (2.26) (with $j \in J^{(k)}$
    instead of $j \in J$) and find their cardinalities $|J_a^{\lambda(k)}|$, $|J_b^{\lambda(k)}|$, $|J^{\lambda(k)}|$. Go to (4).
(4) Calculate
        $\delta(\lambda^{(k)}) := \sum_{j\in J_a^{\lambda(k)}} d_j a_j + \sum_{j\in J_b^{\lambda(k)}} d_j b_j + \frac{1}{\lambda^{(k)}} \sum_{j\in J^{\lambda(k)}} s_j - \sum_{j\in J^{\lambda(k)}} \frac{d_j}{m_j} - \alpha^{(k)}.$    (3.15)
    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 $j \in J_b^\lambda$; $x_j^* := s_j/\lambda d_j - 1/m_j$ for $j \in J^\lambda$. Go to (10).
(9) Problem (CSLE) has no optimal solution because the feasible set $X^L$, defined by (1.5)-(1.6), is empty.
(10) End.
```

Algorithm 3.2. (For problem (CSLE)).

4. Extensions

4.1. Theoretical aspects. Up to now we required $d_j > 0$, $j \in J$, in (1.2) and (1.5) of problems (CSL) and (CSLE), respectively. However, if it is allowed $d_j = 0$ for some j in problems (CSL) and (CSLE), then for such indices j we cannot construct the expressions $s_j/pd_ja_j^p$ and $s_j/pd_jb_j^p$ for problem (CSL), and $s_jm_j/d_j(1+m_ja_j)$ and $s_jm_j/d_j(1+m_jb_j)$ for problem (CSLE), by means of which we define sets J_a^λ, J_b^λ, J^λ for the corresponding problem. In such cases, x_j's are not involved in (1.2) [in (1.5), resp.] for such indices j.

It turns out that we can avoid this difficulty and solve problems (CSL) and (CSLE) with $d_j = 0$ for some j's.

Denote

$$Z0 = \{j \in J : d_j = 0\}. \tag{4.1}$$

Here "0" means the "computer zero." In particular, when $J = Z0$ and $\alpha = 0$, then $X^{\leq}$ (X^L, resp.) is defined only by (1.3) (by (1.6), resp.).

Theorem 4.1 (characterization of the optimal solution to problem (CSL): an extended version). *Problem (CSL) can be decomposed into two subproblems: (CSL1) for $j \in Z0$ and (CSL2) for $j \in J \setminus Z0$.*

The optimal solution to (CSL1) is

$$x_j^* = b_j, \quad j \in Z0, \tag{4.2}$$

that is, subproblem (CSL1) itself is decomposed into $n_0 \equiv |Z0|$ independent problems. The optimal solution to (CSL2) is given by (2.9), (2.10), (2.11) with $J := J \setminus Z0$.

Proof

Necessity. Let $\mathbf{x}^* = (x_j^*)_{j \in J}$ be the optimal solution to (CSL).

(1) Let $j \in Z0$, that is, $d_j = 0$ for this j. The KKT conditions are

$$-\frac{s_j}{x_j^*} - u_j + v_j = 0, j \in Z0, \qquad (2.3)\text{–}(2.8). \tag{4.3}$$

(a) If $x_j^* = a_j$, then $u_j \geq 0$, $v_j = 0$. Using (4.3), it follows that $-s_j/x_j^* = u_j \geq 0$, which is impossible because $s_j > 0$, $x_j^* > 0$.

(b) If $x_j^* = b_j$, then $u_j = 0$, $v_j \geq 0$. Therefore $-s_j/x_j^* = -v_j \leq 0$, which is always satisfied for $s_j > 0$, $x_j^* > 0$.

(c) If $a_j < x_j^* < b_j$, then $u_j = v_j = 0$. Therefore $-s_j/x_j^* = 0$, that is, $s_j = 0$, which is impossible according to the assumption $s_j > 0$.

As we have observed, only case (b) is possible for $j \in Z0$.

(2) Components of the optimal solution to (CSL2) are obtained by using the same approach as that of the proof of "necessity" part of Theorem 2.1, but with the reduced index set $J := J \setminus Z0$.

Sufficiency. Conversely, let $\mathbf{x}^* \in X^{\leq}$ and components of $\mathbf{x}^*$ satisfy: (4.2) for $j \in Z0$, and (2.9), (2.10), (2.11) for $J := J \setminus Z0$. Set:

$$u_j = 0, \quad v_j = \frac{s_j}{b_j} \quad (>0) \quad \text{for } j \in Z0. \tag{4.4}$$

If $\lambda > 0$, set:

$$\begin{aligned}
&\lambda = \frac{s_j}{pd_j(x_j^*)^p} = \lambda(\mathbf{x}^*) \quad (>0) \quad \text{from (2.11);}\\
&u_j = v_j = 0 \quad \text{for } a_j < x_j^* < b_j,\ j \in J \setminus Z0;\\
&u_j = -\frac{s_j}{a_j} + \lambda p d_j a_j^{p-1} \quad (\geq 0), \quad v_j = 0 \quad \text{for } x_j^* = a_j,\ j \in J \setminus Z0;\\
&u_j = 0, \quad v_j = \frac{s_j}{b_j} - \lambda p d_j b_j^{p-1} \quad (\geq 0) \quad \text{for } x_j^* = b_j,\ j \in J \setminus Z0.
\end{aligned} \tag{4.5}$$

If $\lambda = 0$, set:

$$u_j = 0, \quad v_j = \frac{s_j}{b_j} \quad (>0) \quad \text{for } j \in J \setminus Z0. \tag{4.6}$$

As in the proof of Theorem 2.1, $J_a^{\lambda=0} = J^{\lambda=0} = \varnothing$.

It can be verified that $\mathbf{x}^*$, λ, u_j, v_j, $j \in J$, satisfy the KKT conditions (2.2)–(2.8). Then $\mathbf{x}^*$ with components: (4.2) for $j \in Z0$, and (2.9), (2.10), (2.11) with $J := J \setminus Z0$, is the optimal solution to problem (CSL) = (CSL1) $\cup$ (CSL2). □

An analogous result holds for problem (CSLE).

Theorem 4.2 (characterization of the optimal solution to problem (CSLE): an extended version). *Problem (CSLE) can be decomposed into two subproblems: (CSLE1) for $j \in Z0$ and (CSLE2) for $j \in J \setminus Z0$.*

The optimal solution to (CSLE1) is also given by (4.2). The optimal solution to (CSLE2) is given by (2.24), (2.25), (2.26) with $J := J \setminus Z0$.

The proof of Theorem 4.2 is omitted because it repeats in part the proofs of Theorems 2.1 and 4.1.

Thus, with the use of Theorems 4.1 and 4.2 we can express components of the optimal solutions to problems (CSL) and (CSLE) *without* the necessity of constructing the expressions $s_j/pd_ja_j^p$, $s_j/pd_jb_j^p$, $s_jm_j/d_j(1+m_ja_j)$, and $s_jm_j/d_j(1+m_jb_j)$ with $d_j = 0$.

4.2. Computational aspects. Algorithms 3.1 and 3.2 are also applicable 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 step (1) of the algorithms to check whether the corresponding feasible region is empty or nonempty, and at step (3) in the expressions $s_j/pd_jx_j^p$ and $s_jm_j/d_j(1+m_jx_j)$ with $x_j = -\infty$ and/or $x_j = +\infty$, by means of which we construct sets J_a^λ, J_b^λ, J^λ, 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:

$$\begin{aligned} SVN &= \{j \in J \setminus Z0 : a_j > -\infty,\ b_j < +\infty\}, \\ SV1 &= \{j \in J \setminus Z0 : a_j > -\infty,\ b_j = +\infty\}, \\ SV2 &= \{j \in J \setminus Z0 : a_j = -\infty,\ b_j < +\infty\}, \\ SV &= \{j \in J \setminus Z0 : a_j = -\infty,\ b_j = +\infty\}. \end{aligned} \tag{4.7}$$

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.

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

In order to construct the sets J_a^λ, J_b^λ, J^λ without the necessity of calculating the values $s_j/pd_jx_j^p$ with $x_j = -\infty$ or $+\infty$ for problem (CSL), except for the sets J, $Z0$, SV, $SV1$, $SV2$, SVN, we need some subsidiary sets defined as follows.

For SVN:

$$\begin{aligned} J^{\lambda SVN} &= \left\{ j \in SVN : \frac{s_j}{pd_jb_j^p} < \lambda < \frac{s_j}{pd_ja_j^p} \right\}, \\ J_a^{\lambda SVN} &= \left\{ j \in SVN : \lambda \geq \frac{s_j}{pd_ja_j^p} \right\}, \\ J_b^{\lambda SVN} &= \left\{ j \in SVN : \lambda \leq \frac{s_j}{pd_jb_j^p} \right\}; \end{aligned} \tag{4.8}$$

for $SV1$:

$$\begin{aligned} J^{\lambda SV1} &= \left\{ j \in SV1 : \lambda < \frac{s_j}{pd_ja_j^p} \right\}, \\ J_a^{\lambda SV1} &= \left\{ j \in SV1 : \lambda \geq \frac{s_j}{pd_ja_j^p} \right\}; \end{aligned} \tag{4.9}$$

for $SV2$:

$$\begin{aligned} J^{\lambda SV2} &= \left\{ j \in SV2 : \lambda > \frac{s_j}{pd_jb_j^p} \right\}, \\ J_b^{\lambda SV2} &= \left\{ j \in SV2 : \lambda \leq \frac{s_j}{pd_jb_j^p} \right\}; \end{aligned} \tag{4.10}$$

for SV:

$$J^{\lambda SV} = SV. \tag{4.11}$$

```
Step (1)^1 (Initialization) $J := \{1,\dots,n\}$, $k := 0$, $\alpha^{(0)} := \alpha$, $n^{(0)} := n$, $J^{(0)} := J$, $J_a^\lambda := \emptyset$,
    $J_b^\lambda := \emptyset$.
    Construct the set $Z0$. If $j \in Z0$ then $x_j^* := b_j$.
    Set $J := J \setminus Z0$, $J^{(0)} := J$, $n^{(0)} := n - |Z0|$.
    Construct the sets $SVN$, $SV1$, $SV2$, $SV$.
    If $SVN \cup SV1 = J$ then
    if $\sum_{j\in J} d_j a_j^p \le \alpha$ go to step (2)
    else go to step (9) (feasible region $X^{\le}$ is empty)
    else if $SV2 \cup SV \neq \emptyset$ then
    if $SV2 \cup SVN = J$ then
    if $\alpha \le \sum_{j\in J} d_j b_j^p$ 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 (4.12) and find their cardinalities
    $|J_a^{\lambda(k)}|$, $|J_b^{\lambda(k)}|$, $|J^{\lambda(k)}|$, respectively. Go to step (4).
```

Algorithm 4.1. About Algorithm 3.1.

Then:

$$\begin{aligned} J^\lambda &:= J^{\lambda SVN} \cup J^{\lambda SV1} \cup J^{\lambda SV2} \cup J^{\lambda SV}, \\ J_a^\lambda &:= J_a^{\lambda SVN} \cup J_a^{\lambda SV1}, \\ J_b^\lambda &:= J_b^{\lambda SVN} \cup J_b^{\lambda SV2}. \end{aligned} \tag{4.12}$$

We use the sets J^λ, J_a^λ, J_b^λ, defined by (4.12), as the corresponding sets with the same names in Algorithms 3.1 and 3.2.

With the use of results of this section, steps (1) and (3) of Algorithm 3.1 can be modified as follows, respectively, see Algorithm 4.1.

Similarly, we can define subsidiary index sets of the form (4.8)–(4.11) for problem (CSLE) as well, and modify steps (1) and (3) of Algorithm 3.2.

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

5. Computational experiments

In this section, we present results of some numerical experiments, obtained by applying algorithms, suggested in this paper, to problems under consideration. The computations were performed on an Intel Pentium II Celeron Processor 466 MHz/128 MB SDRAM

Table 5.1

Problem	(CSL)		(CSLE)	
Number of variables	$n = 1200$	$n = 1500$	$n = 1200$	$n = 1500$
Average number of iterations	2.10	3.03	3.07	4.10
Average run time (in seconds)	0.06	0.067	0.00009	0.00011

IBM PC compatible. Each problem was run 30 times. Coefficients $s_j > 0$, $m_j > 0$, $d_j > 0$, $j \in J$, for problems (CSL) and (CSLE) were randomly generated, see Table 5.1.

When $n < 1200$, the run time of the algorithms is so small that the timer does not recognize the corresponding value from its computer zero. In such cases the timer displays "0 seconds."

The effectiveness of algorithms for problems (CSL) and (CSLE) 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.

We provide below the solution of two simple particular problems of the form (CLS) and (CLSE), respectively, obtained by using the approach suggested in this paper. The results are rounded to the fourth digit.

Example 5.1.

$$\min\{c(\mathbf{x}) = -\ln(2x_1) - 3\ln x_2\} \tag{5.1}$$

subject to

$$x_1^2 + 2x_2^2 \le 10, \quad 1 \le x_1 \le 3, \quad 1 \le x_2 \le 5. \tag{5.2}$$

The optimal solution, obtained by Algorithm 3.1, is

$$\begin{gathered} \mathbf{x}^* = (x_1^*, x_2^*) = (1.5811, 1.9365), \\ c_{\min} = c(\mathbf{x}^*) = -3.1339. \end{gathered} \tag{5.3}$$

Number of iterations: 1.

Example 5.2.

$$\min\{c(\mathbf{x}) = -2\ln(1+2x_1) - \ln(1+3x_2)\} \tag{5.4}$$

subject to

$$x_1 + 2x_2 = 10, \quad 1 \le x_1 \le 3, \quad 1 \le x_2 \le 5. \tag{5.5}$$

The optimal solution, obtained by Algorithm 3.2, is

$$\begin{gathered} \mathbf{x}^* = (x_1^*, x_2^*) = (3.0, 3.5), \\ c_{\min} = c(\mathbf{x}^*) = -5.2149. \end{gathered} \tag{5.6}$$

Number of iterations: 2.

6. Conclusions

In this paper, we propose an efficient method for solving convex separable minimization problems with logarithmic objective function subject to convex inequality constraint or linear equality constraint, and box constraints.

This approach could be continued and generalized for minimizing arbitrary convex separable objective functions over the same feasible regions.

Acknowledgment

The author is grateful to the editor and referees for their comments and suggestions.

References

[1] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty, *Nonlinear Programming. Theory and Algorithms*, 2nd ed., John Wiley & Sons, New York, 1993.

[2] G. R. Bitran and A. C. Hax, *Disaggregation and resource allocation using convex knapsack problems with bounded variables*, Management Science **27** (1981), no. 4, 431–441.

[3] J.-P. Dussault, J. A. Ferland, and B. Lemaire, *Convex quadratic programming with one constraint and bounded variables*, Mathematical Programming **36** (1986), no. 1, 90–104.

[4] R. Fletcher, *Practical Methods of Optimization*, 2nd ed., John Wiley & Sons, Chichester, 1987.

[5] R. Helgason, J. Kennington, and H. Lall, *A polynomially bounded algorithm for a singly constrained quadratic program*, Mathematical Programming **18** (1980), no. 3, 338–343.

[6] N. Katoh, T. Ibaraki, and H. Mine, *A polynomial time algorithm for the resource allocation problem with a convex objective function*, Journal of the Operations Research Society **30** (1979), no. 5, 449–455.

[7] H. Luss and S. K. Gupta, *Allocation of effort resources among competing activities*, Operations Research **23** (1975), no. 2, 360–366.

[8] O. L. Mangasarian, *Nonlinear Programming*, 2nd ed., Classics in Applied Mathematics, vol. 10, SIAM, Pennsylvania, 1994.

[9] R. T. Rockafellar, *Convex Analysis*, Princeton Landmarks in Mathematics, Princeton University Press, New Jersey, 1997.

[10] S. M. Stefanov, *On the implementation of stochastic quasigradient methods to some facility location problems*, Yugoslav Journal of Operations Research **10** (2000), no. 2, 235–256.

[11] ______, *Convex separable minimization subject to bounded variables*, Computational Optimization and Applications **18** (2001), no. 1, 27–48.

[12] ______, *Separable Programming. Theory and Methods*, Applied Optimization, vol. 53, Kluwer Academic Publishers, Dordrecht, 2001.

[13] ______, *Convex separable minimization problems with a linear constraint and bounded variables*, International Journal of Mathematics and Mathematical Sciences **2005** (2005), no. 9, 1339–1363.

[14] P. H. Zipkin, *Simple ranking methods for allocation of one resource*, Management Science **26** (1980), no. 1, 34–43.

Stefan M. Stefanov: Department of Mathematics, South-West University "Neofit Rilski,"
2700 Blagoevgrad, Bulgaria
E-mail address: stefm@aix.swu.bg