Optimization Letters
(accepted for publication)

On the Coupled Continuous Knapsack Problems:

Projection Onto the Volume Constrained Gibbs N-Simplex

R. Tavakoli

February 17, 2015

Abstract Coupled continuous quadratic knapsack problems (CCK) are introduced in the present study. The solution of a CCK problem is equivalent to the projection of an arbitrary point onto the volume constrained Gibbs N-simplex, which has a wide range of applications in computational science and engineering. Three algorithms have been developed in the present study to solve large scale CCK problems. According to the numerical experiments of this study, the computational costs of presented algorithms scale linearly with the problem size, when it is sufficiently large. Moreover, they show competitive or even superior computational performance compared to the advanced QP solvers. The ease of implementation and linear scaling of memory usage with the problem size are the other benefits of the presented algorithms.

Keywords knapsack problem · convex optimization · linearly constrained optimization · time-linear algorithm

Mathematics Subject Classification (2000) 65K05 · 90C20 · 90C25 · 91B32

1 Introduction

The continuous quadratic knapsack problem is defined as follows [25,26,18–20]:

$$\min_{\mathbf{x}\in\mathcal{D}_\mathcal{E}} \ \frac{1}{2}\,\mathbf{x}^T\mathbf{D}\,\mathbf{x}-\mathbf{c}^T\mathbf{x}, \quad \mathcal{D}_\mathcal{E}:=\{\mathbf{x}\in\mathbb{R}^n \mid \mathbf{a}^T\mathbf{x}=b, \quad \mathbf{l}\leqslant\mathbf{x}\leqslant\mathbf{u}\} \tag{1}$$

where $\mathbf{D}=\mathtt{diag}(\mathbf{d})$, $\mathbf{x}$, $\mathbf{d}$, $\mathbf{c}$, $\mathbf{a}$, $\mathbf{l}$, $\mathbf{u}$ are members of $\mathbb{R}^n$, $\mathbf{l}<\mathbf{u}$, possibly $l_i=-\infty$ or $u_i=\infty$, $d_i>0$, $a_i\neq 0$ and $b\in\mathbb{R}$. In our notation in the present study, matrices and vectors are denoted by bold uppercase and lowercase letters respectively, and their components are shown by non-bold letters together with the corresponding indices. The linear constraint in (1) is commonly called the resource constraint in the literature. Obviously, (1) is strictly convex and when the set of its constraints is nonempty, (1) has a unique solution. Many efficient algorithms have been developed in the literature for efficient solution of (1), for instance: breakpoint searching algorithm [25,18,19], iterative projection method [26], secant method coupled with the bracketing algorithm

R. Tavakoli
Materials Science and Engineering Department, Sharif University of Technology, Tehran, Iran, P.O. Box 11365-9466.
Tel.: +98-21-66165209
Fax: +98-21-66005717
E-mail: rtavakoli@sharif.ir

[12], variable fixing method [20], bisection method [31]. It is interesting to note that the computational costs of these algorithms scale linearly with n, i.e. they are time-linear algorithms. When $\mathbf{D}$ is equal to the identity matrix, $\mathbf{I} \in \mathbb{R}^{n\times n}$, (1) is equivalent to the orthogonal projection of $\mathbf{c}$ onto $\mathcal{D}_{\mathcal{E}}$. Because, there is no technical difference between the solution of (1) in general cases and its special case when $\mathbf{D} = \mathbf{I}$, without loss of generality, we restrict ourself to the following simplified form of (1) in the present study:

$$\min_{\mathbf{x}\in\mathcal{D}_{\mathcal{E}}} \ \frac{1}{2}\ \mathbf{x}^T\mathbf{x} - \mathbf{c}^T\mathbf{x} \tag{2}$$

The set $\mathcal{D}_{\mathcal{E}}$ in (2) can be extended to a more general form, $\mathcal{D}_{\mathcal{I}}$, as follows:

$$\min_{\mathbf{x}\in\mathcal{D}_{\mathcal{I}}} \ \frac{1}{2}\ \mathbf{x}^T\mathbf{x} - \mathbf{c}^T\mathbf{x}, \quad \mathcal{D}_{\mathcal{I}} := \{\mathbf{x}\in\mathbb{R}^n \mid b_l \leqslant \mathbf{a}^T\mathbf{x} \leqslant b_u, \quad \mathbf{l}\leqslant\mathbf{x}\leqslant\mathbf{u}\} \tag{3}$$

where $b_l, b_u \in \mathbb{R}$ and $b_l \leqslant b_u$. When $b_l = b_u$, (3) is identical to (2). We call (3) the bilateral resource constrained quadratic knapsack problem in the present study. According to our knowledge, there is no report on developing time-linear algorithms to solve (3). Because there exist many efficient time-linear algorithms to solve (2), as a part of our contribution in the present study, we shall introduce a simple theory for solving (3) in expense of solving at most two problems of the form of (2). Therefore, virtually, we shall extend all time-linear algorithms corresponding to the solution of (2), for the solution of (3).

Our major contribution in the present study is introducing a new kind of continuous quadratic knapsack problem – called the coupled system of continuous quadratic knapsack problems (CCK) here – and developing efficient algorithms to solve it. Consider $m \in \mathbb{N}$, $m \geqslant 2$ and $m \ll n$. Let us denote a set of m vectors in $\mathbb{R}^n$ by $\mathbf{X}$ (in fact unknowns of our CCK problem), i.e., $\mathbf{X} = (\mathbf{x_1}, \ldots, \mathbf{x_m})$, where $\mathbf{x_j}$ denotes the j-th vector in $\mathbf{X}$. In our notation, the i-th component of $\mathbf{x_j}$ is denoted by $x_{\mathbf{j}i}$. Using the same terminology, we define the following constants (known characteristics of our CCK problem): $s \in \mathbb{R}$, $\mathbf{b} = (b_{\mathbf{1}}, \ldots, b_{\mathbf{m}})^T$, $\mathbf{b} \in \mathbb{R}^m$, $\mathbf{C} = (\mathbf{c_1}, \ldots, \mathbf{c_m})$, $\mathbf{c_j}$ is a member of $\mathbb{R}^n$ for $\mathbf{j} = 1, \ldots, m$; $c_{\mathbf{j}i} \neq 0$. Then CCK problem is defined as follows:

$$\min_{\mathbf{X}\in\mathcal{D}} \ \frac{1}{2}\ \sum_{j=1}^{m}(\mathbf{x_j}^T\mathbf{x_j} - \mathbf{c_j}^T\mathbf{x_j}), \quad \mathcal{D} := \mathcal{B}\cap\mathcal{R}\cap\mathcal{S} \tag{4}$$

where

$$\mathcal{B} := \left\{\ \mathbf{X}\in\mathbb{R}^{n\times m} \ \mid\ \mathbf{l}\leqslant\mathbf{x_j}\leqslant\mathbf{u}, \quad \mathbf{j}=1,\ldots,m\ \right\},$$

$$\mathcal{R} := \left\{\ \mathbf{X}\in\mathbb{R}^{n\times m} \ \mid\ \mathbf{a}^T\mathbf{x_j} = b_{\mathbf{j}}, \quad \mathbf{j}=1,\ldots,m\ \right\},$$

$$\mathcal{S} := \left\{\ \mathbf{X}\in\mathbb{R}^{n\times m} \ \mid\ \sum_{\mathbf{j}=1}^{m} x_{\mathbf{j}i} = s, \quad i=1,\ldots,n\ \right\},$$

When $\mathcal{D} \neq \emptyset$, (4) has a unique solution. Note that (4) is equivalent to m distinct continuous quadratic knapsack problems which are coupled together by an n number of sparse constraints, i.e., set $\mathcal{S}$. It is worth to mention that the developed algorithms in the present study are valid under more general conditions when $\mathbf{l}$, $\mathbf{u}$ and s depend on $\mathbf{j}$. However, the simple situation (4) is considered here for the purpose of convenience. To the best of our knowledge, there is no work on developing a customized solver for the numerical solution of (4). On the other hand, (4) is a special quadratic programming (QP) problem and can be solved by a general purpose QP solver. There are several methods to solve a general QP problem. Some common methods are interior point

[22,32,8], active set [24] and augmented lagrangian [2] methods[1]. However, when n is sufficiently large, the solution of a QP problem can be very expensive in terms of the computational cost and consumed memory. If we denote the problem size by factor $m \times n$, our goal in the present study is to introduce numerical algorithms for the solution of (4), with the following properties: ease of implementation, linear scaling of computational cost as a function of the problem size and linear scaling of memory usage as a function of the problem size. Fortunately, exploiting the special structure of (4), in particular the sparsity of $\mathcal{S}$, led us to desired algorithms, such that they compete (or even outperform) state-of-the-art QP solvers in terms of the computational cost and memory usage.

Considering $\mathbf{b}_l = (b_{l\mathbf{1}}, \dots, b_{l\mathbf{m}})^T$, $\mathbf{b}_u = (b_{u\mathbf{1}}, \dots, b_{u\mathbf{m}})^T$, $\mathbf{b_l}$, $\mathbf{b_u} \in \mathbb{R}^m$ and $b_{l\mathbf{j}} \leqslant b_{u\mathbf{j}}$, the bilateral resource constrained form of knapsack system can be defined as follows:

$$\min_{\mathbf{X} \in \mathcal{I}} \ \frac{1}{2} \ \sum_{j=1}^{m} (\mathbf{x_j}^T \mathbf{x_j} - \mathbf{c_j}^T \mathbf{x_j}), \quad \mathcal{I} := \mathcal{B} \cap \mathcal{R}_{\mathcal{I}} \cap \mathcal{S} \tag{5}$$

where $\mathcal{R}_{\mathcal{I}} := \left\{ \ \mathbf{X} \in \mathbb{R}^{n \times m} \ \mid \ b_{l\mathbf{j}} \leqslant \mathbf{a}^T \mathbf{x_j} \leqslant b_{u\mathbf{j}}, \quad \mathbf{j} = 1, \dots, m \ \right\}$.
Although the focus of the present work is on the solution of (4), because there is no technical difference for extension of our algorithms to solve (5), we will comment on this extension throughout this paper.

At this point it is worth to comment on the practical importance of (4) in the computational physics and engineering fields. To do this, we will introduce two simple examples which are explained together here: Consider Ω as a square open bounded domain in $\mathbb{R}^2$. Assume that Ω is occupied by an m number of incompressible phases (or alternatively Ω is occupied by an alloy that includes an m number of components). Moreover, assume that Ω is divided into an n number of uniform elements after the spatial discretization. If we define the volume fractions of phases (the concentration of components) at center of elements, $\mathbf{X}$, there will be $n \times m$ degrees of freedom to identify the phase-distribution (alloy composition) inside Ω. Because the volume fractions (concentrations) can vary between 0 and 1, it leads to a set of bound constraints similar to $\mathcal{B}$. Moreover, there are n incompressibility constraints due to the incompressibility condition (similarly, the pointwise sum on concentration fields should be equal to the unity). This leads to another set of constraints similar to $\mathcal{S}$. The intersection of $\mathcal{B}$ and $\mathcal{S}$ is commonly called the Gibbs m-simplex or briefly the m-simplex, cf. [28,15], which has been introduced by Willard Gibbs [16] to study the thermodynamics of inhomogeneous systems. Now, assume that the phases are non-reactive. Thus, we have an additional set of constraints due to the conservation of total volume of each phase in Ω (in the case of the multi-component alloy, assuming that Ω is isolated, the total measure of each component has to be preserved). It leads to a new set of resource constraints similar to $\mathcal{R}$. Putting all together, the volume fraction (concentration) fields lie in the interior of the resource constrained Gibbs m-simplex, i.e. $\mathcal{D}$.

The solution of CCK problems has many applications in applied sciences. Some instances are: phase transition in multiphase systems [23,14], multicolor image segmentation [11,10,9], multiclass Labeling [21], multimaterial topology optimization [34,33,5, 30,29] and minimal surface space partitioning (tiling) [1,14].

Having an efficient method for the projection of an arbitrary trial point onto $\mathcal{D}$, we are able to use the projected gradient method [27] to solve the above mentioned problems. Recently, the spectral projected gradient (SPG) method [3] has been proved to be a very efficient method to find approximate solution of large scale convex constrained optimization problems, see [4] as a survey. The SPG method has been particularly recommended when a very accurate solution is not desired. It is the case in variational

[1] For a comprehensive account on bibliography and software corresponding to solution of QP problems visit: `http://www.numerical.rl.ac.uk/people/nimg/qp/qp.html`

problems in which we are looking for approximate solutions. Therefore, the presented algorithm in this study will extend the utility of SPG method to solve the above mentioned problems.

Finally, let us to define two abstract operators for the purpose of abbreviation. The operator $P_{\mathcal{X}}[\cdot]$ will be used to denote the orthogonal projection onto the convex set $\mathcal{X}$. Moreover, for $\mathbf{y}_l$, $\mathbf{y}$, $\mathbf{y}_u \in \mathbb{R}^q$ $(q \geqslant 1)$, the median operator computes $\mathbf{y}_m \in \mathbb{R}^q$ based on the following definition:

$$\mathbf{y}_m = \texttt{mid}\ (\mathbf{y}_l,\ \mathbf{y},\ \mathbf{y}_u) := \max\ \big(\mathbf{y}_l,\ \min\ (\mathbf{y}_u,\ \mathbf{y})\big)$$

where the max and min operators are to be understood componentwise here.

2 Constraint qualification

The constraint qualification is a formal requirement for the success of optimization algorithms from both the theoretical and practical points of view. In order to obtain the qualification of constraints in the case of linear constraints, it is sufficient to show that the equality constraints are linearly independent and that there exists a feasible point satisfying all inequalities strictly. However, equality constraints in $\mathcal{D}$ are not linearly independent. Assuming that $\mathcal{D}$ is nonempty, to ensure the constraint qualification, we remove $\mathbf{x_m}$ from the set of unknown vectors using equality constraints $\mathcal{S}$. Therefore, new version of problem (4) with constraint qualification will have the following form:

$$\min_{\mathbf{X}\in\mathcal{D}'}\ \frac{1}{2}\sum_{j=1}^{m-1}(\mathbf{x_j}^T\mathbf{x_j} - \mathbf{c_j}^T\mathbf{x_j}), \quad \mathcal{D}' := \mathcal{B}' \cap \mathcal{R}' \cap \mathcal{S}' \tag{6}$$

where $\mathbf{X}$, $\mathbf{C}$ and $\mathbf{b}$ are redefined as $\mathbf{X} = (\mathbf{x_1}, \ldots, \mathbf{x_{m-1}})$, $\mathbf{C} = (\mathbf{c_1}, \ldots, \mathbf{c_{m-1}})$, $\mathbf{b} = (b_\mathbf{1}, \ldots, b_{\mathbf{m-1}})^T$ and,

$$\mathcal{B}' := \big\{\ \mathbf{X} \in \mathbb{R}^{n\times(m-1)}\ \mid\ \mathbf{l} \leqslant \mathbf{x_j} \leqslant \mathbf{u},\quad \mathbf{j} = 1, \ldots, m-1\ \big\},$$

$$\mathcal{R}' := \big\{\ \mathbf{X} \in \mathbb{R}^{n\times(m-1)}\ \mid\ \mathbf{a}^T\mathbf{x_j} = b_\mathbf{j},\quad \mathbf{j} = 1, \ldots, m-1\ \big\},$$

$$\mathcal{S}' := \Big\{\ \mathbf{X} \in \mathbb{R}^{n\times(m-1)}\ \mid\ s - u_i \leqslant \sum_{\mathbf{j}=1}^{m-1} x_{\mathbf{j}i} \leqslant s - l_i,\quad i = 1, \ldots, n\ \Big\},$$

To avoid the notational complexity, we simply redefine new variables here without introducing new nomenclator. After the solution of (6), $\mathbf{x_m}$ can be computed by $\mathbf{x_m} = s - \sum_{\mathbf{j}=1}^{m-1}\mathbf{x_j}$.

It is important to note that the above mentioned problem does not exist in the case of set $\mathcal{I}$, i.e., all equality constraints in $\mathcal{I}$ are linearly independent.

3 Time linear algorithms for solution of some projection problems

Considering the structure of $\mathcal{D}'$ and $\mathcal{I}$, we have:

$$\mathcal{D}' = \mathcal{B}' \cap \mathcal{R}' \cap \mathcal{S}', \qquad \mathcal{D}' = \mathcal{B}' \cap \mathcal{D}_{\mathcal{R}'\mathcal{S}'}, \qquad \mathcal{D}' = \mathcal{S}' \cap \mathcal{D}_{\mathcal{R}'\mathcal{B}'}$$

$$\mathcal{I} = \mathcal{B} \cap \mathcal{R}_\mathcal{I} \cap \mathcal{S}, \qquad \mathcal{I} = \mathcal{B} \cap \mathcal{D}_{\mathcal{R}_\mathcal{I}\mathcal{S}}, \qquad \mathcal{I} = \mathcal{S} \cap \mathcal{D}_{\mathcal{R}_\mathcal{I}\mathcal{B}}$$

where:

$$\mathcal{D}_{\mathcal{R}'\mathcal{S}'} := \mathcal{R}' \cap \mathcal{S}', \quad \mathcal{D}_{\mathcal{R}'\mathcal{B}'} := \mathcal{R}' \cap \mathcal{B}', \quad \mathcal{D}_{\mathcal{R}_\mathcal{I}\mathcal{S}} := \mathcal{R}_\mathcal{I} \cap \mathcal{S}, \quad \mathcal{D}_{\mathcal{R}_\mathcal{I}\mathcal{B}} := \mathcal{R}_\mathcal{I} \cap \mathcal{B}$$

In this section we will introduce time-linear algorithms to project the trial point $\mathbf{X} \in \mathbb{R}^{n\times(m-1)}$ onto $\mathcal{B}'$, $\mathcal{S}'$, $\mathcal{R}'$, $\mathcal{D}_{\mathcal{R}'\mathcal{S}'}$, $\mathcal{D}_{\mathcal{R}'\mathcal{B}'}$, and the trial point $\mathbf{X} \in \mathbb{R}^{n\times m}$ onto $\mathcal{B}$, $\mathcal{S}$, $\mathcal{R}_\mathcal{I}$, $\mathcal{D}_{\mathcal{R}_\mathcal{I}\mathcal{S}}$ $\mathcal{D}_{\mathcal{R}_\mathcal{I}\mathcal{B}}$ and the trial point $\mathbf{x} \in \mathbb{R}^n$ onto $\mathcal{D}_\mathcal{E}$, $\mathcal{D}_\mathcal{I}$.

3.1 Projection onto $\mathcal{B}'$, $\mathcal{S}'$, $\mathcal{R}'$, $\mathcal{B}$, $\mathcal{S}$ and $\mathcal{R}_{\mathcal{I}}$

The optimization problems corresponding to the projection onto $\mathcal{B}'$, $\mathcal{S}'$, $\mathcal{R}'$, $\mathcal{B}$, $\mathcal{S}$ and $\mathcal{R}'_{\mathcal{I}}$ are convex separable problems with explicit solutions. By straightforward derivation, we have:

$$
\begin{aligned}
P_{\mathcal{B}}[\mathbf{X}] &= (\hat{\mathbf{x}}_{\mathbf{1}}, \ldots, \hat{\mathbf{x}}_{\mathbf{m}}), \quad \hat{\mathbf{x}}_{\mathbf{j}} = \texttt{mid}\ (\mathbf{l}_{\mathbf{j}},\ \mathbf{x}_{\mathbf{j}},\ \mathbf{u}_{\mathbf{j}}) \\
P_{\mathcal{B}'}[\mathbf{X}] &= (\hat{\mathbf{x}}_{\mathbf{1}}, \ldots, \hat{\mathbf{x}}_{\mathbf{m-1}}), \quad \hat{\mathbf{x}}_{\mathbf{j}} = \texttt{mid}\ (\mathbf{l}_{\mathbf{j}},\ \mathbf{x}_{\mathbf{j}},\ \mathbf{u}_{\mathbf{j}}) \\
P_{\mathcal{R}'}[\mathbf{X}] &= (\hat{\mathbf{x}}_{\mathbf{1}}, \ldots, \hat{\mathbf{x}}_{\mathbf{m-1}}), \quad \hat{\mathbf{x}}_{\mathbf{j}} = \mathbf{x}_{\mathbf{j}} + \mathbf{a}\ \big(b_{\mathbf{j}} - \mathbf{a}^T\mathbf{x}_{\mathbf{j}}\big)/(\mathbf{a}^T\mathbf{a}) \\
P_{\mathcal{R}_{\mathcal{I}}}[\mathbf{X}] &= (\hat{\mathbf{x}}_{\mathbf{1}}, \ldots, \hat{\mathbf{x}}_{\mathbf{m}}), \quad \hat{\mathbf{x}}_{\mathbf{j}} = \mathbf{x}_{\mathbf{j}} - \mathbf{a}\ \Big(\max(\mathbf{a}^T\mathbf{x}_{\mathbf{j}} - b_{u\mathbf{j}}, 0) + \min(\mathbf{a}^T\mathbf{x}_{\mathbf{j}} - b_{l\mathbf{j}}, 0)\Big)/(\mathbf{a}^T\mathbf{a}) \\
P_{\mathcal{S}}[\mathbf{X}] &= (\hat{\mathbf{x}}_{\mathbf{1}}, \ldots, \hat{\mathbf{x}}_{\mathbf{m}}), \quad \hat{x}_{\mathbf{j}i} = x_{\mathbf{j}i} + \big(s - \sum_{\mathbf{k}=1}^{m} x_{\mathbf{k}i}\big)/m \\
P_{\mathcal{S}'}[\mathbf{X}] &= (\hat{\mathbf{x}}_{\mathbf{1}}, \ldots, \hat{\mathbf{x}}_{\mathbf{m-1}}), \quad \hat{x}_{\mathbf{j}i} = x_{\mathbf{j}i} - \Big(\max\big(l_i - s + \sum_{\mathbf{k}=1}^{m-1} x_{\mathbf{k}i}, 0\big) + \min\big(u_i - s + \sum_{\mathbf{k}=1}^{m-1} x_{\mathbf{k}i}, 0\big)\Big)/(m-1)
\end{aligned}
$$

where $\mathbf{1} \in \mathbb{R}^n$ denotes the unity vector in $\mathbb{R}^n$. Interested readers are referred to section 8.1 of [8] for further details about the proof of above results.

3.2 Projection onto $\mathcal{D}_{\mathcal{E}}$

Proposition 1 *Assume that the feasible set $\mathcal{D}_{\mathcal{E}}$ is nonempty. Then for $\mathbf{x} \in \mathbb{R}^n$, we have*

$$P_{\mathcal{D}_{\mathcal{E}}}[\mathbf{x}] = \texttt{mid}\ (\mathbf{l},\ \mathbf{x} - \lambda^*\mathbf{a},\ \mathbf{u}) \tag{7}$$

where $\lambda^* \in \mathbb{R}$ is the unique root of the following function:

$$g(\lambda) = \mathbf{a}^T\mathbf{z}(\lambda) - b, \quad \mathbf{z}(\lambda) = \texttt{mid}\ (\mathbf{l},\ \mathbf{x} - \lambda\mathbf{a},\ \mathbf{u}) \tag{8}$$

Moreover, $g(\lambda)$ is piecewise linear and non-increasing function of λ and assumes its root in interval $[\lambda_{min}, \lambda_{max}]$, where, λ_{min} and λ_{max} are componentwise minimum and maximum of $(\mathbf{x} - \mathbf{u})/\mathbf{a}$ and $(\mathbf{x} - \mathbf{l})/\mathbf{a}$ respectively (the division operator is understood componentwise here).

Proof The proof is given in section 5 of [31].

Considering the finite precision of computer arithmetic, it is possible to find the unique root of (8), up to the machine precision by a finite numbers of steps. It is due to the fact that the length of initial searching interval is finite; it is at most equal to the difference between the maximum and minimum floating point numbers that can be stored by the machine. Thus, its length can be reduced to the machine epsilon by a finite number of bisection steps. Therefore, there exists a practically time-linear algorithms for the projection of trial points onto $\mathcal{D}_{\mathcal{E}}$.

3.3 Projection onto $\mathcal{D}_{\mathcal{I}}$

The following theorem provides an elegant approach to compute the solution of (3) using results of proposition 1. If $b_l = b_u$ then $\mathcal{D}_{\mathcal{I}} \equiv \mathcal{D}_{\mathcal{E}}$. Therefore we assume that $b_l < b_u$.

Theorem 1 *Assume that the feasible set $\mathcal{D}_{\mathcal{I}}$ is nonempty and $b_l < b_u$. Then, we have:*

$$P_{\mathcal{D}_{\mathcal{I}}}[\mathbf{x}] = \texttt{mid}\ (\mathbf{x}_L,\ \mathbf{x},\ \mathbf{x}_U), \tag{9}$$

where, $\mathbf{x}_L = (x_{L1}, \ldots, x_{Ln}) \in \mathbb{R}^n$, $\mathbf{x}_U = (x_{U1}, \ldots, x_{Un}) \in \mathbb{R}^n$,

$$x_{Li} = \begin{cases} w_i, & \textit{if} \quad a_i > 0, \\ v_i, & \textit{if} \quad a_i < 0, \end{cases} \qquad x_{Ui} = \begin{cases} v_i, & \textit{if} \quad a_i > 0 \\ w_i, & \textit{if} \quad a_i < 0 \end{cases} \tag{10}$$

and $\mathbf{v} = (v_1, \ldots, v_n) \in \mathbb{R}^n$, $\mathbf{w} = (w_1, \ldots, w_n) \in \mathbb{R}^n$, *and,*

$$\mathbf{v} := \arg\min_{\mathbf{z}}\ \frac{1}{2}\,\mathbf{z}^T\mathbf{z} - \mathbf{x}^T\mathbf{z} \quad \texttt{s.t.}: \quad \mathbf{a}^T\mathbf{z} = b_l, \quad \mathbf{l} \leqslant \mathbf{z} \leqslant \mathbf{u}, \tag{11}$$

$$\mathbf{w} := \arg\min_{\mathbf{z}}\ \frac{1}{2}\,\mathbf{z}^T\mathbf{z} - \mathbf{x}^T\mathbf{z} \quad \texttt{s.t.}: \quad \mathbf{a}^T\mathbf{z} = b_u, \quad \mathbf{l} \leqslant \mathbf{z} \leqslant \mathbf{u}, \tag{12}$$

Proof If the box constraints in (3) are treated by the projected gradient approach, the solution of (3), denoted by $\mathbf{z}^*$ here, is equal to the unique point satisfying the following optimality conditions:

$$\mathbf{z}^* = \texttt{mid}\ \big(\mathbf{l},\ \mathbf{x} - (\mu_u^* - \mu_l^*)\ \mathbf{a},\ \mathbf{u}\big) \tag{13}$$

$$b_l \leqslant \mathbf{a}^T\mathbf{z}^* \leqslant b_u \tag{14}$$

$$\mu_l^* \geqslant 0, \quad \mu_l^*(b_l - \mathbf{a}^T\mathbf{z}^*) = 0 \tag{15}$$

$$\mu_u^* \geqslant 0, \quad \mu_u^*(\mathbf{a}^T\mathbf{z}^* - b_u) = 0 \tag{16}$$

where μ_l^* and μ_u^* denote the optimal values of lagrange multipliers corresponding to the resource constraints. Optimality conditions (13)-(16) are derived by combining the optimality conditions based on the KKT conditions (cf. [24]) and the projected gradient method (cf. [17]). Now let to define the following function:

$$\mathbf{z}(\mu_l, \mu_u) = \texttt{mid}\ \big(\mathbf{l},\ \mathbf{x} - (\mu_u - \mu_l)\ \mathbf{a},\ \mathbf{u}\big) \tag{17}$$

Obviously $\mathbf{z}^* = \mathbf{z}(\mu_l^*, \mu_u^*)$. Therefore, the remaining job here is to determine optimal values of μ_l and μ_u. In general, three different situations could happen:

(i) The left hand side constraint of (14) is active at the optimal solution, thus,

$$\mathbf{a}^T\mathbf{z}^* = b_l,\ \mu_l^* \geqslant 0,\ \ \mu_u^* = 0,\ \ \mathbf{z}^* = \texttt{mid}\ \big(\mathbf{l},\ \mathbf{x} + \mu_l^*\mathbf{a},\ \mathbf{u}\big)$$

In this case the solution of (3) is identical to the solution of (2) with $b = b_l$, i.e. $\mathbf{z}^* = \mathbf{v}$.

(ii) The right hand side constraint of (14) is active at the optimal solution, thus,

$$\mathbf{a}^T\mathbf{z}^* = b_u,\ \ \mu_u^* \geqslant 0,\ \ \mu_l^* = 0,\ \ \mathbf{z}^* = \texttt{mid}\ \big(\mathbf{l},\ \mathbf{x} - \mu_u^*\mathbf{a},\ \mathbf{u}\big)$$

In this case the solution of (3) is identical to the solution of (2) with $b = b_u$, i.e. $\mathbf{z}^* = \mathbf{w}$.

(iii) Constraint (14) is inactive at the optimal solution, thus,

$$b_l < \mathbf{a}^T\mathbf{z}^* < b_u,\ \ \mu_l^* = \mu_u^* = 0,\ \ \mathbf{z}^* = \texttt{mid}\ \big(\mathbf{l},\ \mathbf{x},\ \mathbf{u}\big).$$

Considering the above results, we have:

$$\begin{cases} w_i \leqslant z_i^* \leqslant v_i, & \texttt{if} \quad a_i > 0, \\ v_i \leqslant z_i^* \leqslant w_i, & \texttt{if} \quad a_i < 0, \end{cases} \quad \overset{(10)}{\Longrightarrow} \quad \mathbf{x}_L \leqslant \mathbf{z}^* \leqslant \mathbf{x}_U \tag{18}$$

It is easy to check that whenever $\mathbf{z}^*$ satisfies (18), it lies in $\mathcal{D}_{\mathcal{I}}$. Therefore, the solution of (3) is equal to the solution of the following problem,

$$\min_{\mathbf{z}}\ \frac{1}{2}\,\mathbf{x}^T\mathbf{x} - \mathbf{x}^T\mathbf{z} \quad \texttt{s.t.}: \quad \mathbf{x}_L \leqslant \mathbf{z} \leqslant \mathbf{x}_U \tag{19}$$

Having $\mathbf{x}_L$ and $\mathbf{x}_U$, the solution of (19) can be computed as follows: $\mathbf{z}^* = \texttt{mid}\ (\mathbf{x}_L,\ \mathbf{x},\ \mathbf{x}_U)$.

3.4 Projection onto $\mathcal{D}_{\mathcal{R}_\mathcal{I}\mathcal{B}}$ and $\mathcal{D}_{\mathcal{R}'\mathcal{B}'}$

The optimization problem corresponding to the projection of a trial point $\mathbf{X} \in \mathbb{R}^{n\times m}$ onto $\mathcal{D}_{\mathcal{R}_\mathcal{I}\mathcal{B}}$, is equivalent to m independent projection problems with a structure similar to the projection onto $\mathcal{D}_\mathcal{I}$:

$$P_{\mathcal{D}_{\mathcal{R}_\mathcal{I}\mathcal{B}}}[\mathbf{X}] = (\hat{\mathbf{x}}_\mathbf{1}, \ldots, \hat{\mathbf{x}}_\mathbf{m}), \quad \hat{\mathbf{x}}_\mathbf{j} = P_{\mathcal{D}_{\mathcal{I}\mathbf{j}}}[\mathbf{x}_\mathbf{j}] \tag{20}$$

where, $\mathcal{D}_{\mathcal{I}\mathbf{j}} := \{\ \mathbf{z} \in \mathbb{R}^n \mid b_{l\mathbf{j}} \leqslant \mathbf{a}^T\mathbf{z} \leqslant b_{u\mathbf{j}}, \quad \mathbf{l} \leqslant \mathbf{z} \leqslant \mathbf{u}\ \}$. Similarly, the optimization problem corresponding to the projection of trial point $\mathbf{X} \in \mathbb{R}^{n\times(m-1)}$ onto $\mathcal{D}_{\mathcal{R}'\mathcal{B}'}$, is equivalent to $m-1$ independent projection problems with a structure similar to the projection onto $\mathcal{D}_\mathcal{E}$:

$$P_{\mathcal{D}_{\mathcal{R}'\mathcal{B}'}}[\mathbf{X}] = (\hat{\mathbf{x}}_\mathbf{1}, \ldots, \hat{\mathbf{x}}_{\mathbf{m-1}}), \quad \hat{\mathbf{x}}_\mathbf{j} = P_{\mathcal{D}_{\mathcal{E}\mathbf{j}}}[\mathbf{x}_\mathbf{j}] \tag{21}$$

where, $\mathcal{D}_{\mathcal{E}\mathbf{j}} := \{\ \mathbf{z} \in \mathbb{R}^n \mid \mathbf{a}^T\mathbf{z} = b_\mathbf{j}, \quad \mathbf{l} \leqslant \mathbf{z} \leqslant \mathbf{u}\ \}$.

3.5 Projection onto $\mathcal{D}_{\mathcal{R}'\mathcal{S}'}$ and $\mathcal{D}_{\mathcal{R}_\mathcal{I}\mathcal{S}}$

The optimization problem corresponding to the projection of the trial point $\mathbf{X} \in \mathbb{R}^{n\times(m-1)}$ onto $\mathcal{D}_{\mathcal{R}'\mathcal{S}'}$ does not have an explicit solution. However, as it will be shown here, we will compute its semi-explicit solution in expense of solving an $(m-1)\times(m-1)$ system of non-linear equations. Assuming $\mathcal{D}_{\mathcal{R}'\mathcal{S}'} \neq \emptyset$, the projection of the trial point $\mathbf{X} \in \mathbb{R}^{n\times(m-1)}$ onto $\mathcal{D}_{\mathcal{R}'\mathcal{S}'}$, denoted by $\mathbf{Z}^* = (\mathbf{z}^*_\mathbf{1}, \ldots, \mathbf{z}^*_{\mathbf{m-1}}) \in \mathbb{R}^{n\times(m-1)}$, is equivalent to finding the unique stationary point, denoted by $(\mathbf{Z}^*, \lambda_l^*, \lambda_u^*, \mu^*) \in (\mathbb{R}^{n\times(m-1)}, \mathbb{R}^n, \mathbb{R}^n, \mathbb{R}^{m-1})$, of the following augmented lagrangian,

$$\mathcal{L}(\mathbf{Z}, \lambda_l, \lambda_u, \mu) = \frac{1}{2}\sum_{j=1}^{m-1}(\mathbf{z_j}^T\mathbf{z_j} - \mathbf{x_j}^T\mathbf{z_j}) + \lambda_u^T\Big(\sum_{\mathbf{j}=1}^{m-1}\mathbf{z_j} - \mathbf{s}_u\Big) + \lambda_l^T\Big(\mathbf{s}_l - \sum_{\mathbf{j}=1}^{m-1}\mathbf{z_j}\Big) + \sum_{\mathbf{j}=1}^{m-1}\mu_\mathbf{j}(\mathbf{a}^T\mathbf{z_j} - b_\mathbf{j})$$

where $\mathbf{s}_u, \mathbf{s}_l \in \mathbb{R}^n$, $s_{ui} = s - l_i$, $s_{li} = s - u_i$ and $\lambda_u, \lambda_l \in \mathbb{R}^n$, $\mu \in \mathbb{R}^{m-1}$ are lagrange multipliers corresponding to inequality and resource constraints. Using the first order KKT necessary optimality conditions results in:

$$\mathbf{z}^*_\mathbf{j} - \mathbf{x}_\mathbf{j} + \lambda_u^* - \lambda_l^* + \mu^*_\mathbf{j}\mathbf{a} = 0, \quad \mathbf{j} = 1, \ldots, m-1 \tag{22}$$

$$\mathbf{a}^T\mathbf{z}^*_\mathbf{j} - b_\mathbf{j} = 0, \quad \mathbf{j} = 1, \ldots, m-1 \tag{23}$$

$$\lambda^*_{ui} \geqslant 0, \quad \lambda^*_{ui}\Big(\sum_{\mathbf{j}=1}^{m-1} z^*_{\mathbf{j}i} - s_{ui}\Big) = 0, \quad \mathbf{i} = 1, \ldots, n \tag{24}$$

$$\lambda^*_{li} \geqslant 0, \quad \lambda^*_{li}\Big(s_{li} - \sum_{\mathbf{j}=1}^{m-1} z^*_{\mathbf{j}i}\Big) = 0, \quad \mathbf{i} = 1, \ldots, n \tag{25}$$

$$\sum_{\mathbf{j}=1}^{m-1} z^*_{\mathbf{j}i} - s_{ui} \leqslant 0, \quad \mathbf{i} = 1, \ldots, n \tag{26}$$

$$s_{li} - \sum_{\mathbf{j}=1}^{m-1} z^*_{\mathbf{j}i} \leqslant 0, \quad \mathbf{i} = 1, \ldots, n \tag{27}$$

Computing $\mathbf{z}^*_\mathbf{j}$ from (22) results in:

$$z^*_{\mathbf{j}i} = x_{\mathbf{j}i} - (\lambda^*_{ui} - \lambda^*_{li}) - \mu^*_\mathbf{j} a_i \tag{28}$$

Considering set $\mathcal{S}'$, for every index i in (26) and (27), there exists three situations for every :

(i) Both of constraints (26) and (27) are inactive. In this case $\lambda^*_{li} = \lambda^*_{ui} = 0$.

(ii) Constraint (26) is active and constraint (27) is inactive. In this case $\lambda^*_{li} = 0$, $\lambda^*_{ui} > 0$ and $\sum_{\mathbf{j}=1}^{m-1} z^*_{\mathbf{j}i} = s_{ui}$. Substituting (28) into this equality constraint results in:

$$\lambda^*_{ui} = \frac{1}{m-1}\left(\sum_{\mathbf{j}=1}^{m-1} x_{\mathbf{j}i} - a_i \sum_{\mathbf{j}=1}^{m-1} \mu^*_{\mathbf{j}} - s_{ui}\right) > 0 \tag{29}$$

(iii) Constraint (27) is active and constraint (26) is inactive. In this case $\lambda^*_{ui} = 0$, $\lambda^*_{li} > 0$ and $\sum_{\mathbf{j}=1}^{m-1} z^*_{\mathbf{j}i} = s_{li}$. Substituting (28) into this equality constraint results in:

$$\lambda^*_{li} = \frac{-1}{m-1}\left(\sum_{\mathbf{j}=1}^{m-1} x_{\mathbf{j}i} - a_i \sum_{\mathbf{j}=1}^{m-1} \mu^*_{\mathbf{j}} - s_{li}\right) > 0 \tag{30}$$

Considering $s_{ui} = s - l_i$, $s_{li} = s - u_i$, we always have

$$\left(\sum_{\mathbf{j}=1}^{m-1} x_{\mathbf{j}i} - a_i \sum_{\mathbf{j}=1}^{m-1} \mu^*_{\mathbf{j}} - s_{li}\right) \geqslant \left(\sum_{\mathbf{j}=1}^{m-1} x_{\mathbf{j}i} - a_i \sum_{\mathbf{j}=1}^{m-1} \mu^*_{\mathbf{j}} - s_{ui}\right)$$

Therefore, by (29) and (30), we have:

$$\lambda^*_u - \lambda^*_l = \frac{1}{m-1}\Bigg[\max\left(\sum_{\mathbf{j}=1}^{m-1} \mathbf{x_j} - \mathbf{a}\sum_{\mathbf{j}=1}^{m-1} \mu^*_{\mathbf{j}} - \mathbf{s}_u, 0\right) + \min\left(\sum_{\mathbf{j}=1}^{m-1} \mathbf{x_j} - \mathbf{a}\sum_{\mathbf{j}=1}^{m-1} \mu^*_{\mathbf{j}} - \mathbf{s}_l, 0\right)\Bigg] \tag{31}$$

Substituting (31) into (28), results in:

$$\mathbf{z}^*_{\mathbf{j}} = \mathbf{x_j} - \mu^*_{\mathbf{j}}\mathbf{a} - \frac{1}{m-1}\Bigg[\max\left(\sum_{\mathbf{j}=1}^{m-1} \mathbf{x_j} - \mathbf{a}\sum_{\mathbf{j}=1}^{m-1} \mu^*_{\mathbf{j}} - \mathbf{s}_u, 0\right) + \min\left(\sum_{\mathbf{j}=1}^{m-1} \mathbf{x_j} - \mathbf{a}\sum_{\mathbf{j}=1}^{m-1} \mu^*_{\mathbf{j}} - \mathbf{s}_l, 0\right)\Bigg] \tag{32}$$

for $\mathbf{j} = 1, \ldots, m-1$. Substituting (32) into (23) leads to the following $(m-1) \times (m-1)$ system of equations:

$$\mu^*_{\mathbf{j}}(\mathbf{a}^T\mathbf{a}) + \frac{\mathbf{a}^T}{m-1}\Bigg[\max\left(\sum_{\mathbf{j}=1}^{m-1} \mathbf{x_j} - \mathbf{a}\sum_{\mathbf{j}=1}^{m-1} \mu^*_{\mathbf{j}} - \mathbf{s}_u, 0\right) + \min\left(\sum_{\mathbf{j}=1}^{m-1} \mathbf{x_j} - \mathbf{a}\sum_{\mathbf{j}=1}^{m-1} \mu^*_{\mathbf{j}} - \mathbf{s}_l, 0\right)\Bigg] = \mathbf{a}^T\mathbf{x_j} - b_{\mathbf{j}} \tag{33}$$

for $\mathbf{j} = 1, \ldots, m-1$. Because, the optimization problem corresponding to the projection onto $\mathcal{D}_{\mathcal{R}'\mathcal{S}'}$ has a unique solution, the stationary point of the augmented lagrangian $\mathcal{L}$ is unique. Therefore, there is a unique solution for the above system of nonlinear equations. Although this system can be solved by modern methods, e.g. semi-smooth Newton, the simple Picard linearization approach is used in the present study. The number of

nonlinear iterations to solve this system of equations depends on m. Therefore, we expect that the computational cost of projection onto $\mathcal{D}_{\mathcal{R}'\mathcal{S}'}$ scales linearly with n when $m \ll n$.

Assuming $\mathcal{D}_{\mathcal{R}_{\mathcal{I}}\mathcal{B}}$ is nonempty, the projection of a trial point $\mathbf{X} \in \mathbb{R}^{n\times m}$ onto $\mathcal{D}_{\mathcal{R}_{\mathcal{I}}\mathcal{B}}$ can be computed in a similar way as discussed for $\mathcal{D}_{\mathcal{R}'\mathcal{S}'}$. However, the details are not discussed here for the purpose of brevity.

4 Projection onto $\mathcal{D}'$ and $\mathcal{I}$

Having efficient time-linear algorithms for the projection of trial points onto $\mathcal{B}'$, $\mathcal{S}'$, $\mathcal{R}'$, $\mathcal{D}_{\mathcal{R}'\mathcal{B}'}$ and $\mathcal{D}_{\mathcal{R}'\mathcal{S}'}$ sets, we will introduce three efficient algorithms to find the projection of trial points onto $\mathcal{D}'$. Because the projection onto $\mathcal{I}$ can be performed by a similar way, it will not be discussed here for the purpose of brevity. Our algorithms here are based on the alternating projection method [6,13]. In spite of using such a simple method, the actual computational efficiency is very surprising. Although we are not able to prove that the computational costs of our algorithms scale linearly with n (i.e., time-linear algorithms), our numerical experiments support this claim when n is sufficiently large. Such an observation is not strange, because recent studies illustrate that elementary splitting approaches exhibit good computational efficiencies for certain classes of large scale practical optimization problems, see [7] as a survey.

Assume that $\varepsilon > 0$ denotes the convergence threshold, k_{max} denotes the maximum number of iterations and $\mathbf{X}^0 \in \mathbb{R}^{n\times(m-1)}$ is a trial point. The following algorithms perform the projection of $\mathbf{X}^0$ onto $\mathcal{D}'$.

Algorithm 1: Alternating projections version #1

Given $\varepsilon > 0$, k_{max}, $\mathcal{S}'$, $\mathcal{R}'$, $\mathcal{S}'$, $\mathbf{X}^0$. Set $k \leftarrow 0$

repeat

  $\mathbf{X}^{k+1} = P_{\mathcal{S}'}\left[P_{\mathcal{R}'}\left[P_{\mathcal{B}'}\left[\mathbf{X}^k\right]\right]\right]$, Set $k \leftarrow k+1$

until $\|\mathbf{X}^{k+1} - \mathbf{X}^k\|_\infty > \varepsilon$ `and` $k < k_{max}$;

Algorithm 2: Alternating projections version #2

Given $\varepsilon > 0$, k_{max}, $\mathcal{S}'$, $\mathcal{D}_{\mathcal{R}'\mathcal{B}'}$, $\mathbf{X}^0$. Set $k \leftarrow 0$

repeat

  $\mathbf{X}^{k+1} = P_{\mathcal{S}'}\left[P_{\mathcal{D}_{\mathcal{R}'\mathcal{B}'}}\left[\mathbf{X}^k\right]\right]$, Set $k \leftarrow k+1$

until $\|\mathbf{X}^{k+1} - \mathbf{X}^k\|_\infty > \varepsilon$ `and` $k < k_{max}$;

Algorithm 3: Alternating projections version #3

Given $\varepsilon > 0$, k_{max}, $\mathcal{B}'$, $\mathcal{D}_{\mathcal{R}'\mathcal{S}'}$, $\mathbf{X}^0$. Set $k \leftarrow 0$

repeat

  $\mathbf{X}^{k+1} = P_{\mathcal{D}_{\mathcal{R}'\mathcal{S}'}}\left[P_{\mathcal{B}'}\left[\mathbf{X}^k\right]\right]$, Set $k \leftarrow k+1$

until $\|\mathbf{X}^{k+1} - \mathbf{X}^k\|_\infty > \varepsilon$ `and` $k < k_{max}$;

Proposition 2 *(Convergence theory of algorithms #1, #2 and #3) Assume that $\mathcal{D}'$ is nonempty. Algorithms #1, #2 and #3 are well defined and by starting from an arbitrary $\mathbf{X}^0 \in \mathbb{R}^{n\times(m-1)}$ they converge to the unique minimizer of* (6).

Proof The proof is directly followed from the convergence theory of the classical alternating projections algorithm. It is well documented in the literature, for instance see [6, 13].

Prior to closing this section, it is worth to comment on the memory requirement of the presented algorithms. For the sake of convenience, we ignore required memory related to non-vector valued variables (like loop counters). If numbers are stored in double precision format, i.e. using 64 bits to store each number, $\big((m+3)\times n+m\big)\times 64$ bits are required to store the original problem. In addition to this memory, algorithms #1-#3 respectively need $(m\times n)\times 64$, $(m\times n)\times 64$ and $\big((m\times n)+2m\big)\times 64$ bits to solve the desired problem. Therefore, if we denote the problem size by $m\times n$, the memory requirement of algorithms #1-#3 scales linearly with the problem size.

5 Numerical results

In this section we will study the success and performance of the presented algorithms by solving a model problem. The selection of this model problem is due its practical applications, as it is mentioned in section 1. To perform numerical experiments, the presented algorithms are implemented in MATLAB language. A personal computer with an Intel 2.93 GHz CPU and 4 GB RAM, with 64 bit Ubuntu Linux operating system is used as the computational resource in the present study.

The details of model problem used here are as follows: $s=1$, $\mathbf{l}=\mathbf{0}\in\mathbb{R}^n$, $\mathbf{a},\mathbf{u}=\mathbf{1}\in\mathbb{R}^n$, $\mathbf{b}=\frac{\mathbf{a}^T\mathbf{1}}{m}\mathbf{1}\in\mathbb{R}^m$. The components of vector $\mathbf{c_j}\in\mathbb{R}^n$ (for $\mathbf{j}=1,\ldots,m-1$) are selected by generating random numbers with uniform distribution in $[0,1]$, using MATLAB function `rand`. This model problem was solved for $m=4,8,12$ and $n=10^3$, 2×10^3, 4×10^3, 6×10^3, 8×10^3, 10^4, 2×10^4, 4×10^4, 6×10^4, 8×10^4, 10^5, 2×10^5, 4×10^5, 6×10^5, 8×10^5 and 10^6.

To compare our results to available advanced QP/SOCP solvers, we have examined four QP/SOCP solvers which are available through CVX[2] interface. These solvers are SeDuMi[3], SDPT3[4], Gurobi [5] and MOSEK [6]. According to our numerical experiences, the MOSEK was superior to other mentioned alternatives to solve our model problem, in terms of the computational cost. In general, it is possible to reformulate a QP problem as a second-order cone program (SOCP) problem and solve it by a SOCP solver. It is well known that for some problems, this new reformulation significantly reduces the computational cost. However, our numerical experiments by MOSEK showed that there is no benefit to reformulate our original QP problem as a SOCP problem. Therefore, our model problem is solved (in its original QP structure) by MOSEK for the purpose of comparison in this section. In all cases, the problem is solved so that the maximum possible accuracy is attained (considering IEEE-standard based double precision storage and arithmetic). For this purpose, we apply no bound on the maximum number of iterations in our algorithms, i.e. $k_{max}=\infty$, and terminate iterations whenever an oscillatory behavior is observed, i.e. when $\|\mathbf{X}^{k+1}-\mathbf{X}^k\|_\infty > \|\mathbf{X}^k-\mathbf{X}^{k-1}\|_\infty$ in line 4 of algorithms #1, #2 and #3. In the case of MOSEK solver, its convergence thresholds

[2] `http://cvxr.com/cvx/`

[3] `http://sedumi.ie.lehigh.edu/`

[4] `http://www.math.nus.edu.sg/~mattohkc/sdpt3.html`

[5] `http://gurobi.com/`

[6] `http://mosek.com/`

are adjusted accordingly to attain the maximum attainable accuracy. The presented results here include number of iterations to meet the convergence[7] (`iter`), L_1-norm of constraints infeasibility ($\mathtt{L_1 - infeasibility}$), L_2-norm distance between $\mathbf{C}$ and its projection point ($\mathtt{L_2 - distance}$) and the CPU-time. To reduce the sensitivity of the results to random character of generated test problems, for every pair (m, n), the model problem is solved 10 times with different randomly generated $\mathbf{C}$, and the averaged results are reported here.

Tables 1-3 show results of these numerical experiments. Because MOSEK encounters the memory limitation on our machine when $n \geqslant 600000$ and $m = 8, 12$, the corresponding columns are empty in the tables 2 and 3. According to tables 1-3 all solvers are successful to solve the model problem in a reasonable time. In all cases, the L_1-norm of constraints infeasibility increases by the problem size ($m \times n$). This observation can be easily justified by considering the following facts. At the best conditions, we are able to solve the problem up to the machine precision. Therefore, the infeasibility of every constraint will be near the machine precision at the best conditions. Moreover, the number of existing constraints in the problem increases with the problem size. Considering the L_1-norm of constraints infeasibility sums the absolute infeasibleness of all constraints, we expect that the L_1-norm of constraints infeasibility increases with the problem size. According to the tables, the L_1-norm constraints infeasibility of all solvers are almost in the same order and it is not possible to rank them based on this measure. It is however important to note that such level of infeasibility is quite sufficient in many practical applications.

According to tables 1-3, algorithm #1 shows the best computational performance among other solvers. In particular, it is an order of magnitude faster than others. On the other hand, algorithm #2 exhibits the worst computational efficiency among others. Moreover, the computational performance of algorithm #3 and MOSEK is competitive. The excellent computational performance of algorithm #1 stems on the fact that it is fully explicit, i.e. has no iterative phase per iteration. On the other hand, algorithms #2 and #3 have an internal iterative phase per iteration. During each iteration of algorithm #2 we have to solve an $m - 1$ number of root finding problems which is iterative and relatively expensive procedure. Similarly, every iteration of algorithm #3 includes the solution of an $(m - 1) \times (m - 1)$ system of nonlinear equations.

The graphs of cpu-time as a function of n are plotted in figures 1, 2 and 3 for m = 4, 8 and 12 respectively. According to the plots, for all solvers, the cpu-time scales linearly when n is sufficiently large. For small n the slope of curves varies slightly with n in some solvers. However, the computational complexity is much better than $\mathrm{O}(n^2)$ in these cases. It is worth noting that the mentioned deviation was not observed in the case of algorithm #1, i.e. it behaves linearly in all spectrum of n. According to tables 1-3, the number of iterations is independent of n. This observation confirms the time-linear complexity of the presented algorithms, if it is assumed that the computational cost of every iteration scales linearly with n. This condition is valid in the case of algorithm #1. Because algorithms #2 and #3 include an internal iterative phase per iteration, the mentioned condition is valid for them if the number of internal iterations is sufficiently small and independent of n. The later conditions hold in practice according to our numerical experiments.

[7] more precisely, maximum attainable accuracy by the solver

Table 1 Results of numerical experiments for $m = 4$.

solver	n	iter	$L_1 - \texttt{infeasibility}$	$L_2 - \texttt{distance}$	$\texttt{cpu} - \texttt{time}$ (sec)
Algorithm #1	1000	25	3.524292e-13	5.662540e+00	0.005174
Algorithm #2	1000	34	3.697709e-13	5.662552e+00	0.254674
Algorithm #3	1000	39	4.382494e-13	5.473170e+00	0.033794
MOSEK	1000	10	7.133849e-13	5.436209e+00	1.684905
Algorithm #1	2000	25	1.392664e-12	6.712955e+00	0.007412
Algorithm #2	2000	35	1.860534e-12	6.712982e+00	0.310595
Algorithm #3	2000	38	2.067213e-12	6.484524e+00	0.046050
MOSEK	2000	10	2.228262e-12	6.438772e+00	1.549418
Algorithm #1	4000	25	2.864908e-12	7.979499e+00	0.018856
Algorithm #2	4000	35	3.002532e-12	7.979506e+00	0.547246
Algorithm #3	4000	37	3.054268e-12	7.709964e+00	0.108535
MOSEK	4000	10	7.094059e-12	7.656304e+00	1.517394
Algorithm #1	6000	25	4.297362e-12	8.823067e+00	0.026109
Algorithm #2	6000	35	6.210454e-12	8.823052e+00	0.643931
Algorithm #3	6000	37	8.528533e-12	8.523505e+00	0.131719
MOSEK	6000	10	1.036824e-11	8.464621e+00	1.570011
Algorithm #1	8000	25	8.458301e-12	9.483905e+00	0.035769
Algorithm #2	8000	35	2.497851e-11	9.483875e+00	0.751230
Algorithm #3	8000	36	1.431182e-11	9.164546e+00	0.161914
MOSEK	8000	10	1.587068e-11	9.100534e+00	1.630091
Algorithm #1	10000	25	1.259650e-11	1.003652e+01	0.042513
Algorithm #2	10000	34	2.006197e-11	1.003651e+01	0.849355
Algorithm #3	10000	36	1.679976e-11	9.696271e+00	0.186495
MOSEK	10000	10	2.678462e-11	9.629119e+00	1.699614
Algorithm #1	20000	25	4.920366e-11	1.194138e+01	0.079828
Algorithm #2	20000	35	9.817045e-11	1.194136e+01	1.362745
Algorithm #3	20000	36	6.665790e-11	1.153743e+01	0.322093
MOSEK	20000	10	7.439667e-11	1.145713e+01	2.173682
Algorithm #1	40000	25	1.311491e-10	1.418664e+01	0.158901
Algorithm #2	40000	35	2.807273e-10	1.418662e+01	1.828441
Algorithm #3	40000	36	1.865544e-10	1.370676e+01	0.626777
MOSEK	40000	10	2.050001e-10	1.361170e+01	3.323780
Algorithm #1	60000	26	5.913535e-10	1.570661e+01	0.244017
Algorithm #2	60000	34	1.095081e-09	1.570659e+01	2.980707
Algorithm #3	60000	36	1.041583e-09	1.517501e+01	1.012986
MOSEK	60000	10	3.079549e-10	1.506956e+01	4.487450
Algorithm #1	80000	25	5.438778e-10	1.687916e+01	0.329332
Algorithm #2	80000	35	1.399945e-09	1.687914e+01	4.930427
Algorithm #3	80000	36	9.158572e-10	1.630964e+01	1.457711
MOSEK	80000	11	4.714821e-10	1.619628e+01	5.829447
Algorithm #1	100000	25	1.163062e-09	1.784474e+01	0.427192
Algorithm #2	100000	35	1.807087e-09	1.784472e+01	7.228820
Algorithm #3	100000	35	1.766217e-09	1.724241e+01	2.285473
MOSEK	100000	11	9.167707e-10	1.712251e+01	7.008362
Algorithm #1	200000	25	5.280890e-09	2.121610e+01	0.914566
Algorithm #2	200000	35	8.107730e-09	2.121606e+01	17.974060
Algorithm #3	200000	35	9.547209e-09	2.049961e+01	5.494565
MOSEK	200000	11	2.297747e-09	2.035744e+01	12.985550
Algorithm #1	400000	25	2.425804e-08	2.522456e+01	1.917146
Algorithm #2	400000	36	4.211352e-08	2.522453e+01	35.916850
Algorithm #3	400000	35	3.387068e-08	2.437239e+01	12.129400
MOSEK	400000	11	8.098141e-09	2.420320e+01	24.998900
Algorithm #1	600000	25	3.591704e-08	2.792316e+01	2.842135
Algorithm #2	600000	35	9.916932e-08	2.792312e+01	52.418450
Algorithm #3	600000	34	5.673730e-08	2.697961e+01	17.948020
MOSEK	600000	11	9.950600e-09	2.679245e+01	37.840440
Algorithm #1	800000	25	1.080014e-07	2.999988e+01	3.794971
Algorithm #2	800000	36	1.298740e-07	2.999983e+01	70.239330
Algorithm #3	800000	34	1.579465e-07	2.898582e+01	24.162290
MOSEK	800000	10	1.378357e-08	2.878444e+01	50.171970
Algorithm #1	1000000	25	1.784763e-07	3.171925e+01	4.765438
Algorithm #2	1000000	35	2.142598e-07	3.171919e+01	85.799120
Algorithm #3	1000000	34	2.671505e-07	3.064683e+01	29.847250
MOSEK	1000000	11	2.486922e-08	3.043398e+01	62.808910

Table 2 Results of numerical experiments for $m = 8$.

solver	n	iter	L_1 − infeasibility	L_2 − distance	cpu − time (sec)
Algorithm #1	1000	92	9.185763e-13	9.661599e+00	0.030596
Algorithm #2	1000	96	7.910783e-13	9.659397e+00	1.568706
Algorithm #3	1000	95	7.083667e-13	9.635578e+00	0.183006
MOSEK	1000	10	1.527667e-12	9.594969e+00	1.424148
Algorithm #1	2000	91	4.224865e-12	1.148149e+01	0.053792
Algorithm #2	2000	96	3.537148e-12	1.147910e+01	1.936123
Algorithm #3	2000	97	3.157785e-12	1.144696e+01	0.266952
MOSEK	2000	10	1.448086e-11	1.140187e+01	1.492126
Algorithm #1	4000	92	6.019829e-12	1.366479e+01	0.114719
Algorithm #2	4000	96	7.203238e-12	1.366203e+01	3.454818
Algorithm #3	4000	98	3.322032e-12	1.361465e+01	0.721250
MOSEK	4000	11	2.245315e-11	1.356832e+01	1.615156
Algorithm #1	6000	91	1.217690e-011	1.513198e+01	0.162270
Algorithm #2	6000	96	1.019493e-011	1.512942e+01	4.127975
Algorithm #3	6000	96	1.032396e-011	1.507714e+01	0.891475
MOSEK	6000	11	6.614300e-11	1.502658e+01	1.822972
Algorithm #1	8000	93	3.164813e-11	1.624974e+01	0.206762
Algorithm #2	8000	96	2.654570e-11	1.624709e+01	4.812220
Algorithm #3	8000	93	2.848004e-11	1.619055e+01	1.034865
MOSEK	8000	11	9.766836e-11	1.613593e+001	1.999692
Algorithm #1	10000	90	4.304954e-11	1.718458e+01	0.249486
Algorithm #2	10000	96	2.649905e-11	1.718176e+01	5.512854
Algorithm #3	10000	94	3.332281e-11	1.712162e+01	1.211829
MOSEK	10000	12	5.290985e-11	1.706337e+01	2.206122
Algorithm #1	20000	89	1.371168e-10	2.042371e+01	0.544347
Algorithm #2	20000	95	1.130954e-10	2.042032e+01	8.851602
Algorithm #3	20000	90	1.064138e-10	2.034898e+01	2.110184
MOSEK	20000	12	1.651642e-10	2.028043e+01	3.406446
Algorithm #1	40000	88	5.335203e-10	2.429703e+01	1.111483
Algorithm #2	40000	96	3.665033e-10	2.429307e+01	11.681860
Algorithm #3	40000	86	5.776692e-10	2.420798e+01	4.049940
MOSEK	40000	12	4.149115e-10	2.412582e+01	5.926457
Algorithm #1	60000	86	1.612989e-09	2.688631e+01	1.699176
Algorithm #2	60000	96	1.038555e-09	2.688203e+01	18.909850
Algorithm #3	60000	93	1.191358e-09	2.678802e+01	6.166266
MOSEK	60000	12	4.379217e-10	2.669699e+01	8.515426
Algorithm #1	80000	86	2.224886e-09	2.889426e+01	2.359484
Algorithm #2	80000	97	1.695007e-09	2.888948e+01	31.804910
Algorithm #3	80000	92	1.485279e-09	2.878829e+01	8.888214
MOSEK	80000	13	1.016087e-09	2.869085e+01	11.415490
Algorithm #1	100000	88	4.004576e-09	3.053834e+01	3.104578
Algorithm #2	100000	96	3.541345e-09	3.053305e+01	46.250940
Algorithm #3	100000	87	2.764879e-09	3.042614e+01	13.875280
MOSEK	100000	13	1.613989e-09	3.032310e+01	14.003310
Algorithm #1	200000	82	1.534256e-08	3.632432e+01	6.405284
Algorithm #2	200000	95	1.177904e-08	3.631815e+01	113.268300
Algorithm #3	200000	88	1.090808e-08	3.619110e+01	33.798230
MOSEK	200000	13	8.044299e-09	3.606874e+01	27.780870
Algorithm #1	400000	82	4.797786e-08	4.319705e+01	13.795880
Algorithm #2	400000	96	5.984427e-08	4.318954e+01	225.505100
Algorithm #3	400000	90	3.988183e-08	4.303840e+01	77.510010
MOSEK	400000	13	5.771581e-08	4.289291e+01	56.419040
Algorithm #1	600000	79	1.140747e-07	4.781224e+01	23.181630
Algorithm #2	600000	93	1.493796e-07	4.780413e+01	348.192600
Algorithm #3	600000	94	6.033820e-08	4.763699e+01	138.240900
MOSEK	600000	-	-	-	-
Algorithm #1	800000	83	1.992976e-07	5.137097e+01	32.392770
Algorithm #2	800000	95	1.626274e-07	5.136211e+01	477.002400
Algorithm #3	800000	84	2.243824e-07	5.118236e+01	183.778700
MOSEK	800000	-	-	-	-
Algorithm #1	1000000	75	3.243423e-07	5.431943e+01	36.961460
Algorithm #2	1000000	96	2.796773e-07	5.431023e+01	596.274400
Algorithm #3	1000000	88	3.110844e-07	5.412025e+01	228.998300
MOSEK	1000000	-	-	-	-

Table 3 Results of numerical experiments for $m = 12$.

solver	n	iter	$L_1 - infeasibility$	$L_2 - distance$	$cpu - time$ (sec)
Algorithm #1	1000	126	6.395773e-13	1.249028e+01	0.062426
Algorithm #2	1000	124	1.160894e-12	1.249319e+01	3.222102
Algorithm #3	1000	133	5.583756e-13	1.249067e+01	0.390431
MOSEK	1000	11	6.951950e-12	1.245982e+01	2.354783
Algorithm #1	2000	133	2.529421e-12	1.484951e+01	0.114152
Algorithm #2	2000	134	2.853362e-12	1.485241e+01	4.246328
Algorithm #3	2000	140	1.993516e-12	1.484953e+01	0.589232
MOSEK	2000	11	2.634124e-11	1.481209e+01	2.288296
Algorithm #1	4000	132	5.926171e-12	1.767684e+01	0.245575
Algorithm #2	4000	134	6.162315e-12	1.768057e+01	7.558384
Algorithm #3	4000	144	4.464340e-12	1.767705e+01	1.621358
MOSEK	4000	12	4.624781e-11	1.763295e+01	2.317519
Algorithm #1	6000	139	1.171518e-11	1.954789e+01	0.378776
Algorithm #2	6000	144	1.112175e-11	1.955219e+01	9.807958
Algorithm #3	6000	141	1.381366e-11	1.954824e+01	2.060957
MOSEK	6000	12	1.633794e-10	1.949947e+01	2.617219
Algorithm #1	8000	137	2.147711e-11	2.100726e+01	0.497407
Algorithm #2	8000	145	2.028722e-11	2.101154e+01	11.429530
Algorithm #3	8000	139	2.137839e-11	2.100740e+01	2.435979
MOSEK	8000	12	9.947598e-11	2.095508e+01	2.977148
Algorithm #1	10000	131	5.101879e-11	2.220466e+01	0.630924
Algorithm #2	10000	144	2.814928e-11	2.220936e+01	13.037910
Algorithm #3	10000	138	4.732044e-11	2.220485e+01	2.856426
MOSEK	10000	12	8.856205e-11	2.214928e+01	3.279846
Algorithm #1	20000	136	1.782319e-10	2.640910e+01	1.334466
Algorithm #2	20000	144	1.009762e-10	2.641453e+01	21.058040
Algorithm #3	20000	142	1.333941e-10	2.640920e+01	5.189846
MOSEK	20000	12	2.434717e-10	2.634338e+01	5.275118
Algorithm #1	40000	131	6.559456e-10	3.141181e+01	2.597049
Algorithm #2	40000	145	3.681665e-10	3.141841e+01	31.696810
Algorithm #3	40000	134	4.583061e-10	3.141205e+01	9.571323
MOSEK	40000	13	9.814812e-10	3.133356e+01	9.151956
Algorithm #1	60000	134	1.462423e-09	3.475545e+01	4.069634
Algorithm #2	60000	145	7.419609e-10	3.476275e+01	47.023570
Algorithm #3	60000	134	1.116603e-09	3.475575e+01	14.554330
MOSEK	60000	13	9.919859e-10	3.466872e+01	13.405950
Algorithm #1	80000	127	2.213489e-09	3.735978e+01	5.442247
Algorithm #2	80000	145	1.780147e-09	3.736737e+01	73.410850
Algorithm #3	80000	132	1.942451e-09	3.735986e+01	19.422540
MOSEK	80000	13	2.337219e-09	3.726674e+01	18.279120
Algorithm #1	100000	124	6.667263e-09	3.950290e+01	6.772145
Algorithm #2	100000	145	2.621760e-09	3.951111e+01	111.982000
Algorithm #3	100000	126	4.858959e-09	3.950313e+01	31.545770
MOSEK	100000	13	1.775697e-09	3.940489e+01	22.203150
Algorithm #1	200000	119	2.789554e-08	4.696020e+01	13.970610
Algorithm #2	200000	144	1.401164e-08	4.696999e+01	270.256300
Algorithm #3	200000	140	1.251096e-08	4.695358e+01	79.091730
MOSEK	200000	14	3.029490e-08	4.684324e+01	44.094940
Algorithm #1	400000	124	5.946722e-08	5.585180e+01	36.606730
Algorithm #2	400000	140	5.934934e-08	5.586315e+01	542.992300
Algorithm #3	400000	120	6.823090e-08	5.585191e+01	197.046900
MOSEK	400000	14	3.172754e-08	5.571279e+01	87.230710
Algorithm #1	600000	114	2.306928e-07	6.180785e+01	51.438620
Algorithm #2	600000	145	9.193727e-08	6.182068e+01	795.195800
Algorithm #3	600000	123	1.113523e-07	6.180821e+01	271.692800
MOSEK	600000	-	-	-	-
Algorithm #1	800000	113	4.301624e-07	6.642294e+01	68.190830
Algorithm #2	800000	141	6.152703e-07	6.643667e+01	1065.022000
Algorithm #3	800000	135	2.029717e-07	6.642326e+01	370.972100
MOSEK	800000	-	-	-	-
Algorithm #1	1000000	118	3.228161e-07	7.023215e+01	88.972150
Algorithm #2	1000000	144	1.856042e-07	7.024669e+01	1375.024000
Algorithm #3	1000000	118	3.125907e-07	7.021183e+01	541.216000
MOSEK	1000000	-	-	-	-

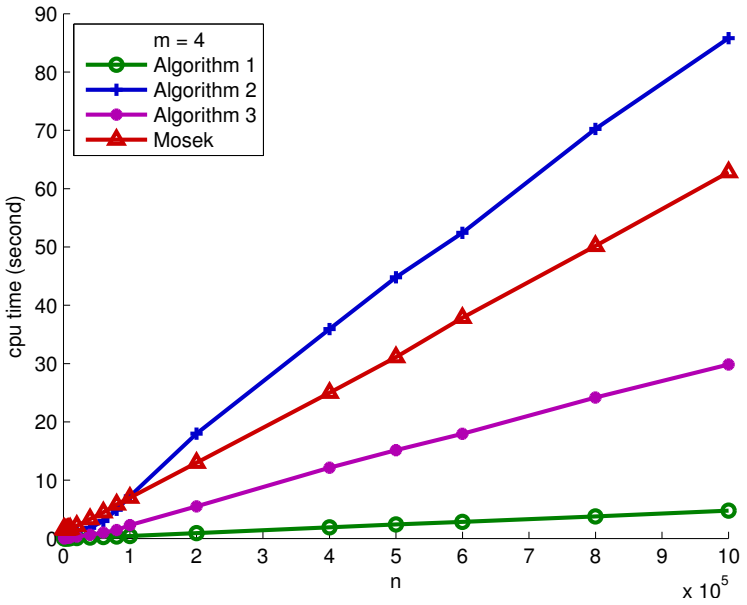

Fig. 1 CPU-time of algorithms #1, #2, #3 and MOSEK as a function of n for $m = 4$.

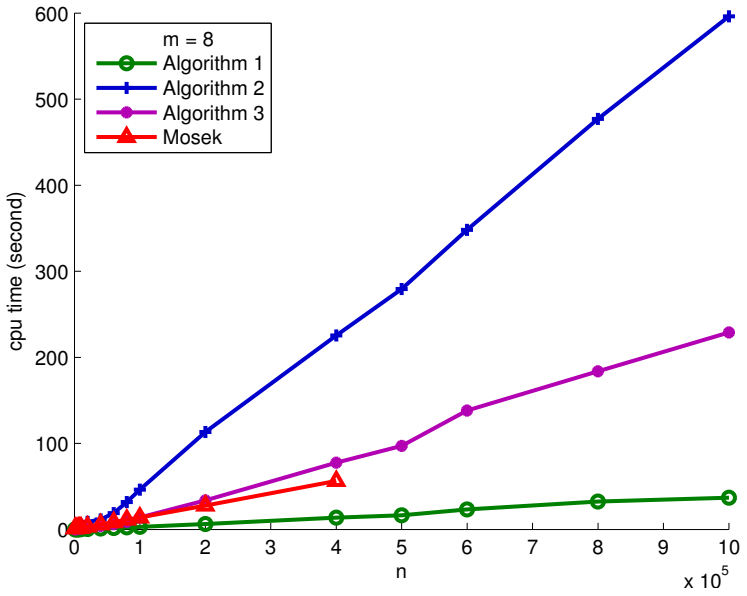

Fig. 2 CPU-time of algorithms #1, #2, #3 and MOSEK as a function of n for $m = 8$.

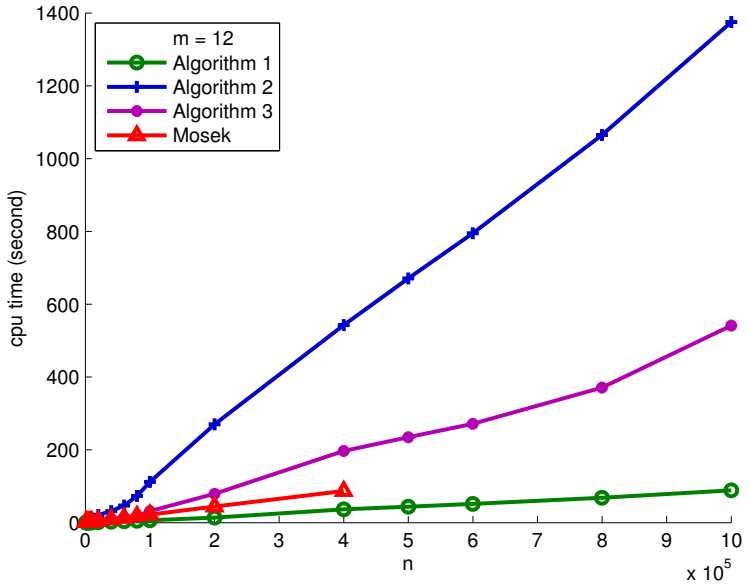

Fig. 3 CPU-time of algorithms #1, #2 and #3 as a function of n for case m = 12.

6 Summary and conclusion

The orthogonal projection of an arbitrary point on to the volume constrained Gibbs N-simplex is equivalent to the solution of a system of coupled continuous knapsack problems. Three efficient algorithms are introduced in the present study to solve the system of coupled continuous knapsack problems. They are based on the decomposition of the feasible domain into a few number of convex sets for which there are time-linear projection algorithms, and, solution of the original problem by the alternating projection algorithm. The accuracy and efficiency of the presented algorithms are studied numerically by solving a model problem with different sizes, ranging from 10^3 to 10^6. According to the numerical experiments of this study, the number of iterations to find the solution with maximum attainable accuracy is independent of parameter n. Moreover, when the size of problem is sufficiently large, the computational cost of the presented algorithms scales linearly with the problem size. Furthermore, the time-linear computational complexity of algorithm #1 is confirmed numerically in the whole spectrum of the problem size. Comparing results of the presented algorithms to those of MOSEK illustrates their competitive accuracy and efficiency in contrast to MOSEK. It is worth to mention that, algorithm #1 behaves an order of magnitude faster than algorithms #2 and #3, as well as MOSEK. Furthermore, the consumed memory of the presented algorithms scales linearly with the problem size. Moreover, they are superior than MOSEK in terms of the memory usage.

In summary, the main benefits of the presented algorithms in contrast to advanced QP/SOCP solvers, in particular MOSEK, can be listed as follows: competitive accuracy, competitive or even superior efficiency (almost the time-linear complexity), lower memory usage, simplicity and the ease of implementation.

Acknowledgements We would like to thanks Ernesto G. Birgin for the suggestion of alternating projection algorithm for the solution of problem studied in this work. Helpful comments by two anonymous reviewers which improve the presentation of our paper are greatly acknowledged.

References

1. S. Baldo. Minimal interface criterion for phase transitions in mixtures of cahn-hilliard fluids. *Ann Inst H. Poincaré (C) Anal Non Linéaire*, 7(2):67–90, 1990.
2. D.P. Bertsekas. Constrained optimization and lagrange multiplier methods. *Computer Science and Applied Mathematics, Boston: Academic Press, 1982*, 1, 1982.
3. E. Birgin, J.M. Martínez, and M. Raydan. Nonmonotone spectral projected gradient methods on convex sets. *SIAM J Optim*, 10(4):1196–1211, 2000.
4. E. Birgin, J.M. Martınez, and M. Raydan. Spectral projected gradient methods: Review and perspectives. *J Statistical Software*, 60(3), 2014.
5. L. Blank, H. Garcke, L. Sarbu, T. Srisupattarawanit, V. Styles, and A. Voigt. Phase-field approaches to structural topology optimization. In *Constrained Optimization and Optimal Control for Partial Differential Equations*, pages 245–256. Springer, 2012.
6. S. Boyd and J. Dattorro. Alternating projections. online note, 2003.
7. S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, 3(1):1–122, 2011.
8. S.P. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge university press, 2004.
9. E. Brown, T. Chan, and X. Bresson. Completely convex formulation of the chan-vese image segmentation model. *Int J Computer Vision*, 98(1):103–121, 2012.
10. A. Chambolle, D. Cremers, and T. Pock. A convex approach to minimal partitions. *SIAM J Imaging Sci*, 5(4):1113–1158, 2012.
11. T. Chan and L. Vese. Active contours without edges. *IEEE T Image Processing*, 10(2):266–277, 2001.
12. Y. Dai and R. Fletcher. New algorithms for singly linearly constrained quadratic programs subject to lower and upper bounds. *Math Prog*, 106(3):403–421, 2006.
13. R. Escalante and M. Raydan. *Alternating Projection Methods*, volume 8. SIAM, 2011.
14. H. Garcke, B. Nestler, B. Stinner, and F. Wendler. Allen–cahn systems with volume constraints. *Math Mod Meth Appl Sci*, 18(08):1347–1381, 2008.
15. H. Garcke, B. Nestler, and B. Stoth. A multiphase field concept: numerical simulations of moving phase boundaries and multiple junctions. *SIAM J Appl Math*, 60(1):295–315, 1999.
16. Josiah Willard Gibbs. *On the equilibrium of heterogeneous substances*. Connecticut Academy of Arts and Sciences, 1877.
17. W. Hager and H. Zhang. A new active set algorithm for box constrained optimization. *SIAM J Optim*, 17(2):526–557, 2006.
18. K. Kiwiel. On linear-time algorithms for the continuous quadratic knapsack problem. *J Optim Theory Appl*, 134(3):549–554, 2007.
19. K. Kiwiel. Breakpoint searching algorithms for the continuous quadratic knapsack problem. *Math Prog*, 112(2):473–491, 2008.
20. K. Kiwiel. Variable fixing algorithms for the continuous quadratic knapsack problem. *J Optim Theory Appl*, 136(3):445–458, 2008.
21. J. Lellmann and C. Schnörr. Continuous multiclass labeling approaches and algorithms. *SIAM J Imaging Sci*, 4(4):1049–1096, 2011.
22. Y. Nesterov and A. Nemirovskii. *Interior-point polynomial algorithms in convex programming*, volume 13. SIAM, 1994.
23. B. Nestler, F. Wendler, M. Selzer, B. Stinner, and H. Garcke. Phase-field model for multiphase systems with preserved volume fractions. *Phys Rev E*, 78(1):011604, 2008.
24. J. Nocedal and S. Wright. *Numerical optimization*. Springer, 2006.
25. P.M. Pardalos and N. Kovoor. An algorithm for a singly constrained class of quadratic programs subject to upper and lower bounds. *Math Prog*, 46(1-3):321–328, 1990.
26. A.G. Robinson, N. Jiang, and C.S. Lerme. On the continuous quadratic knapsack problem. *Mathematical Programming*, 55(1-3):99–108, 1992.
27. Jo Bo Rosen. The gradient projection method for nonlinear programming. part i. linear constraints. *J Soc Indust Appl Math*, 8(1):181–217, 1960.
28. I Steinbach, F Pezzolla, B Nestler, M Seeßelberg, R Prieler, GJ Schmitz, and JLL Rezende. A phase field concept for multiphase systems. *Physica D: Nonlin Phenomena*, 94(3):135–147, 1996.
29. R. Tavakoli. Multimaterial topology optimization by volume constrained allen-cahn system and regularized projected steepest descent method. *Comput Meth Appl Mech Eng*, 276:534–565, 2014.
30. R. Tavakoli and S.M. Mohseni. Alternating active-phase algorithm for multimaterial topology optimization problems: a 115-line matlab implementation. *Struct Multidis Optim*, 49:621–642, 2013.
31. R. Tavakoli and H. Zhang. A nonmonotone spectral projected gradient method for large-scale topology optimization problems. *Numer Algebra, Control Optim*, 2(2):395–412, 2012.
32. S.J. Wright. *Primal-dual interior-point methods*, volume 54. SIAM, 1997.
33. L. Zhen, G. Wei, and S. Chongmin. Design of multi-phase piezoelectric actuators. *J Intel Mater Sys Struct*, 21(8):1851–1865, 2010.
34. S. Zhou and M.Y. Wang. Multimaterial structural topology optimization with a generalized cahn–hilliard model of multiphase transition. *Struct Multidisc Optim*, 33(2):89–111, 2007.