A Simplex Algorithm with the Smallest Index Rule for Concave Quadratic Programming

Mohand Bentobache, Mohamed Telli and Abdelkader Mokhtari

Laboratory of Pure and Applied Mathematics
University of Laghouat, 03000, Laghouat, Algeria
Email: m.bentobache@lagh-univ.dz

***Abstract*—In this work, we propose a new algorithm called ”Simplex Algorithm with the Smallest Index Rule” for finding a local minimum of a concave quadratic function subject to linear equality and nonnegativity constraints. First, we present and prove a new sufficient and necessary condition for local optimality, then we describe the developed algorithm and we give a numerical example for illustration purpose. In order to prove the efficiency of our algorithm, we developed an implementation using MATLAB, then we conducted numerical experiments on randomly generated and Rusakov’s concave quadratic test problems. The obtained numerical results show that our algorithm outperforms the branch and bound algorithm suggested by Rusakov in terms of CPU time and it gives the global optimal solution for the Rusakov’s test problems. Furthermore, it gives the global optimum for some generated test problems and it finds, for other problems, a local minimizer which can be used to initialize global optimization algorithms.**

***Keywords*–Concave quadratic programming; Local minimum; Global minimum, Simplex algorithm, Numerical experiments.**

I. Introduction

The Concave Quadratic Programming (CQP) problem consists in minimizing a concave quadratic function under a convex polyhedron delimited by linear constraints. This optimization problem has important theoretic and practical aspects. Indeed, many practical problems are modeled as CQP problems, we can cite the quadratic assignment problem [1], missile flight testing [2], etc.

Unlike the convex quadratic programming problem, this problem is difficult to solve since a local optimal solution is not in general a global one. Therefore, in many research articles, the authors developed algorithms for approximate the global optimum of the problem. The first algorithm for solving the problem is suggested by Tuy [3]. The principle of the Tuy’s algorithm is to compute a new linear constraint, called Tuy cut, in order to eliminate points in feasible region, which can not be global optimal solutions. Later, many algorithms are developed: branch and bound algorithms [4][5], cutting plane algorithms [6], successive underestimating method [7], metaheuristic algorithms [8], etc.

The majority of the proposed global optimization algorithms starts by a local optimal solution. It is proved in [9] that a local optimal solution of the problem is an extreme point of the convex polyhedron corresponding to the linear constraints. Hence, in this work we suggest a new algorithm called ”Simplex Algorithm with the Smallest Index Rule” (SASIR) for finding an extreme point, which is a local minimum for the considered problem. The principle of our algorithm is similar to the one of the simplex algorithm of linear programming [10]: it starts by an initial extreme point obtained using some existing initialization technique of the simplex method, then it moves in each iteration from one extreme point to a new one having a better value of the quadratic objective function and finally it stops when a local optimality condition is satisfied.

In order to test the efficiency of our method, we have implemented it in MATLAB and conducted numerical experiments on Rusakov’s test problems [5] inspired from practical problems arising in the area of missile flight testing and a set of randomly generated test problems with known global minimum and size varying from 100 constraints and 120 variables up to 200 constraints and 240 variables. The obtained numerical results are very encouraging. Indeed, our algorithm gives a local optimum which is also global for Rusakov’s test problems and it outperforms the Rusakov’s algorithm implemented in his software [11] in terms of CPU time. Furthermore, SASIR finds the global optimum for some randomly generated test problems in reasonable amount of time and it gives, for other test problems, a local minimizer which can be used as initial point for global optimization algorithms.

The paper is organized as follows: in Section II, we present the problem, we give some definitions and we recall some fondamental results of concave quadratic programming. In Section III, we present and prove the suggested local optimality criterion. In Section IV, we describe and justify the suggested algorithm. Moreover, we illustrate our approach with a numerical example. In Section V, we present some numerical results in order to compare our algorithm with the branch and bound algorithm of Rusakov [5] which uses the Tuy cut. Finally, Section VI concludes the paper and gives some future works.

II. Presentation of the problem and definitions

The concave quadratic programming problem with equality and nonnegativity contraints is presented in the following form:

$$\begin{aligned} &\min f(x) = \tfrac{1}{2}x^T D x + c^T x, \\ &\text{subject to} \quad Ax = b,\ x \geq 0, \end{aligned} \tag{1}$$

where D is an $(n \times n)$ real symmetric negative semidefinite matrix, c and x are n-vectors; A is a matrix of dimension $m \times n$, with $Rank(A) = m < n$.

• Let us define the following sets of indices:

$$I = \{1, 2, \ldots, m\}, \quad J = \{1, 2, \ldots, n\}, \quad J = J_B \cup J_N,$$

$$J_B \cap J_N = \emptyset,\ \ |J_B| = m,\ K = \{1, \dots, n-m\}.$$

We can partition the vectors x, c and the matrix A as follows:

$$x = \begin{pmatrix} x_B \\ x_N \end{pmatrix},\ x_B = (x_j, j \in J_B),\ x_N = (x_j, j \in J_N),$$

$$c = \begin{pmatrix} c_B \\ c_N \end{pmatrix},\ c_B = (c_j, j \in J_B),\ c_N = (c_j, j \in J_N),$$

$$A = (a_{ij}, i \in I, j \in J) = (a_j, j \in J),\ a_j = \begin{pmatrix} a_{1j} \\ \vdots \\ a_{mj} \end{pmatrix},$$

$$A = (A_B, A_N),\ A_B = A(I, J_B),\ A_N = A(I, J_N).$$

• We denote the feasible region of problem (1) by

$$S = \{x \in \mathbb{R}^n :\ Ax = b \text{ and } x \geq 0\}.$$

• A vector $x \in S$ is called a feasible solution for the problem (1).
• Let $J_B \subset J$ be a subset of indices such that $|J_B| = |I| = m$. The matrix $A_B = A(I, J_B)$ is said to be a basis matrix if $\det(A_B) \neq 0$. Then the feasible solution $x = \begin{pmatrix} x_B \\ x_N \end{pmatrix}$, with $x_B = A_B^{-1}b \geq 0$ and $x_N = 0$ is called a Basic Feasible Solution (BFS).
• A BFS x is said to be nondegenerate if $x_j > 0,\ j \in J_B$.
• Let $A_B = A(I, J_B)$ be a basis matrix, $J_N = J \setminus J_B$ and x the corresponding BFS. Let $j_0 \in J_N,\ j_1 \in J_B$ be two indices, and $\bar{J}_B = (J_B \setminus \{j_1\}) \cup \{j_0\}$, $\bar{A}_B = A(I, \bar{J}_B)$, such that $det(\bar{A}_B) \neq 0$. Let $\bar{x}$ be the BFS corresponding to the new basis matrix $\bar{A}_B$. Hence, we say that the basic feasible solutions x and $\bar{x}$ are adjacent.
• Let x^* be a feasible solution for problem (1). We say that x^* is a local minimizer if it exists a neighborhood $N(x^*)$ of x^*, such that $\forall x \in N(x^*) \cap S,\ f(x^*) \leq f(x)$. The vector x^* is said to be a global minimizer if $f(x^*) \leq f(x),\ \forall x \in S$.
Let us recall the following fundamental result [9]:

Theorem 1. *Let f be a concave function defined on the bounded, closed convex set Ω. If f has a minimum over Ω, then it is achieved at an extreme point of Ω.*

• Since D is negative semidefinite, the quadratic function f is concave. Therefore, the global minimizer is achieved at an extreme point of the convex polyhedron S. This leads us to give the following definitions: let A_B^* a basis matrix and x^* the corresponding BFS, we denote by $\mathcal{N}(x^*)$ the set of all basic feasible solutions, which are adjacent to x^*. We say that x^* is a local minimizer for problem (1), if it satisfies $f(x^*) \leq f(x),\ \forall x \in \mathcal{N}(x^*)$. We say that x^* is a global minimizer for problem (1), if for any BFS $x \in S$, we have $f(x^*) \leq f(x)$.
• Let J_B be a set of basic indices for problem (1) and $J_N = J \setminus J_B$. We define the following vectors and matrices:

$$\bar{b} = (\bar{b}_i, i \in I) = A_B^{-1}b,\ \bar{A} = (\bar{a}_k,\ k \in K) = -A_B^{-1}A_N, \quad (2)$$

$$Z = \begin{pmatrix} \bar{A} \\ I_{n-m} \end{pmatrix} \text{ and } Q = (q_{ij},\ i, j \in K) = Z^T \bar{D} Z, \quad (3)$$

where I_{n-m} represents the identity matrix of order $n - m$ and $\bar{D} = P^T D P$, P is the permutation matrix obtained by permuting columns of the identity matrix I_n with respect to the partition (J_B, J_N). Note that the matrix Q is negative semidefinite. Indeed, $\forall y \in \mathbb{R}^{n-m}$, we have

$$y^T Q y = y^T Z^T \bar{D} Z y = (Zy)^T \bar{D}(Zy) \leq 0.$$

Moreover, we note that the diagonal elements of a negative semidefinite matrix are less than or equal to zero. Indeed, $\forall y \in \mathbb{R}^{n-m} : y^T Q y \leq 0$. Particularly, for $y = e_k$, where e_k is the vector with all its components equal to zero except for the k^{th} component, it is equal to 1. Hence, we get $y^T Q y = q_{kk} \leq 0$.
• A vector $d \in \mathbb{R}^n$ is said to be a feasible direction for problem (1) if it satisfies $Ad = 0$.

III. Increment formula of the objective function

Using results presented in [12][13][14] on linear and convex quadratic programming, we can deduce the increment formula of the objective function for the concave quadratic programming problem (1), when we move from a BFS to an adjacent one.

Let A_B be a basis matrix for problem (1) and $x = \begin{pmatrix} x_B \\ x_N \end{pmatrix} = \begin{pmatrix} \bar{b} \\ 0 \end{pmatrix}$ the corresponding BFS. Let $\bar{x} = \begin{pmatrix} \bar{x}_B \\ \bar{x}_N \end{pmatrix}$ be an arbitrary feasible solution (not necessarily basic) and $f(\bar{x})$ the value of the objective function at $\bar{x}$.
Since $\bar{x}$ is feasible, we can write:

$$A_B\bar{x}_B + A_N\bar{x}_N = b \Leftrightarrow \bar{x}_B = A_B^{-1}b - A_B^{-1}A_N\bar{x}_N = \bar{b} + \bar{A}\bar{x}_N. \quad (4)$$

The objective function value at $\bar{x}$ is

$$f(\bar{x}) = \begin{pmatrix} c_B \\ c_N \end{pmatrix}^T \begin{pmatrix} \bar{x}_B \\ \bar{x}_N \end{pmatrix} + \frac{1}{2} \begin{pmatrix} \bar{x}_B \\ \bar{x}_N \end{pmatrix}^T \bar{D} \begin{pmatrix} \bar{x}_B \\ \bar{x}_N \end{pmatrix} \quad (5)$$

By replacing the expression of $\bar{x}_B$ in equation (5), we get

$$\begin{aligned} f(\bar{x}) &= c_B^T\bar{b} + (c_N^T + c_B^T\bar{A})\bar{x}_N + \frac{1}{2}\begin{pmatrix} \bar{b} \\ 0 \end{pmatrix}^T \bar{D} \begin{pmatrix} \bar{b} \\ 0 \end{pmatrix} \\ &\quad + \begin{pmatrix} \bar{b} \\ 0 \end{pmatrix}^T \bar{D} \begin{pmatrix} \bar{A}\bar{x}_N \\ \bar{x}_N \end{pmatrix} \\ &\quad + \frac{1}{2}\begin{pmatrix} \bar{A}\bar{x}_N \\ \bar{x}_N \end{pmatrix}^T \bar{D} \begin{pmatrix} \bar{A}\bar{x}_N \\ \bar{x}_N \end{pmatrix} \\ &= c_B^T\bar{b} + \frac{1}{2}\begin{pmatrix} \bar{b} \\ 0 \end{pmatrix}^T \bar{D} \begin{pmatrix} \bar{b} \\ 0 \end{pmatrix} \\ &\quad + \left[c_N^T + c_B^T\bar{A} + \begin{pmatrix} \bar{b} \\ 0 \end{pmatrix}^T \bar{D} \begin{pmatrix} \bar{A} \\ I_{n-m} \end{pmatrix}\right]\bar{x}_N \\ &\quad + \frac{1}{2}\bar{x}_N^T \begin{pmatrix} \bar{A} \\ I_{n-m} \end{pmatrix}^T \bar{D} \begin{pmatrix} \bar{A} \\ I_{n-m} \end{pmatrix} \bar{x}_N. \\ &= f(x) + \left[c_N^T + c_B^T\bar{A} + \begin{pmatrix} \bar{b} \\ 0 \end{pmatrix}^T \bar{D}Z\right]\bar{x}_N \\ &\quad + \frac{1}{2}\bar{x}_N^T Z^T \bar{D} Z \bar{x}_N. \end{aligned}$$

Thus, the objective function increment takes the following final form:

$$f(\bar{x}) - f(x) = l^T\bar{x}_N + \frac{1}{2}\bar{x}_N^T Q \bar{x}_N, \quad (6)$$

where

$$l^T = c_N^T + c_B^T \bar{A} + v^T Z, \text{ with } v^T = \begin{pmatrix} \bar{b} \\ 0 \end{pmatrix}^T \bar{D}. \quad (7)$$

Remark 1.
• *If we denote by $\bar{D} = (\bar{d}_{ij},\ i,j \in J)$ and $v = (v_j, j \in J)$, then the components of the n-vector v are computed as follows:*

$$v_j = \sum_{i=1}^{m} \bar{b}_i \bar{d}_{ij},\ j = 1, 2, \ldots, n.$$

• *When $D = 0$, the problem (1) becomes a linear program and the increment of the objective function becomes:*

$$f(\bar{x}) - f(x) = l^T \bar{x}_N = (c_N^T + c_B^T \bar{A})\bar{x}_N = (c_N^T - c_B^T A_B^{-1} A_N)\bar{x}_N.$$

So, the vector l is equal to the reduced costs vector in the simplex method of linear programming.

Let us denote by

$$J_N = \{j_1, j_2, \ldots, j_{n-m}\} \text{ and } J_B = \{j'_1, j'_2, \ldots, j'_m\}.$$

So, if $k \in K$, then j_k represents the index of position k in J_N, and if $s \in I$, then j'_s represents the index of position s in J_B.
We introduce the following notations:

$$\bar{a}_{j_k} = (\bar{a}_{ij_k}, i \in I) = -A_B^{-1} a_{j_k}. \quad (8)$$

$$\theta_{i_k} = \min_{i \in I} \theta_i^k, \text{ with } \theta_i^k = \begin{cases} \frac{-\bar{b}_i}{\bar{a}_{ij_k}}, & \text{if } \bar{a}_{ij_k} < 0; \\ +\infty, & \text{otherwise.} \end{cases} \quad (9)$$

$$\theta_0^k = \begin{cases} \frac{-2l_k}{q_{kk}}, & \text{if } q_{kk} < 0; \\ -\infty, & \text{if } q_{kk} = 0 \text{ and } l_k > 0; \\ +\infty, & \text{otherwise.} \end{cases} \quad (10)$$

The following theorem gives us a sufficient and necessary condition for the local optimality of the BFS x.

Theorem 2. *The condition*

$$\forall k \in K : \theta_0^k \geq \theta_{i_k} \quad (11)$$

is sufficient for the local optimality of the BFS x and it is also necessary when x is nondegenerate.

Proof:
Sufficient condition. Let $\bar{x}$ be an arbitrary adjacent BFS to x and assume that the basis matrix corresponding to $\bar{x}$ is $\bar{A}_B = A(I, \bar{J}_B)$, where $\bar{J}_B = (J_B \setminus \{j'_s\}) \cup \{j_r\}$.
We assume that condition (11) holds. We have

$$f(\bar{x}) - f(x) = l^T \bar{x}_N + \frac{1}{2}\bar{x}_N^T Q \bar{x}_N.$$

However, for the BFS $\bar{x}$, we have $\bar{x}_{j_k} = 0, k \neq r$ and $\bar{x}_{j_r} = \theta_s \geq 0$. Since $\theta_0^r \geq \theta_s$, we get

$$f(\bar{x}) - f(x) = l_r \bar{x}_{j_r} + \frac{1}{2} q_{rr} \bar{x}_{j_r}^2 = l_r \theta_s + \frac{1}{2} q_{rr} \theta_s^2 \geq 0.$$

Necessary condition. Assume that x is nondegenerate and the condition (11) is not satisfied, i.e.,

$$\exists r \in K : \theta_0^r < \theta_{i_r} \quad (12)$$

Thus, we can move to a new BFS $\bar{x}$, such that $f(\bar{x}) < f(x)$. Indeed, we can improve the point $x = \begin{pmatrix} x_B \\ x_N \end{pmatrix} = \begin{pmatrix} A_B^{-1} b \\ 0 \end{pmatrix}$, by increasing the value of the component x_{j_r} by a positive number θ and letting the other nonbasic components equal to zero. Thus, we obtain a new point $\bar{x} = \begin{pmatrix} \bar{x}_B \\ \bar{x}_N \end{pmatrix}$, such that

$$\bar{x}_N = x_N + \theta e_r = \theta e_r,$$
$$\bar{x}_B = \bar{b} + \bar{A}\bar{x}_N = \bar{b} + \bar{A}(\theta e_r) = \bar{b} + \theta \bar{a}_{j_r},$$

where e_r represents the $(n-m)$-vector of zeros except for the component r, it is equal to 1. The positif number θ can be chosen in such a way that the new point $\bar{x}$ remains feasible:

$$\bar{x}_N = \theta e_r \geq 0,\ \bar{x}_B = \bar{b} + \theta \bar{a}_{j_r} \geq 0,$$

and the objective function decreases:

$$f(\bar{x}) - f(x) = l_r \theta + \frac{1}{2} q_{rr} \theta^2 < 0.$$

Indeed, using the nondegeneracy assumption $(\bar{b}_i > 0, i \in I)$, we get $\theta_s = \theta_{i_r} = \min_{i \in I}\{\theta_i^r\} > 0$, then condition $\theta_0^r < \theta_s$ implies that the number θ can be chosen in the interval $]0, \theta_s]$ if $\theta_0^r \leq 0$, or in the interval $]\theta_0^r, \theta_s]$, otherwise. Hence, we get $\bar{x} \geq 0$ and $f(\bar{x}) < f(x)$. Therefore, the BFS x is not a local minimizer. ■

IV. An iteration of SASIR

Let x be an initial BFS of the problem (1). An iteration of the algorithm SASIR consists in moving from the BFS x to a new BFS $\bar{x}$, with $f(\bar{x}) \leq f(x)$ following the descent feasible direction used in the the simplex method for linear programming. The algorithm stops when the local optimality criterion (11) is satisfied.

A. Computing the feasible descent direction

We define the following set of indices:

$$K^* = \{k \in K : \theta_0^k < \theta_{i_k}\}. \quad (13)$$

Two cases can occur:
Case 1. If $K^* = \emptyset$, then the algorithm stops. The BFS x is a local minimizer.
Case 2. If $K^* \neq \emptyset$, then we choose an index $j_r \in J_N$ that satisfies the smallest index rule, i.e., we choose the index r that satisfies

$$r = \min\{k,\ k \in K^*\}$$

and we compute the feasible descent direction d as follows:

$$d_N = e_r,\ d_B = \bar{a}_{j_r}, \quad (14)$$

where j_r corresponds to the index of position r in J_N.
Note that the direction d is a feasible direction because it satisfies $Ad = 0$.
We move along the direction d with a steplength $\theta^* = \theta_{i_r}$, to achieve a new BFS $\bar{x} = x + \theta^* d$, with a better objective function value:

$$f(\bar{x}) = f(x) + l_r \theta^* + \frac{1}{2} q_{rr} (\theta^*)^2 \leq f(x).$$

Then, we proceed to the change of the basis:

$$\bar{J}_B = (J_B \setminus \{j'_s\}) \cup \{j_r\},\ s = i_r.$$

We start a new iteration with the new BFS $\bar{x}$ and the new basis matrix $\bar{A}_B = A(I, \bar{J}_B)$.

B. Algorithm SASIR

(1) Compute $\bar{b}$, $\bar{A}$, Z, Q and the vector l with formulas (2)-(3) and (7);

(2) (Computing the entering index)
For every index k in K, compute θ_{i_k} and θ_0^k with relations (9)-(10); then compute the set K^* with relation (13);
Cas 1. If $K^* = \emptyset$, then the algorithm stops with a local minimizer x.
Cas 2. If $K^* \neq \emptyset$, then
compute the entering index j_r corresponding to the index of position r in J_N, with the smallest index rule, i.e., $r = \min\{k,\ k \in K^*\}$, and set the leaving index $s = i_r$;

(3) (Computing the direction and the steplength)
Compute the direction d with formula (14);
set the steplength $\theta^* = \theta_s = \theta_{i_r}$;

(4) (Change of the current solution)
Set

$$\bar{x} = x + \theta^* d,\ f(\bar{x}) = f(x) + l_r\theta^* + \frac{1}{2}q_{rr}(\theta^*)^2;$$

(5) (Change of the current basis)
Set

$$\bar{J}_B = (J_B \setminus \{j'_s\}) \cup \{j_r\};$$

(6) Set $x = \bar{x}$, $J_B = \bar{J}_B$ and go to step (1).

Remark 2.
• *Under the nondegeneracy assumption, our algorithm moves from one BFS x to a new one $\bar{x}$, with $f(\bar{x}) < f(x)$. Since the number of extreme points of the convex polyhedron S is finite, our algorithm finds a local optimum in a finite number of steps.*
• *We can choose the entering index j_r with the best improvement rule:*

$$\Delta f_r = \max\{\Delta f_k, k \in K^*\},\ \textit{with}\ \Delta f_k = l_k\theta_{i_k} + \frac{1}{2}q_{kk}\theta_{i_k}^2.$$

This rule gives the maximal local improvement of the objective function. In other words, we move from the current extreme point to the adjacent one with the best objective function value. However, some preliminary numerical experiments show that this version of the algorithm consumes much time than the version SASIR presented above.

C. Numerical example

Consider the following concave quadratic program [14]:

$$\begin{array}{rl} \min & f(x) = -x_1 - 2x_2 - x_1^2 - 3x_2^2, \\ \text{subject to} & -x_1 + x_2 + x_3 = 3, \\ & x_1 - x_2 + x_4 = 6, \\ & x_1 + 2x_2 + x_5 = 12, \\ & x_j \geq 0,\ j = \overline{1,5}. \end{array}$$

We have

$$I = \{1,2,3\},\ J = \{1,2,3,4,5\},\ K = \{1,2\}.$$

$$c = \begin{pmatrix} -1 \\ -2 \\ 0 \\ 0 \\ 0 \end{pmatrix},\ D = \begin{pmatrix} -2 & 0 & 0 & 0 & 0 \\ 0 & -6 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix},$$

$$A = \begin{pmatrix} -1 & 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 1 & 0 \\ 1 & 2 & 0 & 0 & 1 \end{pmatrix},\ b = \begin{pmatrix} 3 \\ 6 \\ 12 \end{pmatrix}.$$

We start SASIR with the following initial BFS:

$$J_B = \{3,4,5\},\ J_N = \{1,2\},\ x_B^T = (3,6,12),$$
$$x_N^T = (0,0),\ x^T = (0,0,3,6,12),\ f(x) = 0.$$

First iteration :
We have

$$A_B = I_3,\ A_N = \begin{pmatrix} -1 & 1 \\ 1 & -1 \\ 1 & 2 \end{pmatrix},\ c_B^T = 0_{\mathbb{R}^3},\ c_N^T = (-1,-2).$$

Computing the vectors $\bar{b}$, l and the matrices $\bar{A}$, Z, Q:

$$\bar{b} = A_B^{-1}b = b,\ \bar{A} = -A_B^{-1}A_N = \begin{pmatrix} 1 & -1 \\ -1 & 1 \\ -1 & -2 \end{pmatrix},$$

$$Z = \begin{pmatrix} \bar{A} \\ I_2 \end{pmatrix} = \begin{pmatrix} 1 & -1 \\ -1 & 1 \\ -1 & -2 \\ 1 & 0 \\ 0 & 1 \end{pmatrix},$$

$$\bar{D} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -2 & 0 \\ 0 & 0 & 0 & 0 & -6 \end{pmatrix},\ Q = Z^T\bar{D}Z = \begin{pmatrix} -2 & 0 \\ 0 & -6 \end{pmatrix},$$

$$v = \begin{pmatrix} \bar{b} \\ 0 \end{pmatrix}^T \bar{D} = 0_{\mathbb{R}^5},\ l^T = c_N^T + c_B^T\bar{A} + v^T Z = (-1,-2).$$

Computing the entering and leaving indices:
We compute θ_{i_k} and θ_0^k for $k \in \{1,2\}$:

$$\theta_{i_1} = \min\{\theta_1^1, \theta_2^1, \theta_3^1\} = \min\{+\infty, 6, 12\} = \theta_2^1 = 6,$$
$$\theta_{i_2} = \min\{\theta_1^2, \theta_2^2, \theta_3^2\} = \min\{3, +\infty, 6\} = \theta_1^2 = 3,$$
$$\theta_0^1 = -\frac{2l_1}{q_{11}} = -1,\ \theta_0^2 = -\frac{2l_2}{q_{22}} = -2/3.$$

So,

$$K^* = \{k \in K : \theta_0^k < \theta_{i_k}\} = \{1,2\}.$$

We choose r with the smallest index rule, we get

$$r = \min\{k,\ k \in K^*\} = \min\{1,2\} = 1 \Rightarrow r = 1,\ s = i_1 = 2.$$

So, the entering index is the index of position $r = 1$ in J_N, i.e., $j_1 = 1$ and the leaving index is the index of position s in J_B, i.e., $j'_2 = 4$.

Computing the feasible descent direction and the steplength: We have

$$d_N = (0,1)^T,\ d_B = \bar{a}_2 = (-1,1,-2)^T,$$
$$d^T = (1,0,1,-1,-1),\ \theta^* = \theta_{i_1} = 6.$$

Computing the new solution $\bar{x}$ and the new value of the objective function $f(\bar{x})$:

$$\bar{x} = x + \theta^* d = (6,0,9,0,6)^T,\ f(\bar{x}) = -42 < f(x).$$

Change of the basis:

$$\bar{J}_B = (J_B \setminus \{4\}) \cup \{1\} = \{3,1,5\},\ \bar{J}_N = \{4,2\}.$$

Second iteration :
We have

$$J_B = \{3,1,5\},\ J_N = \{4,2\},\ x^T = (6,0,9,0,6),$$
$$f(x) = -42,\ A_B = \begin{pmatrix} 1 & -1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix},\ A_N = \begin{pmatrix} 0 & 1 \\ 1 & -1 \\ 0 & 2 \end{pmatrix},$$
$$c_B = (0,-1,0)^T,\ c_N = (0,-2)^T.$$

Computing the matrices $\bar{A}$, Z, Q and the vector l:

$$Z = \begin{pmatrix} -1 & 0 \\ -1 & 1 \\ 1 & -3 \\ 1 & 0 \\ 0 & 1 \end{pmatrix},\ Q = \begin{pmatrix} -2 & 2 \\ 2 & -8 \end{pmatrix},\ l = \begin{pmatrix} 13 \\ -15 \end{pmatrix}.$$

We compute θ_{i_k} and θ_0^k, $k \in \{1,2\}$:

$$\theta_{i_1} = \{9,6,+\infty\} = \theta_2^1 = 6,\ \theta_0^1 = 13,$$
$$\theta_{i_2} = \{+\infty,+\infty,2\} = \theta_3^2 = 2,\ \theta_0^2 = -15/4.$$

So,
$$K^* = \{k \in K : \theta_0^k < \theta_{i_k}\} = \{2\}.$$

We choose r with the smallest index rule, we get

$$r = \min\{k,\ k \in K^*\} = 2 \Rightarrow r = 2,\ s = i_2 = 3.$$

So, the entering index $j_2 = 2$ and the leaving index is $j_3' = 5$.
Computing the feasible descent direction and the steplength:

$$d^T = (1,1,0,0,-3),\ \theta^* = \theta_{i_2} = 2.$$

Computing the new BFS $\bar{x}$ and the new value of the objective function $f(\bar{x})$:

$$\bar{x} = x + \theta^* d = (8,2,9,0,0)^T,\ f(\bar{x}) = -88 < f(x).$$

Change of the basis:

$$\bar{J}_B = (J_B \setminus \{5\}) \cup \{2\} = \{3,1,2\},\ \bar{J}_N = \{4,5\}.$$

Third iteration :
We have

$$J_B = \{3,1,2\},\ J_N = \{4,5\},\ x^T = (8,2,9,0,0),$$
$$f(x) = -88,\ A_B = \begin{pmatrix} 1 & -1 & 1 \\ 0 & 1 & -1 \\ 0 & 1 & 2 \end{pmatrix},\ A_N = \begin{pmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix},$$
$$c_B = (0,-1,-2)^T,\ c_N = (0,0)^T.$$

Computing the matrices Z, Q and the vector l:

$$Z = \begin{pmatrix} -1 & 0 \\ -2/3 & -1/3 \\ 1/3 & -1/3 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}, Q = \begin{pmatrix} -14/9 & 2/9 \\ 2/9 & -8/9 \end{pmatrix},$$
$$l = (20/3,\ 31/3)^T.$$

We compute θ_{i_k} and θ_0^k, $k \in \{1,2\}$:

$$\theta_{i_1} = \{9,12,+\infty\} = \theta_1^1 = 9,\ \theta_0^1 = 60/7,$$
$$\theta_{i_2} = \{+\infty,24,6\} = \theta_3^2 = 6,\ \theta_0^2 = 93/4.$$

So,
$$K^* = \{k \in K : \theta_0^k < \theta_{i_k}\} = \{1\}.$$

We choose r with the smallest index rule, we get

$$r = \min\{k,\ k \in K^*\} = 1 \Rightarrow r = 1,\ s = i_1 = 1.$$

So, the entering index is $j_1 = 4$ and the leaving index is $j_1' = 3$.
The feasible descent direction and the steplength are:

$$d^T = (-2/3,1/3,-1,1,0),\ \theta^* = \theta_{i_1} = 9.$$

Computing the new BFS $\bar{x}$ and the new value of the objective function $f(\bar{x})$:

$$\bar{x} = x + \theta^* d = (2,5,0,9,0)^T,\ f(\bar{x}) = -91 < f(x).$$

Change of the basis:

$$\bar{J}_B = (J_B \setminus \{3\}) \cup \{4\} = \{4,1,2\},\ \bar{J}_N = \{3,5\}.$$

Fourth iteration:
We have

$$J_B = \{4,1,2\},\ J_N = \{3,5\},\ x^T = (2,5,0,9,0),\ f(x) = -91.$$
$$A_B = \begin{pmatrix} 0 & -1 & 1 \\ 1 & 1 & -1 \\ 0 & 1 & 2 \end{pmatrix},\ A_N = \begin{pmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix},$$
$$c_B = (0,-1,-2)^T,\ c_N = (0,0)^T.$$

Computing the matrices Z, Q and the vector l:

$$Z = \begin{pmatrix} -1 & 0 \\ 2/3 & -1/3 \\ -1/3 & -1/3 \\ 1 & 0 \\ 0 & 1 \end{pmatrix},\ Q = \begin{pmatrix} -14/9 & -2/9 \\ -2/9 & -8/9 \end{pmatrix},$$
$$l = (22/3,\ 37/3)^T.$$

We compute θ_{i_k} and θ_0^k, $k \in \{1,2\}$:

$$\theta_{i_1} = \min\{9,+\infty,15\} = \theta_1^1 = 9,\ \theta_0^1 = 66/7,$$
$$\theta_{i_2} = \min\{+\infty,6,15\} = \theta_2^2 = 6,\ \theta_0^2 = 111/4.$$

So,
$$K^* = \{k \in K : \theta_0^k < \theta_{i_k}\} = \emptyset.$$

Therefore, the local optimal BFS and the corresponding objective function value are:

$$x^* = (2,5,0,9,0)^T,\ f(x^*) = -91.$$

Let us remark that the local minimizer found by our algorithm in this example is also global.

V. Numerical experiments

In order to compare our algorithm with the algorithm of Rusakov, we have developed an implementation with MATLAB2017a on a PC with a processor Intel Pentium Dual-Core 2.20 GHz and 4 Go of RAM. The Rusakov's algorithm is a branch and bound algorithm which uses the Tuy cut. It is implemented in his free software [5][11]. Note that the Rusakov's test problems have one constraints and n bounded variables and the available free version of the software is limited to one constraint and 20 bounded variables, that is why we have done numerical experiments on small size test problems. Numerical results on Rusakov's test problems are presented in Table I.

In order to test the performance of our algorithm on medium size test problems, we have generated concave quadratic test problems with known global optimum using the generation procedure presented in [15], see Table II.

The notations in the first row of Tables I and II : m, n, $niter_j$, opt_j, $cput_j$ represent respectively number of constraints, number of variables, number of iterations, the optimum found, the CPU time for Algorithm j, where Algorithm 1 is the SASIR algorithm and Algorithm 2 is the Rusakov's algorithm. In Table II, $gopt$ represents the global optimum of the corresponding test problem and "Mean" represents the average CPU time and the average number of iterations for each problem size.

TABLE I: NUMERICAL RESULTS ON RUSAKOV'S TEST PROBLEMS.

n	Algorithm 1 (SASIR)			Algorithm 2 (Rusakov)	
	$niter_1$	opt_1	$cput_1$ (s)	opt_2	$cput_2$ (s)
5	4	-11.0694	0.0031	-11.0694	1.1298
10	6	-40.8382	0.0041	-40.8382	2.0481
15	9	-88.9456	0.0051	-88.9456	6.1074
18	10	-127.0726	0.0075	-127.0726	9.7121
20	11	-156.1234	0.0092	-156.1234	11.5236

TABLE II: NUMERICAL RESULTS ON RANDOMLY GENERATED TEST PROBLEMS.

Prob	$m \times n$	$niter_1$	opt_1	$gopt$	$cput_1$ (s)
1	100×120	40	-9.5894	-6.2500	0.1439
2	100×120	2	-4.3225	-0.0625	0.0217
3	$\mathbf{100 \times 120}$	**30**	**-1.5625**	**-1.5625**	**0.2773**
4	$\mathbf{100 \times 120}$	**39**	**-4.0272**	**-4.0272**	**0.5287**
5	100×120	3	-2.2500	-0.1600	0.0247
6	100×120	6	-1.9492	-1	0.0330
7	100×120	14	-2.9417	-2.2500	0.0564
8	100×120	2	-23.6580	-0.3600	0.0240
9	$\mathbf{100 \times 120}$	**18**	**-3.0625**	**-3.0625**	**0.0552**
10	$\mathbf{100 \times 120}$	**20**	**-1.7778**	**-1.7778**	**0.0786**
Mean	100×120	37.2	-	-	0.0988
11	200×240	2	-12.2500	-0.1600	0.0426
12	200×240	2	-11.6224	-0.3600	0.0394
13	200×240	18	-10.5901	-4.0000	0.4594
14	200×240	2	-9.8212	-0.4900	0.0386
15	200×240	2	-15.4605	-0.3600	0.0387
16	200×240	2	-57.4326	-0.0400	0.0389
17	$\mathbf{200 \times 240}$	**27**	**-4**	**-4**	**0.5332**
18	200×240	2	-11.3498	-0.6400	0.0454
19	200×240	5	-39.3321	-0.1600	0.0824
20	200×240	2	-1.4400	-0.2304	0.0409
Mean	200×240	6.4	-	-	0.1359

We have started our algorithm by the extreme point, with objective function equal to zero. Table I shows clearly that our algorithm has succesfully found the global optimum for the Rusakov's test problems. Moreover, the simplex algorithm with the smallest index rule outperforms the branch and bound algorithm of Rusakov in terms of CPU time. Table II shows that our algorithm has found the global optimum for 5 generated test problems in a short CPU time (test problems 3,4,9,10 and 17). However, for other test problems, our algorithm gives a local minimizer which can be used for the initialization of the global optimization algorithms.

VI. Conclusion

In this work, we have adapted the simplex algorithm of linear programming for finding a local optimum of a concave quadratic function subject to linear and nonnegativity contraints. In order to stop the algorithm, we suggested a simple sufficient and necessary condition for local optimality of the current extreme point. Numerical experiments on Rusakov and randomly test problems show that our algorithm is very fast and can find the global optimum for some problems. In a future work, we will combine our algorithm with an existing global optimization algorithm in order to find the global optimum of medium and large-scale concave quadratic programming problems.

References

[1] E. L. Lawler, "The Quadratic Assignment Problem," Management Sci., vol. 9, no. 4, 1963, pp. 586–599, ISSN: 1526-5501.

[2] A. I. Rusakov, "An Improved Reduction Algorithm to Check Hypotheses for the Multicollinear Regression Model," Automation and Remote Control, vol. 62, no. 5, 2001, pp. 762–771, ISSN: 1608-3032.

[3] H. Tuy, "Concave Programming under Linear Constraints," Dokl. Akad. Nauk SSSR English translation in Soviet Math. Dokl., vol. 5, 1964, pp. 1437–1440, ISSN: 1531-8362.

[4] R. Horst, "An Algorithm for Nonconvex Programming Problems," Math. Programming, vol. 10, 1976, pp. 312–321, ISSN: 1436-4646.

[5] A. I. Rusakov, "Concave programming under the simplest linear constraints," Computational Mathematics and Mathematical Physics, vol. 43, no. 7, 2003, pp. 951–960, ISSN: 0044-4669.

[6] H. Konno, "Maximization of a Convex Quadratic Function under Linear Constraints," Math. Programming, vol. 11, 1976, pp. 117–127, ISSN: 1436-4646.

[7] K. L. Hoffman, "A Successive Underestimating Method for Concave Minimization Problems," Ph.D. thesis, The George Washington University, 1975.

[8] H. Konno, C. Gao, and I. Saitoh, "Cutting plane/tabu searh algorithms for low rank concave quadratic programming problems". Journal of Global Optimization, vol. 13, no. 3, 1998, pp. 225–240, ISSN: 1573-2916.

[9] D. G. Luenberger and Y. Ye, Linear and nonlinear programming. Third edition, New York, NY, USA: Springer-Verlag, 2008, ISBN: 978-0-387-74502-2.

[10] G. B. Dantzig, Linear Programming and Extensions. Princeton University Press, Princeton, N.J., 1998, ISBN: 978-0-691-05913-6.

[11] A. I. Rusakov, "Concave software manual," URL: http://www.rusakov.donpac.ru/index1.htm [accessed: 2018-05-02].

[12] N. Ikheneche, Support Method for the Minimization of a Convex Quadratic Function. Master thesis, University of Bejaia (in french), 2004, (in french).

[13] M. Bentobache and M. O. Bibi, Numerical Methods of Linear and Quadratic Programming: Theory and Algorithms. French Academic Editions, Germany, 2016, ISBN: 978-3-8416-4112-0 (in french).

[14] A. Chikhaoui, B. Djebbar, A. Belabbaci, and A. Mokhtari, "Optimization of a quadratic function under its canonical form". Asian journal of applied sciences, vol. 2, no. 6, 2009, pp. 499–510, ISSN: 1996-3343.

[15] N. V. Thoai, "On the construction of test problems for concave minimization algorithms," Journal of Global Optimization, vol. 5, no. 4, 1994, pp. 399–402, ISSN: 1573-2916.