On Subproblem Solving in an Augmented Lagrangian Algorithm for LP with Upper Bounds

Eligius M. T. Hendrix[1(✉)], Pablo Guerrero-García[2], and Ana Maria A. C. Rocha[3]

[1] Computer Architecture, Universidad de Málaga, Spain and Wageningen University, 29071 Málaga, Spain
eligius@uma.es
[2] Applied Mathematics, Universidad de Málaga, 29071 Málaga, Spain
[3] Centro ALGORITMI/LASI, Universidade do Minho, Campus de Gualtar, 4710-057 Braga, Portugal

Abstract. The augmented Lagrangian method has been well investigated for Linear Programming problems. We study LP problems where variables are nonnegative and have upper bounds. The resulting augmented Lagrangian subproblems require solving convex Quadratic Programming (QP) problems over a bound constrained area. We investigate the adaptation of three classical methods to solve the subproblems. We describe the algorithms and illustrate them with a small example.

Keywords: linear programming · augmented Lagrangian · bound constraints · quadratic programming

1 Introduction

Motivated by energy management problems [10], we are interested in Linear Programming (LP) problems with equality constraints and upper bounds on the variables. The upper bounds $\overline{x}$ on the variables in our application are caused by maximum storage and charging and discharging of batteries. In abstract mathematical terms this implies a problem of the form:

$$\max c^T x, \quad \text{s.t.} \quad Ax = b,\ x \in [0, \overline{x}] \subset \mathbb{R}^n, \tag{1}$$

where $c \subset \mathbb{R}^n$ the objective function gradient, b the right hand side, $\overline{x} \subset \mathbb{R}^n_+$ a vector of upper bounds and A a (sparse) $m \times n$ matrix.

Such a form includes upper bounds in what is called in several handbooks e.g. [1,2] the standard form LP; $\{\max c^T x | Ax = b, x \geq 0\}$. To avoid confusion,

This work has been supported by FCT Fundação para a Ciência e Tecnologia within the R&D Unit Project Scope UID/00319/2025 - Centro ALGORITMI (ALGORITMI/UM) https://doi.org/10.54499/UID/00319/2025.

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2027
O. Gervasi et al. (Eds.): ICCSA 2026 Workshops, LNCS 16766, pp. 206–216, 2027.
https://doi.org/10.1007/978-3-032-30539-8_14

it should better be called the standard equality form. For this form, [3] and [4] investigated augmented Lagrangian (AL) algorithms to find a feasible point. Motivated by these studies, we constructed an AL algorithm to solve LP problem (1) with upper bounds on the variables and applied it in [10] comparing the results with the simplex method. In the application, we employed a generic constrained NLP algorithm to solve subproblems. However, specifically we are dealing with a bounded convex quadratic program. Our research question in this paper deals with an investigation of traditional methods that can solve these type of problems.

This paper is organized as follows. Section 2 introduces the setting of the augmented Lagrangian algorithm for the upper bounded LP. Sections 3, 4 and 5 discuss each a basic algorithm that can be adapted to solve the subproblems. Section 6 summarizes our findings.

2 Augmented Lagrange Embedding

We consider the upper bound extended equality LP model (1). By adding the linear constraints with Lagrange multipliers $\lambda \in \mathbb{R}^m$ to the objective, we obtain the equivalent Lagrangian problem

$$\begin{array}{l} \max_{0\le x\le \overline{x}}[c^T x, \quad \text{s.t.} \quad Ax = b] \\ \min_\lambda \max_{0\le x\le \overline{x}}[\mathcal{L}(x,\lambda) := c^T x + \lambda^T(Ax-b)] \\ \max_{0\le x\le \overline{x}}[c^T x + \min_\lambda\{\lambda^T(Ax-b)\}]. \end{array} \tag{2}$$

The analysis illustrates that if no feasible point x with $Ax = b$ exists, the Lagrangian is unbounded. The augmented Lagrangian approach we focus on, is to add a quadratic penalty to handle the equality constraints and iteratively solve the convex quadratic programming subproblem

$$\max_{0\le x\le \overline{x}}[c^T x + \lambda^T(Ax-b) - \frac{\rho}{2}(Ax-b)^T(Ax-b)], \tag{3}$$

where ρ is a penalty parameter which is updated during iterations. This problem is equivalent to a convex Quadratic programming (QP) problem, which we will write as a minimization problem

$$x^* \in \text{argmin}_{0\le x\le \overline{x}}[f(x) := r^T x + \frac{1}{2}x^T G x], \tag{4}$$

taking $G = \rho A^T A$ and $r = -A^T(\lambda + \rho b) - c$. Notice that the matrix $A^T A$ is positive semi-definite.

Algorithm 1 is the AL setting we investigate for optimizing (1). It first generates a feasible point x_0 by minimizing the square of the Euclidean norm of the slack. This is a so-called bounded least squares problem (LS). Notice that the LP has either no feasible point, one minimizer, or infinitely many; this corresponds to the argmin set. If no solution exists, then the minimum is not zero and the algorithm ends. Otherwise, the algorithm proceeds with an iterative process with x_0 and an estimate of the Lagrange multipliers λ_0 as starting point.

Algorithm 1. Augmented Lagrangian $(A, b, c, \overline{x}, \omega, \rho_0, \epsilon)$

Require: A: matrix
b, $\overline{x}$, c: vectors
$\rho_0 > 0$: penalty parameter, $\omega > 1$: update parameter, ϵ: accuracy
1: Find $x_0 \in \text{argmin}_{x\in[0,\overline{x}]} \frac{1}{2}(Ax-b)^T(Ax-b)$ ▷ bounded least squares
2: **if** $\|Ax_0 - b\| \geq 10^{-6}$ **then** ▷ no feasible point exists
3: **return** "No feasible point exists"
4: **else**
5: Initialize $k = 0, \rho_0 > 0$, $\lambda_0 = 0$ ▷ x_0 as starting value for the AL optimization
6: **repeat**
7: $x_{k+1} \in \text{argmax}_{x\in[0,\overline{x}]} \left(c^T x + \lambda_k^T(Ax-b) - \frac{\rho_k}{2}(Ax-b)^T(Ax-b)\right)$
8: $\lambda_{k+1} = \lambda_k - \rho_k(Ax_{k+1} - b)$
9: $\rho_{k+1} = \omega\rho_k$ ▷ update ρ
10: $k = k + 1$
11: **until** $\|x_k - x_{k-1}\| < \epsilon$
12: **return** x_k

In our preliminary experiments for the energy management problem using the AL algorithm, we solved subproblems using the FMINCON solver from MATLAB®, specifically the active set solver. However, such an approach is a generic algorithm for solving general NLP problems, whereas problem (4) has specific characteristics. Therefore, this paper investigates several algorithms from literature that are more specific and that can be adapted to solve (4), where positive semi-definite matrix G and vector r are given.

Exploiting the bound constraints, notice that the so-called Karush-Kuhn-Tucker (KKT) necessary conditions [6] for optimality in order to certify the necessary conditions for a found point to be an optimal solution of (4) are relatively easy. For a minimization problem, the negative of the objective function gradient should be a nonnegative linear combination of the gradients of the active constraints, e.g. [5]. For a bound constrained problem given by $x_j \in [0, \overline{x}_j]$, active (binding) constraints in an optimizer point x^* are characterized by $x_j^* = 0$ or $x_j^* = \overline{x}_j$, see also [7]. One can derive the following result.

Proposition 1. *Let x^* be a minimum point of (4) and define its objective function gradient as $g = Gx^* + r$. Then for $x_j^* = 0$, we have $g_j \geq 0$, for $x_j^* = \overline{x}_j$ we have $g_j \leq 0$ and for $0 < x_j^* < \overline{x}_j$ holds $g_j = 0$.*

We elaborate three specific algorithms from literature and illustrate them with a small instance. We first develop an algorithm to generate a solution of the bounded least squares (LS) problem in Line 1 of Alg. 1 to be presented in Sect. 3 called a projected gradient descent algorithm. We can adapt this algorithm to solve (4). In Sect. 4, this is followed by the description of a generic classical algorithm based on gradient projection and we adapt that to the box constraints. Finally, in Sect. 5 we study a specific gradient projection algorithm to solve a bound constrained QP (BCQP) problem. We first introduce a small example instance for which we sketch the steps of the algorithms.

Example 1. Consider an LP problem defined by

$$\max\{2x_1 + x_2\} \quad \text{s.t.} \quad 3x_1 + 2x_2 = 4,\ 0 \le x_j \le 1,\ j = 1, 2. \tag{5}$$

The optimal point is $x^* = (1, 0.5)^T$ with objective function value $c^T x^* = 2.5$. For the KKT conditions for this LP problem, it can be seen that the gradient c is a positive combination of the gradients of the active constraints $3x_1 + 2x_2 = 4$ and $x_1 \le 1$, specifically $(2, 1)^T = 0.5(3, 2)^T + 0.5(1, 0)^T$.

3 Projected Gradient Descent to Solve a Bounded LS

Although [9] speaks about projected gradient, what is projected is not the gradient, but a point following from a gradient step onto a bound constrained area. This is usually called projected gradient descent. We base our algorithm on [9] to find a solution of

$$\min_x \frac{1}{2}\|Ax - \beta\|^2, x_j \in [0, \overline{x}_j], j = 1, \ldots, n. \tag{6}$$

The algorithm in [9] is aiming at undetermined systems with $m \gg n$, where a solution $Ax = \beta$ does not exists, but should be approximated as good as possible. The problem to be solved in Line 1 of Algorithm 1 intends to find a feasible point of (1), where $n \gg m$. Notice that finding an optimal solution of (6) is equivalent to solving QP problem (4) where we take $G = A^T A$ and $r = -A^T \beta$.

The basic steps of the algorithm as depicted in Algorithm 2 is to use a step-size based on $L := \text{maxeig}(A^T A)$ in the direction $-\nabla f$. L is the maximum eigenvalue of $A^T A$. After taking the step, the result is projected on the box as feasible region. Projection here of a point z means that $P(z)$ takes component-wise:

$$P_j(z) = \max[\min(z_j, \overline{z}_j), 0].$$

Algorithm 2. Projected gradient descent $(A, \beta, \overline{x}, x_0)$

Require: A: matrix
β, $\overline{x}$: vectors, x_0: starting point
1: Compute L as $\text{maxeig}(A^T A)$
2: Initialize $k = 0$
3: **repeat**
4: $z = x_k - \frac{1}{L}\nabla f(x_k)$
5: $x_{k+1} = P(z)$
6: $k = k + 1$
7: **until** x_k is a stationary point or KKT point
8: **return** x_k

By taking $\beta = b$ and considering $G = A^T A$ and $r = -A^T b$ in (4), the gradient of $f(x)$ in (6) is given by

$$\nabla f(x) = A^T(Ax - b). \tag{7}$$

After using this gradient to find a feasible point in Line 1 of Algorithm 1, we can change the the quadratic objective in (4) and use it for iteratively solving the augmented Lagrangian steps in Line 7 of Algorithm 1, i.e.

$$\nabla f(x_k) = -A^T(\lambda_k - \rho_k b) - c + \rho_k A^T A x_k. \tag{8}$$

Example 2. We use the data of Example 1 setting $\beta = b$, compute $L = 13$ and set the initial value on $x_0 = (0,0)^T$. The corresponding least squares objective function value is $f(x_0) = \frac{4}{2} = 2$. The gradient in x_0 is $\nabla f(x_0) = -(12,8)^T$ providing the iterate $x_1 = \frac{1}{13}(12,8)$, which falls in the bound constraints and fulfills the equality constraint $Ax_1 = b$. This means that $\nabla f(x_1) = 0$, so it is a stationary point causing Algorithm 2 to stop.

We can now return to Algorithm 1 and use the outcome x_1 as feasible starting point $x_0 = \frac{1}{13}(12,8)$ in lines 6 to 11 setting initially $\rho_0 = 1$ and multiplier vector $\lambda_0 = 0$. For the AL optimization in Line 7, we now use the projected gradient descent Algorithm 2 using (8) to compute the gradient.

The gradient in x_0 is $\nabla f(x_0) = -(2,1)^T$, such that we want to increase both components of x. The gradient descent step leads to $z = x_0 - \frac{1}{13}\nabla f(x_0) = \frac{1}{13}(14,9)$, which is outside of the box. Projecting z provides point $(1, \frac{9}{13})^T \approx (1, 0.6923)^T$. This point is not a KKT point, because the second component of gradient is not zero. Notice that the KKT point the algorithm should converge to is $(1, 0.75)^T$. However, due to the fixed step size, it can be observed that the iterates approximate very slowly to this point. In fact, it takes the algorithm 92 iterations to reach a machine accuracy KKT point. Figure 1 shows 6 iterates of the algorithm.

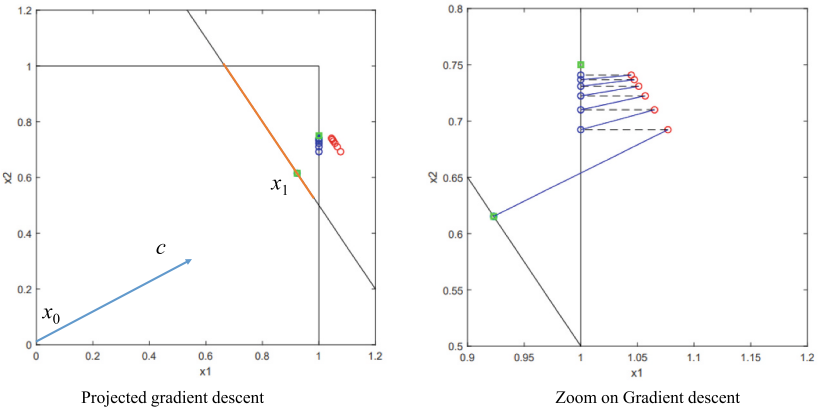

Fig. 1. At the left, the feasible set in orange with iterates of Algorithm 1 (Color figure online) as green squares. In red dots the gradient descent steps z, which are projected as blue points onto the feasible set. At the right a zoom and providing with (blue) lines the gradient descent steps and with black dashed lines the projection of 6 iterates.

As said, in one iteration a feasible point x_0 is reached. After that, the iterates zigzag between gradient descent and projection. This illustrates the slow convergence of the projected gradient descent algorithm.

The observation of slow convergence is also proven in theoretical sense; [9] shows that Alg. 2 converges with a speed proportional to $\frac{1}{k}$. Notice that such a convergence speed is less than linear.

4 A Gradient Projection on Linear Constraints

The original gradient projection concept was introduced by Rosen in 1960, see [11]. We investigate its adaptation to solve (4). It is based on projecting the gradient on the active constraints. In principle, this requires an inverse operation to project the gradient at each iteration where a new active constraint is found. Let M be the matrix of active constraints and g be the gradient in an iterate x_k, then

$$p = (I - M^T(MM^T)^{-1}M)g \tag{9}$$

is an increasing direction of the objective function in x_k. For minimizing the objective, the idea is to do a line-search in the direction of $-p$ taking care of the inactive constraints. Such a method appears unattractive when we are dealing with general linear constraints, as it requires a matrix inverse at each iteration. For problem (4), we are dealing with simple lower and upper bounds. That means, we can adapt the projection of the gradient in a point x taking component-wise:

$$p_j = \begin{cases} 0 & \text{if } x_j = 0 \text{ and } g_j > 0, \\ 0 & \text{if } x_j = \overline{x}_j \text{ and } g_j < 0, \\ g_j & \text{else.} \end{cases}$$

Stepping into the direction $-p$ along the line $x_k - \alpha p$, we have the directional derivative $dp = -r^T p - x_k^T G p$ for the convex quadratic objective. It may be clear that if $dp \geq 0$ apparently x_k cannot be improved in direction $-p$ and we found the best point. We adapt the generic approach to a quadratic objective which has the line-search as minimizing a quadratic one-dimensional function. This means we compute the directional second derivative $h = p^T G p$ and determine the corresponding step $\alpha^* = -\frac{dp}{h}$, taking care of $h > 0$. The maximum step size in Line 8 of Algorithm 3 is determined by the inactive constraints and calculated as

$$\overline{\alpha} := \min \left\{ \frac{x_j}{p_j} : p_j x_j > 0; \frac{x_j - \overline{x}_j}{p_j} : p_j(x_j - \overline{x}_j) > 0 \right\}. \tag{10}$$

Example 3. We take from Example 2 the reached feasible point $x_0 = \frac{1}{13}(12, 8)$ to start the optimization of Algorithm 1 Line 7, now using Algorithm 3. This means, we have as input $G = \rho_0 A^T A = \begin{pmatrix} 9 & 6 \\ 6 & 4 \end{pmatrix}$ and $r = \begin{pmatrix} -14 \\ -9 \end{pmatrix}$, such that $g = -(2, 1)^T$. As no box constraint is active, the projected gradient is $p = g$. In

Algorithm 3. Rosen gradient projection $(G, r, \overline{x}, x_0)$

Require: G: positive semi-definite matrix
$r, \overline{x}$: vectors, x_0: feasible starting point
Initialize $k = 0$
repeat
  $g = Gx_k + r$
  Determine projected gradient p and maximum step size $\overline{\alpha}$ according to (10)
  Compute directional derivative $dp = -(r^T p + x_k^T Gp)$
  Compute second derivative $h = p^T Gp$ and step-size $\alpha^* = \frac{-dp}{h}$
  if $dp < 0$ **then**
    $x_{k+1} = x_k - \min\{\alpha^*, \overline{\alpha}\}p$
  else
    $x_{k+1} = x_k$
  $k = k + 1$
until x_k is a stationary point or KKT point
return x_k

order not to get out of the box in direction $-g$, the maximum step is $\overline{\alpha} = \frac{1}{26}$, whereas $\alpha^* = \frac{5}{64}$. The corresponding next iterate $x_1 = (1, \frac{17}{26})^T \approx (1, 0.6538)^T$ is not a KKT point. Notice the numerical difference with x_1 in the projected gradient descent approach (Algorithm 2), which projects after taking a step. As the first component of x_1 is on the upper bound and both derivatives are negative, the projected gradient only contains the second component as nonzero, $p = -(0, 0.3846)^T$. Line-search to determine the step-size α^* provides the KKT point $x_2 = (1, 0.75)^T$. This is the optimum of the AL function for $\rho_0 = 1$ and $\lambda_0 = 0$ in Algorithm 1. The iterates are depicted in Fig. 2. In green are the main iterates of the AL algorithm.

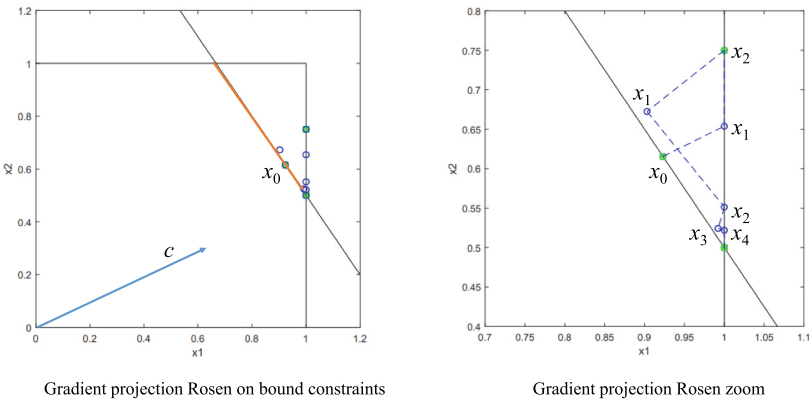

Fig. 2. Feasible set in orange with AL iterates of Algorithm 1 (Color figure online) in green squares. At the left, iterates of Algorithm 3 in blue. At the right, a zoom and the steps depicted by lines.

Following Algorithm 1, we should now update Lagrange multiplier $\lambda_1 = \lambda_0 - \rho_0(Ax_1 - b) = -0.5$ and increase $\rho_1 = 2$. We call Algorithm 3 again for Line 7 of Algorithm 1 with input $G = \rho_1 A^T A = \begin{pmatrix} 18 & 12 \\ 12 & 8 \end{pmatrix}$ and $r = \begin{pmatrix} -24.5 \\ -16 \end{pmatrix}$. So, $g = \begin{pmatrix} 2.5 \\ 2 \end{pmatrix}$. Algorithm 3 now requires 5 steps to reach the KKT point $x^* = (1, 0.5)^T$ which coincides with the optimal point of the LP problem.

5 A Gradient Projection Method to Solve a BCQP

We focus on a gradient projection method for the bound constrained quadratic programming (BCQP) problem outlined in [8]. The initial ideas can be found in [7]. It has been developed to find specifically a minimum point of $\frac{1}{2}x^T Gx + r^T x$ over bound constraints. Substituting $G = \rho A^T A$ and $r = -A^T(\lambda + \rho b) - c$, it may be used to solve Line 7 in Algorithm 1. In contrast to the method introduced by Rosen (see Sect. 4), the search direction is bent along the active box constraints to compute a so-called Cauchy point such that it stays in the box constrained area. Only after a minimum point is found along the search trajectory, we compute a new gradient. To describe the trajectory, [7] introduced the concept of breakpoints. Its computation can be derived from the description in [8].

The algorithm is depicted in Algorithm 4. In addition to the usual iterates x_k we have an inner loop with iterates z_i along the breakpoints of the trajectory starting at x_k. The algorithm follows iterative gradient driven trajectories up to finding an optimal point on them. It identifies breakpoints t_j in the direction of the negative gradient where bound constraints become active. Given the gradient in $g = Gx + r$ in iterate x, we find for each component $j = 1, \ldots, n$ the breakpoint

$$t_j = \begin{cases} \frac{x_j}{g_j} & \text{if } g_j > 0, \\ \frac{x_j - \bar{x}_j}{g_j} & \text{if } g_j < 0, \\ \infty & \text{else.} \end{cases} \tag{11}$$

We order the set of breakpoints removing duplicates starting (potentially adding) at 0 and ending at ∞, to generate the list $\tau_0 = 0, \tau_1, \tau_2, \ldots, \infty$. The ordered τ_i define a piecewise linear trajectory.

Each line-piece $[0, \tau_1], [\tau_1, \tau_2], \ldots, [\tau_i, \tau_{i+1}]$ corresponds to a (negative) projected gradient vector q with elements

$$q_j = \begin{cases} -g_j & \text{if } t_j > \tau_i, \\ 0 & \text{else.} \end{cases} \tag{12}$$

Starting at $z_0 = x_k$, the algorithm walks along the intervals. In the interval from point z_i to next point $z_{i+1} = z_i + (\tau_{i+1} - \tau_i)q$, we look for the minimum of the quadratic function. Define the directional derivative $dq = r^T q + z_i^T Gq$. It may be clear that if $dq \geq 0$, apparently z_i cannot be improved in direction q and we found the best point. If we can still improve, we compute the directional

Algorithm 4. Trajectory gradient projection for BCQP $(G, r, \overline{x}, x_0)$

Require: G: positive semi-definite matrix
$r, \overline{x}$: vectors, x_0: starting point
1: Initialize $k = 0$
2: **repeat**
3: $g = Gx_k + r$
4: Determine breakpoints $t_1, \ldots, t_n$ according to (11)
5: Obtain ordered line-piece intervals $[0, \tau_1], [\tau_1, \tau_2], \ldots$
6: Set $i = 0$, next_interval = true and $z_0 = x_k$ ▷ start the trajectory
7: **while** next_interval **do**
8: Compute direction q given τ_i according to (12)
9: Compute directional derivative $dq = r^T q + z_i^T Gq$
10: **if** $dq \geq 0$ **then** ▷ minimum for τ_i
11: $x_{k+1} = z_i$ and next_interval = false
12: **else**
13: Compute second derivative $h = q^T Gq$ and step $\alpha^* = \frac{-dq}{h}$
14: **if** $\tau_i + \alpha^* \leq \tau_{i+1}$ **then** ▷ interior minimum over line-piece
15: $x_{k+1} = z_i + \alpha^* q$ and next_interval = false
16: **else** ▷ look into next interval
17: $z_{i+1} = z_i + (\tau_{i+1} - \tau_i)q$
18: $i = i + 1$
19: $k = k + 1$
20: **until** x_k is a stationary point or KKT point
21: **return** x_k

second derivative $h = q^T Gq$ and determine the corresponding step $\alpha^* = -\frac{dq}{h}$. If $\alpha^* \leq \tau_{i+1}$ we take as minimum point $z_i + \alpha^* q$ and stop. Otherwise we go to the next interval $[\tau_{i+1}, \tau_{i+2}]$ to find a better point. The complete algorithm is sketched in Algorithm 4. Notice that [7] proved theoretically that under mild conditions, the algorithm converges in a finite number of steps.

We elaborate the algorithm for the example instance where it is used in the context of solving problem (3) in an iterative way.

Example 4. Like in Example 3, we take the outcome of Algorithm 2 $x_0 = \frac{1}{13}(12, 8)$, as starting point and for Algorithm 1 we set $\rho = 1$. This means the initial input of Algorithm 4 is $G = \rho_0 A^T A = \begin{pmatrix} 9 & 6 \\ 6 & 4 \end{pmatrix}$ and $r = \begin{pmatrix} -14 \\ -9 \end{pmatrix}$. Consequently, $g = \begin{pmatrix} -2 \\ -1 \end{pmatrix}$, such that we want to increase both components of x. In order not to get out of the box in direction $-g$ we get breakpoints $t_1 = \frac{1}{26}$ and $t_2 = \frac{5}{13}$. The minimum is found along the piecewise linear trajectory in between those breakpoints providing iterate $x_1 = (1, 0.75)$. Point $x^* = x_1$ is a KKT point according to Prop. 1, as $g = Gx^* + r = (-0.5, 0)^T$, i.e. $g_1 < 0$ for $x_1^* = \overline{x}_1$ and $g_2 = 0$ for $0 < x_2^* < \overline{x}_2$.

The outcome $x_1 = (1, 0.75)^T$ of the gradient projection approach for $\rho_0 = 1$ provides us with a minimum point of Line 7in Algorithm1. Following the

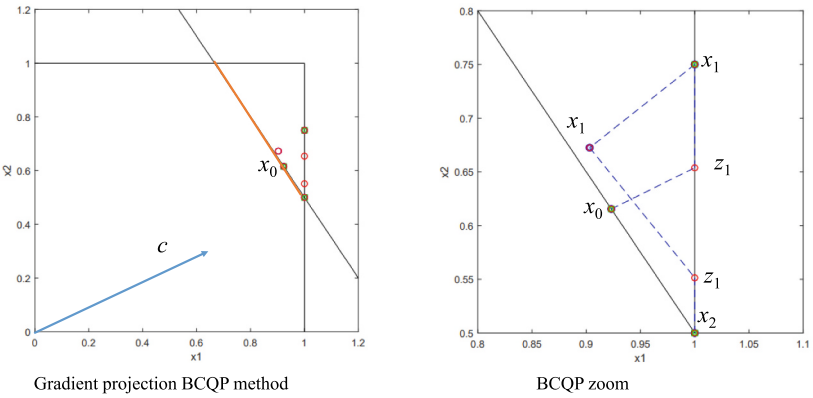

Fig. 3. Feasible set in orange with AL iterates of Algorithm 1 (Color figure online) in green squares. At the left, iterates of Algorithm 4 in blue. At the right, intermediate points z_j of Algorithm 4 in red dots.

iterations of Algorithm 1, we compute the update of the Lagrange multiplier $\lambda_1 = \lambda_0 - \rho(Ax_1 - b) = -0.5$ and increase $\rho_1 = 2$. This provides input G and r like in Example 3.

To decrease both components of x gives breakpoints $t_1 = \frac{2}{5}$ and $t_2 = \frac{3}{8}$. This provides an interior optimum over the first line-piece of $x = (0.9031, 0.6725)^T$ close to the feasible set of the LP with gradient $g = (-0.1739, 0.2173)^T$. Direction $q = -g$ goes in parallel along the equality constraint and leads to point $z_1 = (1, 0.5514)^T$. From there it bends the directory towards $z_2 = x_2 = (1, 0.5)^T$. This point corresponds to the optimal solution x^* of the original LP problem. The method of Rosen requires a few more steps in the end to reach the optimum. The difference is that it computes the gradient in each intermediate point. The steps of the AL algorithm and the intermediate trajectories are depicted in Fig. 3, where intermediate points z_i are given in red.

6 Conclusions

This paper focuses on an augmented Lagrangian approach to solve a linear programming equality problem with upper bounds. Such LP problems are relevant in energy management. It describes three algorithms that can be used to solve the bound constrained quadratic programming (BCQP) subproblem in the context of the augmented Lagrangian algorithm.

First we present a least squares projected gradient descent algorithm adapted to solve the subproblem. The second algorithm is the classical algorithm of Rosen adapted to the box constraints using an explicit line search. Finally, we study a specific gradient projection algorithm to solve a BCQP problem. A very small instance with only one equality illustrates some characteristics of the methods.

We found that the projected gradient descent method converges extremely slow due to the fixed step size. The classical gradient projection approach

of Rosen, solves the problem in a few iterations thanks to exploiting a one-dimensional quadratic line-search to determine the step-size. The specific BCQP approach also solves the small instance fast, but its scheme is more complicated due to following a piecewise linear trajectory. It requires even less steps for the small example than the gradient projection of Rosen. The computational burden of the last two approaches is relatively low.

In future research, we will study their behavior on practical instances.

References

1. Bertsimas, D., Tsitsiklis, J.N.: Introduction to Linear Optimization. Athena Scientific, Belmon, Massachusetts (1997)
2. Claassen, G.D.H., Hendriks, T.H.B., Hendrix, E.M.T.: Decision Science. Wageningen Academic Publishers, Wageningen (2007)
3. Galabova, I., Hall, J.: The 'idiot' crash quadratic penalty algorithm for linear programming and its application to linearizations of quadratic assignment problems. Optimization Meth. Softw. **35**(3), 488–501 (2020). https://doi.org/10.1080/10556788.2019.1604702
4. Guerrero-García, P., Hendrix, E.M.T., Rocha, A.M.A.C.: On augmented Lagrangian for LP feasibility. In: Gervasi, O., et al. (eds.) Computational Science and Its Applications – ICCSA 2025 Workshops, pp. 167–179. Springer Nature Switzerland, Cham (2026). https://doi.org/10.1007/978-3-031-97596-7_11
5. Hendrix, E.M.T., Tóth, G., B.: Introduction to Nonlinear and Global Optimization, Springer (2010)
6. Kuhn, H.W., Tucker, A.W.: Nonlinear programming. In: Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability, 1950, pp. 481–492. Univ. California Press, Berkeley-Los Angeles, Calif. (1951)
7. Moré, J.J., Toraldo, G.: Algorithms for bound constrained quadratic programming problems. Numerische Mathematik **55**(4), 377–400 (1989). https://doi.org/10.1007/BF01396045
8. Nocedal, J., Wright, S.J.: Numerical Optimization, 2e edn. Springer, New York, NY, USA (2006)
9. Polyak, R. A.: Projected gradient method for non-negative least square. Contemp. Math. **636**, 167–179 (2015). https://doi.org/10.1090/conm/636/12735
10. Rocha, A.M.A.C., Guerrero-García, P., Hendrix, E.M.T.: On battery storage management by sparse linear programming. In: Gervasi, O., et al. (eds.) Computational Science and Its Applications – ICCSA 2026 Workshop. Springer Nature Switzerland, Cham (2026). https://doi.org/10.1007/978-3-032-30539-8_13
11. Rosen, J.B.: The gradient projection method for nonlinear programming. part I. linear constraints. J. Soc. Indust. Appl. Math. **8**(1), 181–217 (1960). https://doi.org/10.1007/978-3-032-30539-8_13