University of Pisa

School of Engineering

Master of Science in Artificial Intelligence and Data Engineering

Symbolic and Evolutionary Artificial Intelligence

Internal Point Method
for Lexicographic MOQP problems

Student:
Tommaso Baldi

Academic year 2022-2023

Contents

1 Introduction

The objective of this project is to design and develop an Interior Point Method (IPM) variant to solve Lexicographic Multi-Objective Quadratic Programming (LMOQP) problems.

IPM represents a powerful class of optimization algorithms employed to solve a wide range programming problems. These methods have gained prominence due to their efficiency in addressing linear, quadratic, and nonlinear optimization problems with equality and inequality constraints. The fundamental principle of IPM involves navigating the interior of the feasible region, aiming to find an optimal solution while satisfying all constraints. Unlike traditional optimization methods, such as the Simplex algorithm, which traverse the boundary of the feasible region, IPM iteratively approach the optimum by moving through the interior, maintaining feasibility throughout the optimization process.

Lexicographic Multi-Objective Programming (LMOP) consists in the ordered prioritization of objectives according to a lexicographic order. In a typical LMOP scenario, a decision-maker is confronted with several objectives, each expressing a desired outcome, and the challenge lies in finding a solution that optimally balances these objectives. The lexicographic ordering establishes a clear hierarchy among these objectives, signifying their relative importance or precedence.

$$\begin{aligned} \min_{x} \quad & f_1(x), ..., f_n(x) \\ \text{s.t.} \quad & Ax = b, \\ & x \geq 0 \end{aligned}$$

Common ways to tackle this kind of problems are:

- **Scalarization**: this approach transforms the multi-objective problem into a single objective one by computing a weighted sum of the functions according to their priority (the most important one takes a higher weight, the lowest important a lower weight). Unfortunately, the correct a priori choice of the weights is unknown and depends on the function's range and on the optimal solution as well. The lack of any guarantee of correct convergence is not the only drawback, the choice of too different, i.e., very high and very small weights, may induce a loss of precision and numerical instabilities.

$$\begin{aligned} \min_{x} \quad & w_1 f_1(x) + ... + w_n f_n(x) \\ \text{s.t.} \quad & Ax = b, \\ & x \geq 0 \end{aligned}$$

- **Preemptive**: this approach optimizes one function at a time and uses its optimal value as an additional constraint for subsequent tasks. This idea guarantees convergence to optimality, but it is requires solving multiple optimization tasks (one for each function) with increasing complexity, which implies wasting computational resources. Furthermore, the nature of the problem can vary and become more complicated over time. For instance, in a Quadratic Programming (QP) problem, all the optimizations after the first one shall involve a convex constraint, turning them into quadratically constrained programming problems. As a result, the optimization needs a different, more complex, and less performing algorithm than the one it was supposed to be used at the beginning.

$$\begin{aligned} \min_{x} \quad & f_i(x) \\ \text{s.t.} \quad & f_j(x) = f_j^*(x) \forall j = 1, ..., i-1, \\ & Ax = b, \\ & x \geq 0 \end{aligned}$$

Recently has been develop a more advanced approach to face LMOP, the NA-IPM, [1] which leverage the non-Archimedean computations introduced in the Alpha Theory. The key idea is to scalarize the problem by adopting infinite and infinitesimal weights. Such a choice of weights guarantees the satisfaction of lexicographic ordering during the optimization of the single-objective constrained (and non-Archimedean) function. The results achieved by this approach are really good, however non-Archimedean numbers are too big to fit in a machine, so those numbers need to be approximated by the utilization of bounded algorithmic numbers (BAN) [3]. The utilization of BANs introduce numerical instabilities which need to be handled with proper techniques. This last issue can be solved by the utilization of dedicated hardware designed to accomplish operation with BANs, despite that in this work we are interested in methods and approaches which work on regular hardware.

$$\begin{aligned} \min_{x} \quad & f_1(x)\alpha^0 + ... + f_n(x)\alpha^{1-n} \\ \text{s.t.} \quad & Ax = b, \\ & x \geq 0 \end{aligned}$$

The aim of this project is to find a new approach, similar to the last one explained, which allows us to use the IPM to solve LMOQP problems without the utilization of BAN numbers. The idea of the existence of a possible new approach come from an observation of the NA-IPM, indeed if we looks at the iterations of this approach we can see that the final results are composed by standard numbers (without infinite or infinitesimal part), so probably there might be a way to reformulate the problem to achieve the same results by rewriting the problem and without the utilization of non-standard numbers.

2 Background

This section provides a brief refresher about the necessary background to understand this work.

2.1 Quadratic Programming

Quadratic programming (QP) is the task of solving a problem of the following form:

$$\begin{aligned} \min_{x} \quad & \frac{1}{2}x^T Q x + c^T x \\ \text{s.t.} \quad & Ax = b, \\ & x \geq 0 \end{aligned}$$

where $x \in \mathbb{R}$ is the unknown, $Q \in \mathbb{R}^{n \times n}$ is symmetric, $c \in \mathbb{R}^n$ together with Q forms the cost function, while $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$ are the feasibility constraints, $m, n \in \mathbb{N}$.

A famous algorithm to solve QP problems is the so-called interior point method (IPM). This work focuses on the predictor–corrector infeasible primal-dual IPM, which is broadly accepted today to be the most efficient one. The next subsection is devoted to resuming the algorithm in question.

2.2 Interior Point Method

First of all, the assumption that the problem is formulated according to the previous equation. If not, one needs to rewrite the problem by adding some slack variables, one for each constraint, without penalizing them in the cost function. Second, duality theory states that first-order optimality conditions for QP problems (also known as KKT conditions) are the ones reported in the following equation, where λ are the dual variables and s the dual slack variables (a slight modification which includes an additional parameter μ is provided in the second equation and will be discussed later).

$$\begin{cases} Ax = b \\ A^T\lambda + s - Qx = c \\ XS1 = 0 \\ (x, s) \geq 0 \end{cases}$$

$$\begin{cases} Ax = b \\ A^T\lambda + s - Qx = c \\ XS1 = \mu 1 \\ (x, s) \geq 0 \end{cases}$$

Third, iterative algorithms based on the Newton–Raphson method find the solution of systems as the one in the previous equation (positivity of x and s excluded) starting from a solution (x^0, λ^0, s^0) and repeatedly solving the following linear systems (second equation presents a slight modification with two additional parameters μ and σ which will be introduced in the next lines), where $rb = Ax - b$ and $rc = A^T\lambda + s - Qx - c$, while X and S are the diagonal matrices obtained by x and s.

$$\begin{bmatrix} -Q & A^T & I \\ A & 0 & 0 \\ S & 0 & X \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta\lambda \\ \Delta s \end{bmatrix} = \begin{bmatrix} -r_c \\ -r_b \\ -XS1 \end{bmatrix} \tag{1}$$

$$\begin{bmatrix} -Q & A^T & I \\ A & 0 & 0 \\ S & 0 & X \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta\lambda \\ \Delta s \end{bmatrix} = \begin{bmatrix} -r_c \\ -r_b \\ \sigma\mu 1 - XS1 \end{bmatrix} \tag{2}$$

This system is known as Newton's step equation, and once solved the approximated solutions of the KKT system are updated accordingly: $(x^{k+1}, \lambda^{k+1}, s^{k+1}) \leftarrow (x^k + \nu\Delta x, \lambda^k + \nu\Delta, s^k + \nu\Delta s)$ with $\nu \in (0, 1]$ opportunely chosen.

The IPM is designed as a path-following algorithm, i.e., an algorithm that tries to stay as close as possible to a specific curve during the optimization. Such a curve, also known as central path, is uniquely identified by all the duality-gap values $\mu \in (0, \frac{x^{0T}s^0}{n})$, where (x^0, λ^0, s^0) is the primal-dual starting point. Its role is to keep the intermediate solutions away from the feasible region's boundaries as much as possible. The points belonging to the central path are those satisfying conditions in the KKT system, which is a slight modification of the traditional one, for a given value of μ. The uniqueness of the curve is guaranteed whenever the feasible region admits at least one strictly feasible point, that is a primal-dual solution such that $(x, s) > 0$. Newton's step is now computed accordingly with the previous equation, where $\sigma\mu$ is the next duality gap to achieve, $\sigma \in (0, 1)$; furthermore, $\sigma\mu$ uniquely identifies the primal-dual point on the central path to the target. Since Equation (2) approximates (1) more and more closely as μ goes to zero, if the latter converges to anything as $\mu \to 0$ (and it does if the feasible region is bounded), then IPM eventually converges to an optimal solution of the QP problem.

In this work the predictor-corrector version of the IPM is used. In this version the Newton's direction computation is split in two stages:

- *Predictor step*: the algorithm estimates the next iterate by predicting the direction in which the solution will move toward the optimum. This prediction is typically made using a centering parameter and a search direction calculated based on the current iterate and a search direction obtained from the Newton's method.

- *Corrector step*: This step refines the predicted solution to ensure feasibility and move closer to the optimal solution. It involves adjusting the solution iteratively using a correction process. This correction is designed to maintain the feasibility of the solution with respect to the equality and inequality constraints.

An exhaustive presentation of the IPM algorithm is out of the scope of this report, anyway here you can find the whole procedure which can be used as reference.

Algorithm 1: Infeasible Primal-Dual IPM with Predictor-Corrector

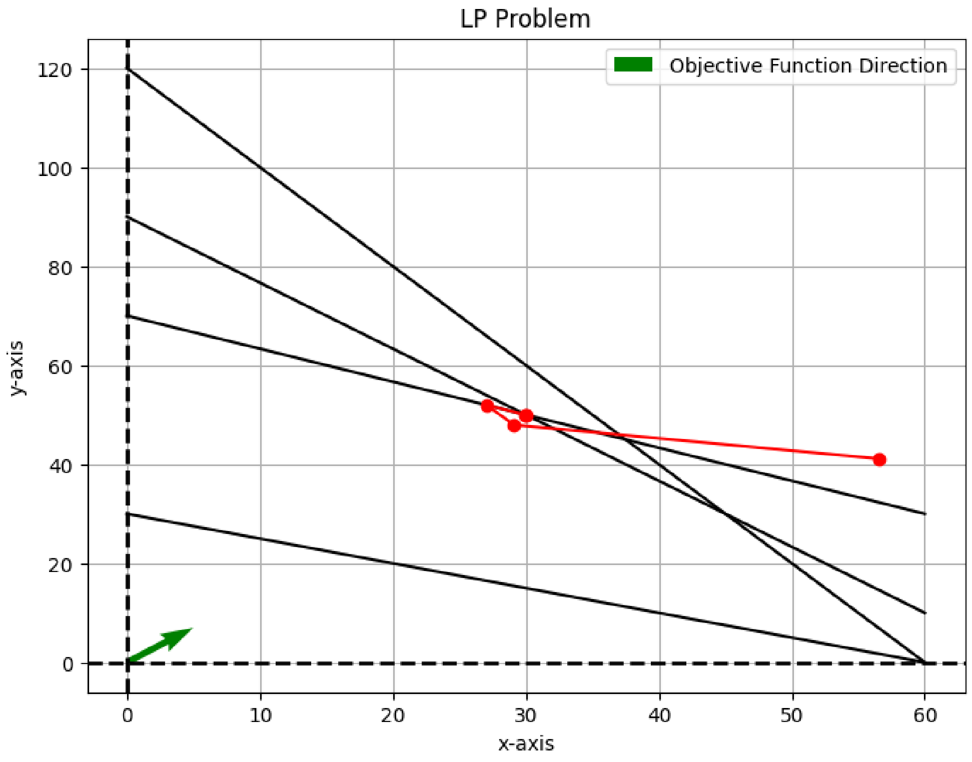
```
    Data: $A, b, c, Q, \epsilon, \omega, \max\_\mathrm{it}$
    Result: $x, \lambda, s, \mathrm{flag}$
 1  flag $\leftarrow$ False
 2  $x, \lambda, s \leftarrow \mathrm{starting\_point}(A, b, c, Q)$
 3  for $k \leftarrow 1$ to $max\_it$ do
        /* Compute residuals                                        */
 4      $r_b \leftarrow Ax - b$
 5      $r_c \leftarrow A^T\lambda + s - Qx - c$
 6      $r_\mu \leftarrow x \cdot s$
 7      $\mu \leftarrow \frac{r_\mu^T 1}{n}$
        /* KKT conditions satisfaction parameters                   */
 8      $\rho_1 \leftarrow \frac{\|r_b\|}{1+\|b\|}$
 9      $\rho_2 \leftarrow \frac{\|r_c\|}{1+\|c\|}$
10      $\rho_3 \leftarrow \frac{\mu}{1+\left|\frac{1}{2}x^T Qx + c^T x\right|}$
11      if $\rho_1 < \epsilon$ and $\rho_2 < \epsilon$ and $\rho_3 < \epsilon$ then
12          flag $\leftarrow$ True
13          return $x, \lambda, s, flag$
        /* Compute predictor direction                              */
14      $\Delta x_p, \Delta\lambda_p, \Delta s_p \leftarrow \mathrm{predict}(A, b, c, Q, r_b, r_c, r_\mu)$
15      $\nu_{pp} \leftarrow 0.99 \min\left(\max_{\bar\nu}\left\{\bar\nu \mid x + \bar\nu\Delta x_p \geq 0\right\}, 1\right)$
16      $\nu_{pd} \leftarrow 0.99 \min\left(\max_{\bar\nu}\left\{\bar\nu \mid s + \bar\nu\Delta s_p \geq 0\right\}, 1\right)$
17      $\nu \leftarrow \min(\nu_{pp}, \nu_{pd})$
18      $\tilde{x} \leftarrow x + \nu\Delta x_p$
19      $\tilde{s} \leftarrow s + \nu\Delta s_p$
20      $\mu^{new} \leftarrow \frac{\tilde{x}^T\tilde{s}}{n}$
21      $\sigma \leftarrow (\frac{\mu^{new}}{\mu})^3$
        /* Compute corrector direction                              */
22      $\Delta x_c, \Delta\lambda_c, \Delta s_c \leftarrow \mathrm{corrector}(A, b, c, Q, \sigma\mu 1 - \Delta x_p \cdot \Delta s_p)$
23      $\Delta x \leftarrow \Delta x_p + \Delta x_c$
24      $\Delta\lambda \leftarrow \Delta\lambda_p + \Delta\lambda_c$
25      $\Delta s \leftarrow \Delta s_p + \Delta s_c$
26      $\nu_p \leftarrow 0.99 \min\left(\max_{\bar\nu}\left\{\bar\nu \mid x + \bar\nu\Delta x \geq 0\right\}, 1\right)$
27      $\nu_d \leftarrow 0.99 \min\left(\max_{\bar\nu}\left\{\bar\nu \mid s + \bar\nu\Delta s \geq 0\right\}, 1\right)$
28      $\nu \leftarrow \min(\nu_p, \nu_d)$
        /* Compute target primal-dual solution                      */
29      $x \leftarrow x + \mu\Delta x$
30      $\lambda \leftarrow \lambda + \mu\Delta\lambda$
31      $s \leftarrow s + \mu\Delta s$
32      if $\|(x, s)\| > \omega$ then
            /* Divergence detected                                  */
33          return $x, \lambda, s, flag$
34  return $x, \lambda, s, flag$
```

2.2.1 Example

Here, you can find two examples of the IPM algorithm for both Linear Programming (LP) problem and Quadratic Programming (QP) problem.

$$\begin{aligned}
\min_{x} \quad & -10x_1 - 14x_2 \\
\text{s.t.} \quad & 2x_1 + x_2 \leq 120, \\
& 2x_1 + 3x_2 \leq 210, \\
& 4x_1 + 3x_2 \leq 270, \\
& x_1 + 2x_2 \geq 60
\end{aligned}$$

Iteration	x	$f(x)$
1	$56.5547, 41.1261$	-1141.3121
2	$29.1250, 47.9415$	-962.4304
3	$27.0382, 51.9481$	-997.6556
4	$29.9708, 50.0189$	-999.9727
5	$29.9997, 50.0001$	-999.9997

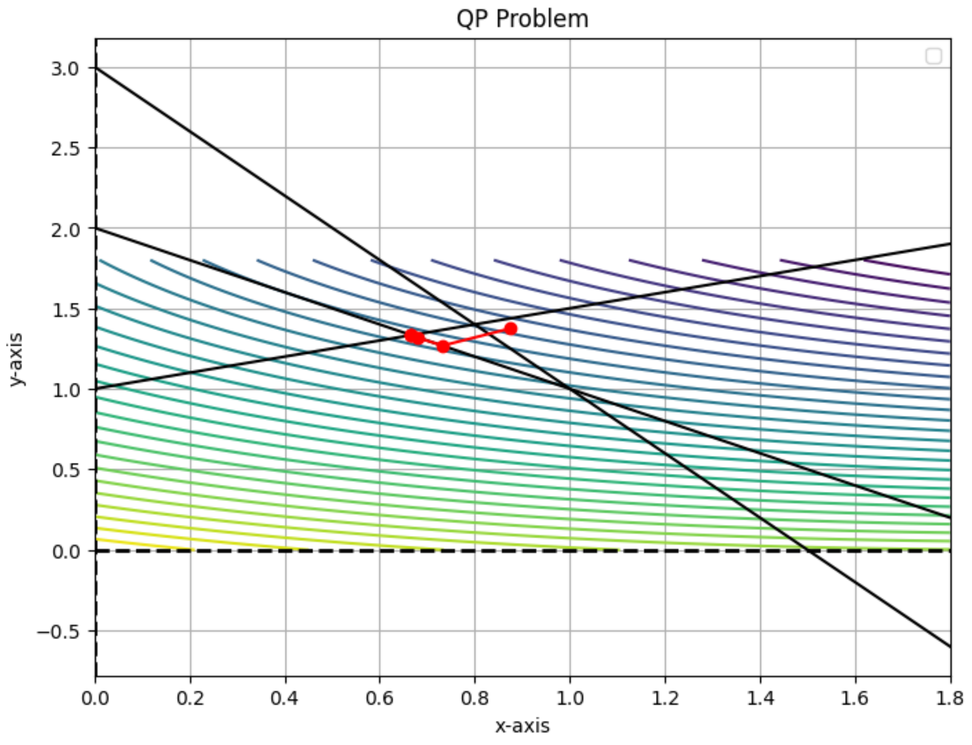


$$\begin{aligned}
\min_{x} \quad & \frac{1}{2}x^T Q x + c^T x \\
\text{s.t.} \quad & x_1 + x_2 \leq 2, \\
& -x_1 + 2x_2 \leq 2, \\
& 2x_1 + x_2 \leq 3
\end{aligned}$$

where:

$$c = \begin{bmatrix}-2 & -6\end{bmatrix}, Q = \begin{bmatrix}1 & -1 \\ -1 & 2\end{bmatrix} \tag{3}$$

Iteration	x	$f(x)$
1	$0.8754, 1.3754$	-8.9323
2	$0.7325, 1.2682$	-8.1267
3	$0.6801, 1.3174$	-8.1938
4	$0.66691.3330$	-8.2218
5	$0.6667, 1.3333$	-8.2222



3 Implementation

In this section we explain the ideas behind the two new naive approaches tested in this project, they are both extension of the preemptive approach and, they aim to overcome the main issues of the original approach. The idea of developing an extended version of the scalarization approach has been dropped, because in the following paper [2], the author demonstrates that if the preemptive approach reaches an optimal solution then there exist a combination of weights which reach the same result, but there is no way, for the moment, to compute this set of weights.

3.1 Preemptive approach with tangent

This method aims to use the preemptive approach with LMOQP problems keeping the constraints linear even when the solved objective function is quadratic. As previously presented, the preemptive approach optimizes one function at a time and uses its optimal value as an additional constraint for subsequent tasks, but if the objective function is quadratic then the problem turns into a quadratically constrained programming problems which need a slower algorithm to be solved. The idea is to find a way to add a new linear constraint with the same effects of the quadratic one.

The solution of this naive approach is to add as new constraint the tangent respect to the quadratic objective function computed in the optimal point found in that iteration. It is easy to demonstrate that if the optimal point reached by an objective function is close to the polyhedron face then the next objective function is forced to move within that face to optimize itself (1), but if the optimal point reached by an objective function lands on a vertex of the polyhedron we can early stop the algorithm because the other objective functions cannot move from that point (2). Early stopping condition can be checked by looking at the x vector, which includes the unknown variables and the slack variables, at the end of each optimization, if the number of values close to zero is greater or equal to the number of unknowns of the problem then we can asses that we had reached a vertex. Therefore, by adding as new constraint the tangent of the quadratic function computed in the optimal point we are obtaining the same effect of adding the whole quadratic function but keeping the linearity of the constraints which allows us to use the standard IPM. In the following pages you can find the pseudo code of the algorithm:

The algorithm can be further improved by checking if the new constraint is linear dependent respect to the existing ones with the Cauchy-Schwarz inequality, if it is the case, instead of adding a new constraint, we can set to zero the value of the slack variable relative to the dependent constraint obtaining the same result without increasing the dimension of the matrix A and the vector b. Unfortunately, this optimization is not free because the complexity of the control scales badly with the number of constraints.

The solution reached by the algorithm is equal to the ones achieved by NA-IPM, however this approach brings with it all the other drawbacks of the preemptive approach such as bad usage of the hardware. In the next section we are going to see another extension of the preemptive approach which leverages the same intuition of this algorithm, but it tries to solve the problem of the hardware utilization.

Algorithm 2: Preemptive approach with tangent

```
   Data: $A, b, objectives, \epsilon, \omega$, max_it
   Result: $x, \lambda, s$, flag
1  flag $\leftarrow$ False
2  $iter \leftarrow 0$
3  for $obj \leftarrow objectives$ do
4      $x, \lambda, s \leftarrow$ starting_point$(A, b, c, Q)$
5      $Q, c \leftarrow obj$
       /* Solve the QP problem                                          */
6      $x, \lambda, s, flag, n\_iter \leftarrow$ solve_QP $(A, b, c, Q, \epsilon, \omega, max\_it)$
7      $iter \leftarrow iter + n\_iter$
       /* Check if the objective function reached a vertex of the
          polyhedron                                                    */
8      if $early\_stopping\ (x)$ or $iter > max\_it$ then
9          return $x, \lambda, s, flag$
       /* Compute the gradient of the objective function                */
10     $g \leftarrow x^T Q + c$
11     $k \leftarrow g \cdot x$
       /* Add the new constraint                                        */
12     $A \leftarrow$ concatenate$(A, g)$
13     $b \leftarrow$ concatenate$(b, k)$
14 return $x, \lambda, s, flag$
```

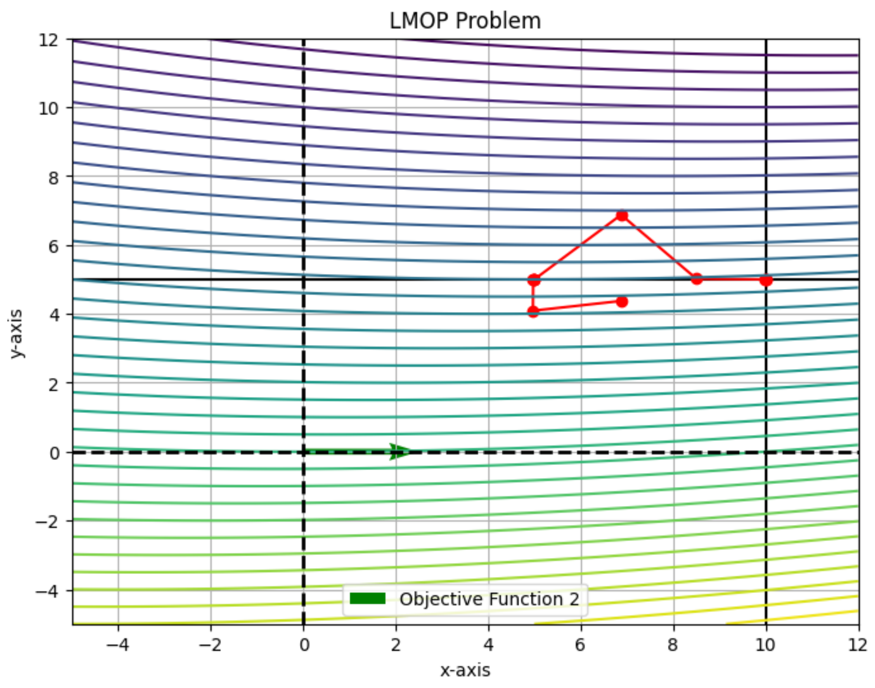


Figure 1: Example of the preemptive approach when the first quadratic objective lands on a face of the polyhedron.

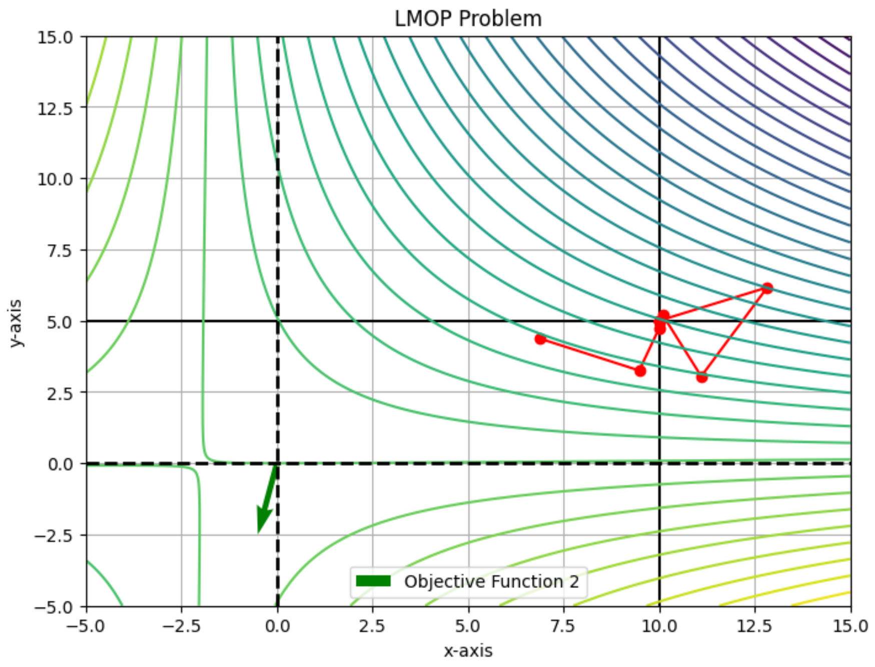


Figure 2: Example of the preemptive approach when the first quadratic objective lands on a vertex of the polyhedron. We can see that the second objective function is not able to move the solution away from the vertex.

3.2 Penalty approach

As the previous solution, also this one aim to solve the LMOQP problems extending the preemptive. However, this solution tries to solve one of the major issues of the preemptive approach which is the bad hardware utilization. Indeed, the traditional preemptive approach iterates over the objective functions adding constraints at the end of each iteration, this procedure turns out to waste the cache memory of the machine, because every time we instantiate a new optimization problem waiving the spatial locality of the information stored in the cache.

The idea to tackle this problem is to reformulate the LMOQP problem in a MOQP which evolve the Q matrix and c vector during the optimization by adding penalties to each objective functions coming from the previous ones. Specifically, we reformulate the objective function in the following way:

$$\begin{aligned} \min_{x} \quad & \sum_{i=1}^{n} z_i f_i(x) + \sum_{j=k+1}^{n} z_j \epsilon_k \bar{f}_k(x) \forall k = 1, ..., n \\ \text{s.t.} \quad & Ax = b, \\ & x \geq 0 \end{aligned}$$

where $z \in \mathbb{R}^n$ is the vector of enablers, all the values of the vector are set to zero except for the value of the current objective function which is set to one. The idea is to specify which objective function we are optimizing and activate the relative penalties inherited from the previous objective function. We also add a new set of variables $\epsilon \in \mathbb{R}^{n-1}$ which is used to add the contribution of the penalties and they are paired with the unknowns like the slack variables. The actual penalty is represented by $\bar{f}(x)$ which is defined as:

$$\bar{f}_i(x) = \frac{1}{2}\bar{x_i}^T Q_i x + c_i^T x - f_i^* \tag{4}$$

When one objective function reaches an optimal solution, we can compute this function where $\bar{x_i}$ is the optimal solution finds by that objective function and f_i^* is the value returned by the objective function in that point.

This manipulation of the problem allows the algorithm to take in consideration the gradient of the previously optimized objective functions computed in the last optimal points, forcing the IPM to reach a solution which is closer to that gradients. If you look carefully, this approach is not much different from what we were doing in the previous algorithm where the gradient of the optimized function was used as new constraint of the problem, but now, due to the fact that we are optimizing a single objective function and we know at priori the number of penalties that we are going to have (i.e. the number of ϵ variables), we do not need to change problem after a few iteration achieving a better hardware utilization. Specifically we need to rewrite the matrix Q and the vector c in the following way:

$$c = \begin{bmatrix} c^{(1)} & ... & c^{(n)} & \sum_{i=2}^{n} z_i(-\bar{f}_1^*) & ... & z_n(-\bar{f_{n-1}}^*) \end{bmatrix}, \tag{5}$$

$$Q = \begin{bmatrix} \sum_{i=1}^{n} z_i Q_i & \sum_{i=2}^{n} z_i(\bar{x_1}^T Q_1 + c_1^T)^T & ... & z_n(\bar{x_n}^T Q_n + c_n^T)^T \\ \sum_{i=2}^{n} z_i(\bar{x_1}^T Q_1 + c_1^T) & 0 & ... & 0 \\ \vdots & \vdots & \ddots & \vdots \\ z_n(\bar{x_n}^T Q_n + c_n^T) & 0 & ... & 0 \end{bmatrix} \tag{6}$$

Looking at the figures 3 and 4 we can appreciate that the reformulation of the problem keeps the same property of the previous approach. You can find the pseudo code of the updated algorithm in the following pages.

Besides the advantage of a better hardware utilization, this method does not need to recompute the starting point after each objective optimization, however must allow the ϵ variable to reach values lower than zero because if the current solution moves too far from the gradient of the previous optimal solutions, those values can decrease so fast forcing the algorithm to move the solution back closer.

In the next section we will compare the results of these two new approaches with the same benchmarks used to test the NA-IPM.

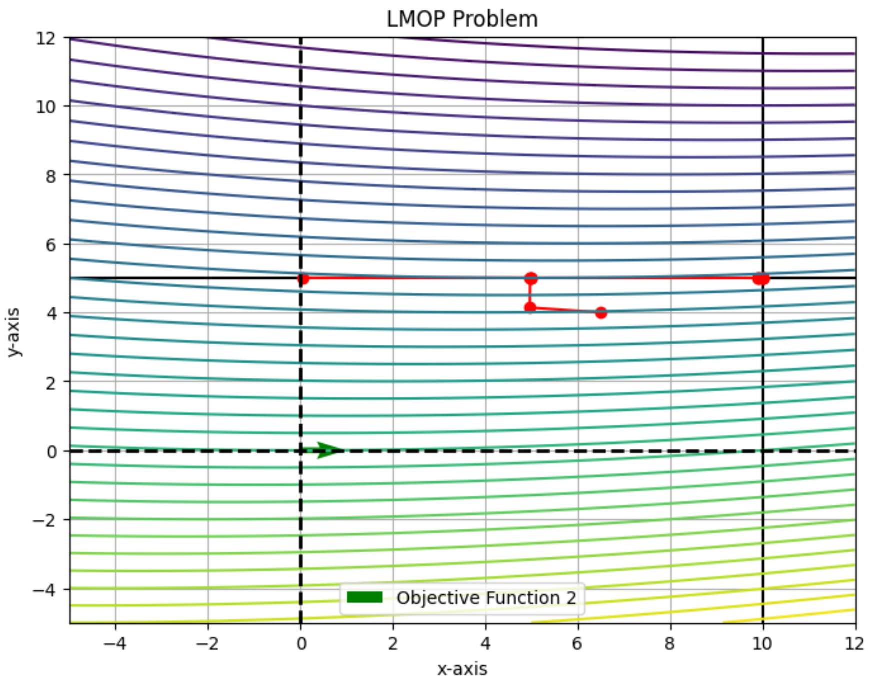


Figure 3: Example of the penalty approach when the first quadratic objective lands on a face of the polyhedron.

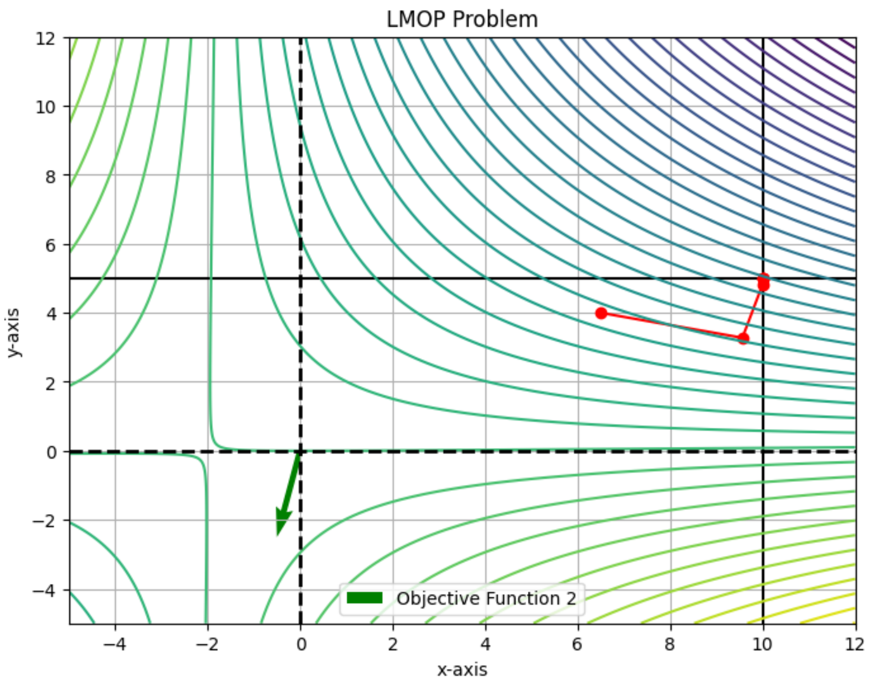


Figure 4: Example of the penalty approach when the first quadratic objective lands on a vertex of the polyhedron. We can see that the second objective function is not able to move the solution away from the vertex.

Algorithm 3: Infeasible Primal-Dual IPM with Penalties

```
   Data: $A, b, objectives, \epsilon, \omega$, max_it
   Result: $x, \lambda, s$, flag
1  flag $\leftarrow$ False
   /* initialize the necessary data structure */
2  $z = [0] * objectives.length$
3  $z[0] \leftarrow 1$
4  $optimal\_points \leftarrow []$
5  $optimal\_solutions \leftarrow []$
6  $Q \leftarrow$ update_Q $(objectives, z, optimal\_points)$
7  $c \leftarrow$ update_c $(objectives, z, optimal\_solutions)$
8  $x, \lambda, s \leftarrow$ starting_point$(A, b, c, Q)$
9  for $k \leftarrow 1$ to $max\_it$ do
       /* Compute residuals */
10     $r_b \leftarrow Ax - b$, $r_c \leftarrow A^T\lambda + s - Qx - c$, $r_\mu \leftarrow x \cdot s$
11     $\mu \leftarrow \frac{r_\mu^T 1}{n}$
       /* KKT conditions satisfaction parameters */
12     $\rho_1 \leftarrow \frac{\|r_b\|}{1+\|b\|}$, $\rho_2 \leftarrow \frac{\|r_c\|}{1+\|c\|}$, $\rho_3 \leftarrow \frac{\mu}{1+\left|\frac{1}{2}x^TQx+c^Tx\right|}$
13     if $\rho_1 < \epsilon$ and $\rho_2 < \epsilon$ and $\rho_3 < \epsilon$ then
14         if $z[-1] == 1$ then
               /* All the objectives have been optimized */
15             flag $\leftarrow$ True
16             return $x, \lambda, s, flag$
           /* Update the data structure with the new information */
17         $z \leftarrow$ update_z ()
18         append$(optimal\_points, x)$
19         append$(optimal\_solutions, \frac{1}{2}x^TQx + c^Tx)$
20         $Q \leftarrow$ update_Q $(objectives, z, optimal\_points)$
21         $c \leftarrow$ update_c $(objectives, z, optimal\_solutions)$
22         continue
       /* Compute predictor direction */
23     $\Delta x_p, \Delta\lambda_p, \Delta s_p \leftarrow$ predict$(A, b, c, Q, r_b, r_c, r_\mu)$
24     $\nu_{pp} \leftarrow 0.99 \min\left(\max_{\bar{\nu}}\{\bar{\nu} \mid x + \bar{\nu}\Delta x_p \geq 0\}, 1\right)$
25     $\nu_{pd} \leftarrow 0.99 \min\left(\max_{\bar{\nu}}\{\bar{\nu} \mid s + \bar{\nu}\Delta s_p \geq 0\}, 1\right)$
26     $\nu \leftarrow \min(\nu_{pp}, \nu_{pd})$
27     $\tilde{x} \leftarrow x + \nu\Delta x_p$, $\tilde{s} \leftarrow s + \nu\Delta s_p$
28     $\mu^{new} \leftarrow \frac{\tilde{x}^T\tilde{s}}{n}$, $\sigma \leftarrow (\frac{\mu^{new}}{\mu})^3$
       /* Compute corrector direction */
29     $\Delta x_c, \Delta\lambda_c, \Delta s_c \leftarrow$ corrector$(A, b, c, Q, \sigma\mu 1 - \Delta x_p \cdot \Delta s_p)$
30     $\Delta x \leftarrow \Delta x_p + \Delta x_c$, $\Delta\lambda \leftarrow \Delta\lambda_p + \Delta\lambda_c$, $\Delta s \leftarrow \Delta s_p + \Delta s_c$
31     $\nu_p \leftarrow 0.99 \min\left(\max_{\bar{\nu}}\{\bar{\nu} \mid x + \bar{\nu}\Delta x \geq 0\}, 1\right)$
32     $\nu_d \leftarrow 0.99 \min\left(\max_{\bar{\nu}}\{\bar{\nu} \mid s + \bar{\nu}\Delta s \geq 0\}, 1\right)$
33     $\nu \leftarrow \min(\nu_p, \nu_d)$
       /* Compute target primal-dual solution */
34     $x \leftarrow x + \mu\Delta x$, $\lambda \leftarrow \lambda + \mu\Delta\lambda$, $s \leftarrow s + \mu\Delta s$
35     if $\|(x, s)\| > \omega$ then
           /* Divergence detected */
36         return $x, \lambda, s, flag$
37 return $x, \lambda, s, flag$
```

4 Results

The purpose of this section is to compare the results achieved by the solutions proposed in this work and, we are going to use as quality measure the number of iterations required to achieve the optimal value. Other quality measures may be used, such as the hardware utilization and the speed of the algorithm, but we decided to ignore those metrics because we are going to use as baseline of our comparison the results of the NA-IPM, which use BANs on standard hardware without leveraging its optimization, therefore, that kind of comparison it would have been unfair.

4.1 Kite problem

First, we test a classic LMOLP called "kite".

$$\begin{aligned}
\min_{x} \quad & -8x_1 - 12x_2, -14x_1 - 10x_2 \\
\text{s.t.} \quad & 2x_1 + x_2 \leq 120, \\
& 2x_1 + 3x_2 \leq 210, \\
& 4x_1 + 3x_2 \leq 270, \\
& x_1 + 2x_2 \geq 60
\end{aligned}$$

Algorithm	**Number of Iterations**	**Optimal Solution (x^*)**
NA-IPM	11	[30.00, 50.00]
Preemptive with tangent	8	[30.00, 50.00]
Penalty approach	11	[30.00, 50.00]

4.2 Pyramid and cylinder

This test has two objective functions, the first one is quadratic and the second is linear. The feasible region is a pyramid.

$$\begin{aligned}
\min_{x} \quad & \frac{1}{2}x^T Q_1 x + c_1^T x, c_2^T x \\
\text{s.t.} \quad & -x_1 + x_2 + x_3 \leq 1, \\
& -x_1 + -x_2 + x_3 \leq 1, \\
& x_1 + -x_2 + x_3 \leq 1, \\
& x_1 + x_2 + x_3 \leq 3, \\
& x_3 \geq 0
\end{aligned}$$

where:

$$c_1 = \begin{bmatrix} -16 & -16 & -16 \end{bmatrix}, Q_1 = \begin{bmatrix} 10 & -2 & 4 \\ -2 & 10 & 4 \\ 4 & 4 & 4 \end{bmatrix} c_2 = \begin{bmatrix} -1 & -1 & 0 \end{bmatrix} \qquad (7)$$

Algorithm	Number of Iterations	Optimal Solution (x^*)
NA-IPM	11	[1.50, 1.50, 0]
Preemptive with tangent	9	[1.50, 1.50, 0]
Penalty approach	5	[1.49 1.49 0.008]

4.3 Pyramid and two paraboloid

This test has three objective functions, the first one is linear and the next two are quadratic. The feasible region is a pyramid.

$$\begin{aligned} \min_{x} \quad & c_1^T x, \frac{1}{2}x^T Q_2 x + c_2^T x, \frac{1}{2}x^T Q_3 x + c_3^T x \\ \text{s.t.} \quad & -x_1 + x_2 + x_3 \leq 1, \\ & -x_1 + -x_2 + x_3 \leq 1, \\ & x_1 + -x_2 + x_3 \leq 1, \\ & x_1 + x_2 + x_3 \leq 3, \\ & x_3 \geq 0 \end{aligned}$$

where:

$$c_1 = \begin{bmatrix} -1 & -1 & -1 \end{bmatrix}, \tag{8}$$

$$c_2 = \begin{bmatrix} -5 & -5 & 0 \end{bmatrix}, Q_2 = \begin{bmatrix} 2 & 2 & 0 \\ 2 & 2 & 0 \\ 0 & 0 & 4 \end{bmatrix} \tag{9}$$

$$c_3 = \begin{bmatrix} -5 & -3 & 2 \end{bmatrix}, Q_3 = \begin{bmatrix} 4 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{10}$$

Algorithm	Number of Iterations	Optimal Solution (x^*)
NA-IPM	16	[1.67, 1.17, 0.17]
Preemptive with tangent	18	[1.67, 1.17, 0.17]
Penalty approach	52	[1.67, 1.17, 0.17]

4.4 Conclusions

From the previously presented results, we can see that all the algorithms reach the same final solution. The number of iterations is also quite close except for the penalty approach which spent a lot of iteration to reach the optimum of the third problem. Looking at the intermediate iterations we can observe that the algorithm suffers from numerical instabilities as the number of penalties increases, so it needs further investigation to check if it is possible to mitigate the problem.

This comparison may be also slightly bias by the technical implementation of these algorithms, indeed the author of NA-IPM used Julia as a programming language, which is more suited for those kind of application, instead the algorithms of this work have been developed in Python with NumPy and SciPy packages.

The results and the actual implementation of this work can be checked at the following link: GitHub.

References

[1] Lorenzo Fiaschi and Marco Cococcioni. A non-archimedean interior point method and its application to the lexicographic multi-objective quadratic programming. *Mathematics*, 10(23):4536, November 2022.

[2] Soyster Sherali, H.D. Preemptive and nonpreemptive multi-objective programming: Relationship and counterexamples. (23), February 1983.

[3] Marco Cococcioni Vieri Benci. The algorithmic numbers in non-archimedean numerical computing environments. discrete and continuous dynamical systems. (23), September 2021.