A Decision Space Algorithm for Multiobjective Convex Quadratic Integer Optimization

Marianna De Santis[a], Gabriele Eichfelder[b]

[a]*Dipartimento di Ingegneria Informatica Automatica e Gestionale, Sapienza Università di Roma, Via Ariosto, 25, 00185 Roma, Italy, (*marianna.desantis@uniroma1.it*)*
[b]*Institute for Mathematics, Technische Universität Ilmenau, Po 10 05 65, D-98684 Ilmenau, Germany (*gabriele.eichfelder@tu-ilmenau.de*)*

Abstract

We present a branch-and-bound algorithm for minimizing multiple convex quadratic objective functions over integer variables. Our method looks for efficient points by fixing subsets of variables to integer values and by using lower bounds in the form of hyperplanes in the image space derived from the continuous relaxations of the restricted objective functions. We show that the algorithm stops after finitely many fixings of variables with detecting both the full efficient and the nondominated set of multiobjective strictly convex quadratic integer problems. A major advantage of the approach is that the expensive calculations are done in a preprocessing phase so that the nodes in the branch-and-bound tree can be enumerated fast. We show numerical experiments on biobjective instances and on instances with three and four objectives.

Keywords: Multiobjective Optimization, Convex Quadratic Optimization, Integer Quadratic Programming, Branch-and-Bound Algorithm
2000 MSC: 90C10, 90C25, 90C29, 90C57

1. Introduction

We study the minimization of m strictly convex quadratic objective functions over integer variables. It is known, see [7, 8], that already minimizing one of such functions w.r.t. integer variables is NP-hard. In this paper, we devise a method able to exactly detect both the efficient and the nondominated set of multiobjective strictly convex quadratic integer problems.

Focusing on this specific class of problems allows us to set some theoretical basis for the definition of algorithms for multiobjective quadratic integer problems. While multiobjective linear integer optimization has been deeply studied in the last two decades, literature on multiobjective nonlinear integer optimization is still very limited. Addressing convex quadratic objective functions can be seen as the next step after studying linear objective functions, thus, in the multiobjective integer context, the next step to address nonlinear problems. Therefore, this paper gives

a contribution to the study of multiobjective nonlinear integer optimization and to the definition of algorithms able to address this class of problems. The theoretical results provided in this paper can be used for further developments in the context of multiobjective quadratic integer optimization, as non-convex quadratic or linearly constrained problems. In particular, they can be combined with the approaches proposed in [4, 5, 6] for computing lower bounds of single-objective quadratic integer minimization problems.

There is a multitude of applications that are modeled as convex quadratic integer programs. We refer here for instance to [27, p. 61] and a (single-objective) application in electronics known as filtered approximation. For a short description of this problem see [3]. The closest vector problem [29] is another such example, in the literature also known as the Integer Least Squares Problem, which has again several applications, cf. [27, p. 62], and as soon as a lattice point is searched with closest distance to several targets one ends up with a multiobjective formulation as treated in this paper. Quadratic multidimensional knapsack problems are another example for convex quadratic integer programs, see [23], and as soon as multiple gain vectors exist this also results in a multiobjective quadratic integer problem.

For multiobjective linear integer optimization, enormous progress can be seen. The algorithms devised for this class of problems often belong to one of the following main groups: decision space search algorithms, i.e., approaches that work in the space of feasible points, and criterion space search algorithms, i.e., methods that work in the space of objective function values. For a recent paper on this topic which includes a good literature review, we mention [17]. Our algorithm will be based on the decision space where the branching will be done by fixing subsets of variables. This will guarantee to find all efficient solutions for the multiobjective optimization problem.

In the literature, decision space based multiobjective integer optimization algorithms are further differentiated in branch-and-bound and branch-and-cut algorithms. Branch-and-bound is a well-known method in various fields of optimization. In the context of integer optimization branch-and-bound methods are based on a potentially full enumeration of all feasible integer solutions, which can be viewed as a tree search, combined with criteria based on bounds which allow to cut some of the branches. Good bounding criteria allow for an efficient algorithm. A survey on branch-and-bound methods for multiobjective linear integer problems is given in [22]. The proposed algorithm is a branch-and-bound method that uses the ideal point for bounding. We also examine more general lower bound sets where cuts in the criterion space allow for tighter lower bound sets.

Branch-and-cut algorithms work with relaxations of the feasible set. Typically, the integrality assumption is relaxed, as we also do it for obtaining our lower bounds in the nodes of the branch-and-bound tree. As these relaxations are typically not tight, one can improve them by adding cuts in the decision space, as done for biobjective linear binary optimization problems in [14]. We do not follow this approach here, as we are exploring unconstrained problems only.

With this paper we want to contribute to further developments in multiobjective integer optimization by now studying convex quadratic instead of linear objective functions, i.e., we approach multiobjective nonlinear integer optimization. For multiobjective nonlinear integer optimization, only a few algorithms have been proposed so far. Most of them are for binary problems only, see [20, 25, 30] or deal with only two objective functions. Blanco and Puerto examine in [2] multiobjective polynomial integer optimization problems and present an algebraic approach for solving these problems. They first rewrite the problem as a binary problem by introducing additional variables and constraints. In this way, they are able to deal with the optimization problem by solving one or more systems of polynomial equations. Tests on multiobjective knapsack and biobjective portfolio selection problems as well as on random instances have been performed.

Another paper for biobjective integer, but not necessarily linear, problems is given by Dogan, Karsu and Ulus [11]. However, they assume integer-valued objective functions. The algorithm considers boxes in the image space which are iteratively refined by solving single-objective integer subproblems, thus the algorithm is a criterion space approach. A somehow related idea was proposed by De Santis, Grani and Palagi [10]. That method can handle nonlinear problems too, as long as some assumptions on the values of the objective functions are satisfied. Both algorithms make use of the clear structure of a two-dimensional image-space while our algorithm can also be applied to more than two objective functions. A disadvantage of algorithms that work in the criterion space is that they do not easily scale with the number of objective functions. Attempts to overcome this issue have recently been made (see, e.g., [16, 26, 28]).

A key tool used in the solution approaches mentioned above is scalarization. This means the replacement of the multiobjective problem by a parameter-dependent single-objective optimization problem. The single-objective subproblems are integer optimization problems and have to be solved by appropriate solvers. In order to find several nondominated solutions, this has to be done repeatedly considering different values of the parameters. Typical issues of these methods are the choice of the parameters for problems with more than two or three objective functions, as well as to steer the parameters in such a way that it can be guaranteed that all efficient points can be found. Moreover, the choice of the scalarization is important. Without a specific strategy to remove dominated parts of the feasible region, the weighted-sum method will fail to find all efficient points of a multiobjective integer optimization problem as the problem is non-convex due to the integrality constraint. Other scalarizations, such as the ε-constraint method, produce single-objective subproblems adding further constraints to the original feasible set. Our approach does not make use of scalarization of the original problem.

We finally mention solvers developed for multiobjective convex mixed integer optimization problems, see for instance [9] and the references therein. These solvers can clearly also be applied to deal with multiobjective convex quadratic integer programs. In our numerical experiments, we compare our method with the solver

MOMIX devised in [9]. For that reason, we shortly recall its basic ingredients. MOMIX is a decision space branch-and-bound method. In case one applies it to purely integer problems, the branching corresponds to adding integer lower and upper bounds on the values of the variables. Hence, the branching rule is not based on the fixing of the integer variables as we will do it here. The bounding is done by comparing lower bounds with upper bounds, which are obtained as the images of feasible points.

In MOMIX, lower bounds are first calculated as ideal points of the relaxed problems, where relaxed means that the integrality constraints are ignored. The evaluation whether the lower bounds are 'above' the upper bounds is then performed by solving single-objective convex continuous optimization problems. In case this test fails, refined lower bounds are built by solving single-objective convex (mixed integer) subproblems. Ideal points are considered as lower bounds in the algorithm proposed here as well. Possible improvements and alternative lower bounds are discussed in Section 3.2. As fast enumeration of the nodes is essential in the branch-and-bound method we devise, the alternative lower bounds are defined in order to exploit a preprocessing phase as much as possible.

In contrast to the method proposed here, the method devised in [9] requires a starting box in which the feasible set is contained. For this reason, in order to apply MOMIX to unconstrained instances, we have to artificially set lower and upper bounds for the integer variables. Moreover, MOMIX does not exploit the structure of the quadratic objectives. So neither convexity is used for pruning branches nor any preprocessing speeds up the iteration steps.

The preprocessing phase which we use to speed up the calculations along the iterations follows ideas developed by Buchheim, Caprara and Lodi in [3], where single-objective unconstrained convex quadratic integer optimization problems are addressed. As in [3], we branch in the pre-image space by fixing subsets of integer variables and we do a preprocessing which allows fast computations within the nodes.

The key to deal with more than one objective function is in the definition of a pruning criterion which guarantees the finiteness of our algorithm. In other words, we are able to guarantee that the enumeration of the nodes is finite despite the problem is unconstrained. While the optimal value of a single-objective problem is unique (if it exists) it is in general non-unique in the case of multiple objectives. A multiobjective problem has many optimal solutions in general, called efficient solutions. Hence, multiobjective problems are often numerically more challenging compared to single-objective ones, as branch-and-bound methods need to explore a higher number of nodes. The fact that we look for a set of optimal solutions influences also the theoretical analysis of the algorithms and in particular the proofs of their finiteness.

Another difference to the single-objective case is that the continuous relaxations of the problem, namely the subproblems obtained by relaxing the integrality constraint, do not have a unique minimizer. In fact, each of the continuous relaxations is a multiobjective continuous problem having its own set of efficient solutions. We handle this issue by considering the ideal point of the multiobjective continuous re-

laxation, that is a lower bound. As detailed in the paper, the use of a preprocessing phase allows its fast computation along the nodes. We also suggest alternative lower bounds to the ideal point as already mentioned above. The non-uniqueness of the optimal solutions of the subproblems also influences the bounds for the fixing of the variables, which are now determined by intervals that are derived from the unique minimizers of all functions.

The paper is structured as follows. In Section 2 we state the multiobjective convex quadratic integer optimization problem under examination as well as some basic definitions and observations. Section 3 contains the new algorithm together with the underlying theoretical results about the correctness of the pruning steps and the finiteness of the enumeration (Section 3.1). As already mentioned, in Section 3.2 we present and discuss several alternatives for the computation of the lower bounds which are used within the algorithm. In Section 3.3 we comment on the inclusion of bounds in the optimization problem. We illustrate the algorithm on a toy example in Section 4, before we present the numerical results in Section 5. We compare the new algorithm with the multiobjective mixed-integer solver `MOMIX` [9] and with a scalarization approach, and test it on scalable and random instances.

2. Multiobjective Quadratic Integer Optimization

For our multiobjective optimization problem, we assume the objective functions to be given by $f_j \colon \mathbb{R}^n \to \mathbb{R}$,

$$f_j(x) = x^T Q_j x + {c^j}^T x + a_j,$$

$j = 1, \dots, m$, $m \geq 2$, with symmetric positive definite matrices $Q_1, \dots, Q_m \in \mathcal{S}^n$, vectors $c^1, \dots, c^m \in \mathbb{R}^n$, and scalars $a_1, \dots, a_m \in \mathbb{R}$. The multiobjective quadratic integer programming problem which we study in this paper is then

$$\begin{array}{ll} \min & (f_1(x), \dots, f_m(x))^T \\ \text{s.t.} & x \in \mathbb{Z}^n. \end{array} \tag{MOQIP}$$

Even for small dimensions of n a full enumeration of all integer settings is not immediately possible, as there are no bounds for the integer variables. We discuss the possible inclusion of bounds in Section 3.3. Note that problem (MOQIP) is nonconvex because of the presence of integrality constraints. However, when some of the integer variables are fixed and the integrality constraint for the remaining is relaxed, it reduces to a continuous convex quadratic multiobjective problem as the objective functions are convex quadratic. Continuous convex quadratic functions are symmetric with respect to the minimizers, see the following remark. This helps us to prove later that the branch-and-bound tree is finite.

Remark 2.1. *Let $j \in \{1, \dots, m\}$. It holds for $\bar{x} = -\frac{1}{2} Q_j^{-1} c^j$ the minimal solution of $\min_{x \in \mathbb{R}^n} f_j(x)$ that for any $x \in \mathbb{R}^n$*

$$f_j(x) = f_j(\bar{x}) + (x - \bar{x})^T Q_j (x - \bar{x}).$$

While for $n \in \mathbb{N}$, $m = 1$, solving (MOQP) is easy and the optimal solution is $\bar{x} = -\frac{1}{2}Q_1^{-1}c^1$ with objective value $a_1 - \frac{1}{4}(c^1)^T Q_1^{-1} c^1$, problem (MOQIP) is already NP-hard.

The image of the feasible set of the problem, here $\mathbb{Z}^n$, under the vector-valued function $f : \mathbb{R}^n \to \mathbb{R}^m$ represents the feasible set in the criterion space, or the image set. In general, there does not exist a feasible point $\bar{x}$ which minimizes all objective functions at the same time. This means that the objective functions are competing. We also assume this for our examinations. A point $\bar{x} \in \mathbb{Z}^n$ is called an *efficient* point of (MOQIP) if there is no $x \in \mathbb{Z}^n$ with $f(x) \neq f(\bar{x})$ and $f(x) \leq f(\bar{x})$. Here, $\leq$ is understood componentwise. The image $f(\bar{x})$ of an efficient point of (MOQIP) is called *nondominated*. There also exists the notion of weak efficiency. A point $\bar{x} \in \mathbb{Z}^n$ is called a *weakly efficient* point of (MOQIP) if there is no $x \in \mathbb{Z}^n$ with $f(x) < f(\bar{x})$. Weakly efficient solutions are in general not of interest from a practical point of view. We aim at detecting the efficient and/or the nondominated set.

Note that for the relaxed problem

$$\begin{array}{ll} \min & (f_1(x), \ldots, f_m(x))^T \\ \text{s.t.} & x \in \mathbb{R}^n \end{array} \tag{MOQP}$$

there are no two efficient points $x, \bar{x} \in \mathbb{R}^n$, $x \neq \bar{x}$, mapping on the same nondominated point $z \in f(\mathbb{R}^n)$, as the objective functions are strictly convex. This can easily be seen by considering the point $w := (x + \bar{x})/2$ for some $x \neq \bar{x}$, as we obtain in case of $f(x) = f(\bar{x})$ for $j = 1, \ldots, m$ that $f_j(w) < \frac{1}{2} f_j(x) + \frac{1}{2} f_j(\bar{x}) = f_j(\bar{x})$. Thus, to any nondominated point $z \in f(\mathbb{R}^n)$ there exists a unique $\bar{x} \in \mathbb{R}^n$ with $z = f(\bar{x})$. While this holds for (MOQP), it does not hold for (MOQIP) in general:

Example 2.2. *For $n = 1$, $m = 2$ and the functions $f_1(x) = (x - 1.5)^2$ and $f_2(x) = (x - 1.5)^2 + 3$ we have as set of efficient points $\{1, 2\}$ for* (MOQIP)*, while the set of nondominated points is a singleton and equals $\{(0.25, 3.25)^T\}$. For another example with concurrent objective functions see the toy example in Section 4.*

We aim on finding all efficient solutions for (MOQIP), and based on the above, the cardinality of the set of efficient solutions can be larger than the cardinality of the set of nondominated points for (MOQIP).

Some approaches in multiobjective optimization, as for instance some scalarizations as the ε-constraint method, only guarantee to find weakly efficient points. Thus it is important to know whether there are weakly efficient points which are not also efficient. For (MOQP) it is known that no weakly efficient point exists which is not also efficient for (MOQP). For strictly convex objective functions we have for any points $x, \bar{x} \in \mathbb{R}^n$, $x \neq \bar{x}$, with $f(x) \leq f(\bar{x})$ and w as above $f(w) < f(\bar{x})$. Thus, if a point is not efficient, it is also not weakly efficient. However, this is not true for (MOQIP). In case of integrality constraints, there can be weakly efficient points which are not also efficient, as the following simple example shows:

Example 2.3. *Let* $n = 1$, $m = 2$ *and consider problem* (MOQIP) *with* $f_1(x) = (x-1.5)^2$ *and* $f_2(x) = (x-1)^2$. *Then* $f(1) = (0.25, 0)^T$ *and* $f(2) = (0.25, 1)^T$. *Here,* $x = 1$ *and* $x = 2$ *are weakly efficient but only* $x = 1$ *is efficient.*

This has to be taken into account in case one aims to solve (MOQIP) with an approach based on scalarizations. With our algorithm, we determine the set of efficient points of problem (MOQIP) and no point being weakly efficient only.

Finally, we would like to mention that not all efficient points for problem (MOQIP) can be found by the weighted-sum method which uses as scalarization for parameters $w \in \mathbb{R}^m_+$, $w \neq 0$, problems of the form $\min_{x\in\mathbb{Z}^n} w^T f(x)$. Not all nondominated points are located on the convex hull of $f(\mathbb{Z}^n)$. This can be seen for example in instance (Inst1) on page 26 and Figure 4 on page 26, and the weighted-sum approach will fail to find those.

In the following, for any $x \in \mathbb{R}$, we denote by $\lfloor x \rfloor$ and by $\lceil x \rceil$ the next integer above or below x. We adopt the same notation for vectors $x \in \mathbb{R}^n$ by applying it componentwise.

3. `BB-MOQIP`: a Branch-and-Bound method on the decision space

We propose a branch-and-bound method able to exactly detect the efficient and the nondominated set of Problem (MOQIP). Thereby we branch in the decision space where we built a rooted tree which basically allows to perform a systematic enumeration of all candidate solutions. The set $\mathbb{R}^n$ is chosen at the root and then we explore the branches by recursively assigning values for the integer variables. To avoid a full enumeration, which would in fact be infinite in the unconstrained case, and to make the procedure efficient, we include bounding strategies.

This means that before enumerating the further branches, a lower bound for all remaining possible values in the image space is calculated. As in our multiobjective setting the values form a subset of $\mathbb{R}^m$, the lower bound has to take this into account and has to be a lower bound for all values in this set. This lower bound can either be a vector in $\mathbb{R}^m$ or a subset of $\mathbb{R}^m$ and is then compared to upper bounds for the nondominated set which have been generated so far. Upper bounds are generated from images of feasible points. Thus, each integer assignment gives an upper bound by evaluating its image under f. In single-objective optimization, there is one best, i.e., smallest, upper bound which is greater than the optimal value. In the multiobjective setting there can be a list of upper bounds, all being non-comparable, and there can be nondominated points which are not componentwise smaller than one of these upper bounds. Still, the upper bounds determine regions in the image space which can no longer contain nondominated points. Within the node the lower bound is checked against that list of upper bounds and allows in some cases the discarding of further nodes.

Our method adopts a depth-first enumeration strategy, based on fixing variables to integer values, as the one used in [3] for solving single-objective convex quadratic integer programs. The order in which the variables are fixed is predetermined,

meaning that the branch-and-bound tree is formed by exactly n levels. Assume w.l.o.g. that we fix the variables from 1 to n. With $d = 0$ we denote the root node. At every node of level $d \in \{1, \dots, n\}$ the variables $x_1, \dots, x_d$ are fixed to certain integer values, say, $r_1, \dots, r_d$. Let $r \in \mathbb{Z}^d$ be the vector $(r_1, \dots, r_d)^T$. From now on we will use the following notation. For every $j = 1, \dots, m$, we define $f_j^r : \mathbb{R}^{n-d} \to \mathbb{R}$ by

$$f_j^r(x) := x^T Q_j^d x + (\bar{c}^{j,r})^T x + \bar{a}_{j,r},$$

where the positive definite symmetric matrix Q_j^d is obtained by deleting the corresponding d rows and columns of Q_j and $\bar{c}^{j,r}$ and $\bar{a}_{j,r}$ are set to

$$\bar{c}_{i-d}^{j,r} := c_i^j + 2\sum_{k=1}^{d} q_{ki} r_k, \text{ for } i = d+1, \dots, n$$

and

$$\bar{a}_j := a_j + \sum_{k=1}^{d} c_k r_k + \sum_{k=1}^{d}\sum_{i=1}^{d} q_{ki} r_k r_i.$$

Simple calculations using the symmetry of Q_j yield the following relation:

Lemma 3.1. *For f_j^r, $j \in \{1, \dots, m\}$, as defined above it holds*

$$f_j^r(x) = f_j(r_1, \dots, r_d, x_1, \dots, x_{n-d}).$$

By $x^{*j,r} \in \mathbb{R}^{n-d}$, $j \in \{1, \dots, m\}$, we denote the unconstrained minimizers of f_j^r, uniquely defined as

$$x^{*j,r} = -\frac{1}{2}(Q_j^d)^{-1}\bar{c}^{j,r}. \tag{1}$$

Assume that we are in a node of level d and the variables $x_1, \dots, x_d$ are fixed to $r = (r_1, \dots, r_d)^T$. The values at which variable x_{d+1} is fixed in the children nodes are determined as follows. Let

$$\alpha(d+1) := \min_{j=1,\dots,m} x_1^{*j,r}, \qquad \beta(d+1) := \max_{j=1,\dots,m} x_1^{*j,r}\ . \tag{2}$$

We report in Figure 1 an example with three objective functions and one variable, where the values of $\lfloor\alpha\rfloor$ and $\lceil\beta\rceil$ are highlighted.

We start from fixing x_{d+1} to $\lfloor\alpha(d+1)\rfloor$ and then we consecutively fix x_{d+1} to increasing integer values $\lfloor\alpha(d+1)\rfloor+1, \lfloor\alpha(d+1)\rfloor+2, \dots$, until we reach $\lceil\beta(d+1)\rceil$. We go on fixing x_{d+1} to increasing integer values until a specific condition is satisfied. This condition is based on comparing some upper bounds with lower bounds and it ensures that no efficient point with some larger fixings exist. We discuss this condition in Section 3.1. Due to convexity, we can show that at some point this condition is satisfied and we can stop, see Lemma 3.6. Then we go on fixing x_{d+1} to decreasing integer values starting from $\lfloor\alpha(d+1)\rfloor - 1$, again, until the condition is satisfied. The rules adopted to fix the variables are detailed in Algorithm 1.

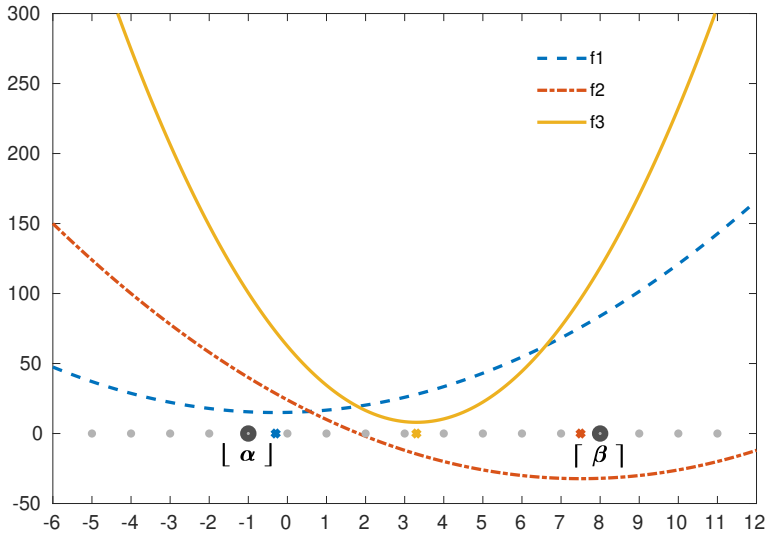

Figure 1: Definition of $\lfloor \alpha \rfloor$ and $\lceil \beta \rceil$ considering three objective functions.

Algorithm 1 Update r_d

INPUT: $r_d, \alpha(d)$
OUTPUT: r_d

1: **if** $r_d \geq \lfloor \alpha(d) \rfloor$ **then**
2: Set $x_d = r_d + 1$;
3: **else**
4: Set $x_d = r_d - 1$;
5: **end if**
6: Set $r_d = x_d$;

We report in Algorithm 3 the scheme of our branch-and-bound `BB-MOQIP`. In Section 4 we have included a toy example on which we also illustrate the behaviour of our algorithm. With $\mathcal{L}_{PNS}$ and $\mathcal{E}$ we denote lists of points in the criterion space and in the decision space respectively, which are updated along the iterations of the algorithm. We will show that when the algorithm stops, these are exactly the nondominated and the efficient set of (MOQIP), respectively. Before entering in the main loop, we perform a preprocessing phase (see Algorithm 2), where the inverse matrices of Q_j^d, $j = 1, \ldots, m$, $d = 0, \ldots, n-1$ are computed. Note that with $d = 0$, we mean the root node where $Q_j^d = Q_j$. The role of r^* is only to initialize the set $\mathcal{E}$ and can thus be chosen arbitrarily, for instance $r^* = 0$. As we have a depth-first enumeration strategy, this list is updated quickly.

Algorithm 2 Preprocessing

INPUT: m strictly convex quadratic functions $f_j : \mathbb{R}^n \to \mathbb{R}$, $j = 1, \dots, m$.
OUTPUT: (Q_j^d), $(Q_j^d)^{-1}$ for $d = 0, \dots, n-1$, for j=1,...,m; $\mathcal{E}$; $\mathcal{L}_{PNS}$.

1: Determine a variable order $x_1, \dots x_n$ and resort Q and c accordingly;
2: For $d = 0, \dots, n-1$ and $j = 1, \dots, m$ let Q_j^d be the submatrix of Q_j given by rows and columns $d+1, \dots, n$;
3: For $d = 0, \dots, n-1$ and $j = 1, \dots, m$ compute $(Q_j^d)^{-1}$;
4: Choose an initial feasible solution $r^* \in \mathbb{Z}^n$;
5: Set $\mathcal{E} \leftarrow \{r^*\}$ and $\mathcal{L}_{PNS} \leftarrow \{(f_1(r^*), \dots, f_m(r^*))\}$.

In Algorithm 3, at every iteration of the loop, namely at each node of the branch-and-bound tree, we first compute $x^{*j,r}$ and a lower bound. For our problem (MOQIP) a lower bound is any set $LB \subseteq \mathbb{R}^m$ such that $LB + \mathbb{R}^m_+$ contains the image of integer feasible points through f, namely $f(\mathbb{Z}^n) \subseteq LB + \mathbb{R}^m_+$. The lower bound LB within an iteration of the loop thus has to satisfy

$$\{f(x) \in \mathbb{R}^m \mid x \in \mathbb{Z}^n,\ (x_1, \dots, x_d)^T = (r_1, \dots, r_d)^T\} \subseteq LB + \mathbb{R}^m_+.$$

In Algorithm 3 we consider as lower bound a singleton set which consists of the ideal point for the problem

$$\min_{x \in \mathbb{R}^{n-d}} (f_1^r(x), \dots, f_m^r(x))^T.$$

As the set is a singleton only, we omit the brackets, and just set

$$LB := (f_1^r(x^{*1,r}), \dots, f_m^r(x^{*m,r}))^T. \tag{3}$$

Note that the preprocessing phase (Algorithm 2) allows us to compute the minimizers $x^{*j,r}$ - and then LB as defined above - along the nodes very efficiently.

In the literature, the ideal point is often used within branch-and-bound algorithms for bounding, for instance in multiobjective nonconvex optimization, see [9, 13, 21, 24]. It can also be interpreted as defining a lower bound of $f^r(\mathbb{R}^{n-d})$, and thus of $f^r(\mathbb{Z}^{n-d})$, with the help of m hyperplanes, each defined by

$$\left\{y \in \mathbb{R}^m \mid (e^j)^T y = f_j^r(x^{*j,r})\right\}, \tag{4}$$

where e^j denotes the jth unit vector. In Section 3.2, we will propose other lower bounds that can be used within our scheme. One of the possibilities will be to add additional hyperplanes to enrich the lower bound set.

The next step in Algorithm 3 is the computation of $\alpha(d+1)$ and $\beta(d+1)$ that will determine the values at which variable x_{d+1} will be fixed, see (2).

Condition (Cond) checks whether the lower bound computed at the node is dominated by any of the upper bounds encountered so far and will be defined in the next subsection. The condition makes use of the so far best found points in the criterion space, i.e., of the list $\mathcal{L}_{PNS}$. As in the first iteration the list $\mathcal{L}_{PNS}$ was just

Algorithm 3 `BB-MOQIP`: a branch-and-bound method for MOQIP

INPUT: m strictly convex quadratic functions $f_j : \mathbb{R}^n \to \mathbb{R}$, $j = 1, \ldots, m$

OUTPUT: $\mathcal{E}$, $\mathcal{L}_{PNS}$

Perform Algorithm 2
Set $d := 0$;
while $d \geq 0$ **do**
    if $d \leq n-1$ **then**
        Compute $x^{*j,r} \in \mathbb{R}^{n-d}$ and set $LB := (f_1^r(x^{*1,r}), \ldots, f_m^r(x^{*m,r}))^T$
        Compute $\alpha(d+1)$ and $\beta(d+1)$ according to (2)
    else
        Set (Cond)=false
    end if
    if (Cond) and $(r_d < \lfloor \alpha(d) \rfloor)$ **then**
        Set $d = d - 1$;
        if $(d > 0)$ Update r_d with Algorithm 1 **else** STOP **endif**
    else
        if (Cond) and $r_d \geq \lfloor \alpha(d) \rfloor$ **then**
            if $r_d \geq \lceil \beta(d) \rceil$ **then**
                Set $x_d = \lfloor \alpha(d) \rfloor - 1$, $r_d = x_d$;
            else
                Set $x_d = r_d + 1$, $r_d = x_d$;
            end if
        else
            if $d = n$ **then**
                Update $\mathcal{E}$ by r and $\mathcal{L}_{PNS}$ by $(f_1(r), \ldots, f_m(r))$
                if $r_d \in [\lfloor \alpha(d) \rfloor, \lceil \beta(d) \rceil - 1]$ **then**
                    Set $x_d = r_d + 1$, $r_d = x_d$;
                else
                    Set $d = d - 1$.
                    if $(d > 0)$ Update r_d with Algorithm 1 **else** STOP **endif**
                end if
            else
                Set $d = d + 1$;
                Set $r_d = \lfloor \alpha(d) \rfloor$, $x_d = r_d$;
            end if
        end if
    end if
end while

initialized with one point, and as the objective functions are competing, we never have that (Cond) is satisfied for $d = 0$. Thus, for $d = 0$, we never have to evaluate $r_d < \lfloor \alpha(d) \rfloor$ in line 10 or $r_d \geq \lfloor \alpha(d) \rfloor$ in line 14 and we never reach $d = d - 1$ in line 11. Instead, for $d = 0$, we go directly to line 29.

In the further iterations, (Cond) allows us to check whether the current node might be pruned: thanks to the convexity of the objective functions, in case we are at a node where variable x_d is fixed to values outside the interval $[\lfloor \alpha(d) \rfloor , \lceil \beta(d) \rceil]$, we can prune the node and all its outer siblings as soon as condition (Cond) is satisfied. Otherwise, we need to go on enumerating the nodes and explore level d by fixing x_d to further integer values. See Section 3.1 for further details.

Note that our scheme adopts a depth-first enumeration. Along the iterations, we go on increasing d, or, in other words, we go deep in the branching tree, until we reach a leaf where $d = n$ and all the variables are fixed to integer values $r = (r_1, \ldots, r_n)$. We then get an additional upper bound by evaluating the objective functions at this integer point $r \in \mathbb{Z}^n$.

In our algorithm the list of *potentially nondominated solutions* $\mathcal{L}_{PNS}$ is initialized by $f(r^*)$, with $r^* \in \mathbb{Z}^n$ chosen in the preprocessing phase (Algorithm 2). Then, every time an upper bound $z \in f(\mathbb{Z}^n)$ is computed (this happens when all the variables are fixed to integer values), we check whether it is dominated by any point in $\mathcal{L}_{PNS}$. If this is the case, z is not added to $\mathcal{L}_{PNS}$. Otherwise, we update the list by adding z to $\mathcal{L}_{PNS}$ and by removing from $\mathcal{L}_{PNS}$ all the upper bounds dominated by z. The same is done in the pre-image space for the set $\mathcal{E}$.

When our branch-and-bound algorithm stops, $\mathcal{E}$ and $\mathcal{L}_{PNS}$ are made exactly of the efficient and the nondominated points of Problem (MOQIP), respectively. We prove this in Theorem 3.7.

3.1. Pruning of the nodes

In this section, we go into details of our pruning strategy. On the one hand, we prune nodes, i.e., we discard further branches of the branch-and-bound tree, by using the general idea of a multiobjective branch-and-bound method: we compare the current lower bound with the generated upper bounds. On the other hand, by using convexity arguments, we also give criteria which allow to prune further nodes on the same level of the branching tree, i.e., some of the siblings of the current node. This also guarantees that the branching tree is finite despite the problem is unconstrained.

Assume that we are at a node of level d where x_d is fixed to an integer value outside the interval $[\lfloor \alpha(d) \rfloor , \lceil \beta(d) \rceil]$. We will prove that if the lower bound is dominated by some upper bound in the $\mathcal{L}_{PNS}$ list, we can prune the node (Lemma 3.3) as well as all its outer siblings (Lemma 3.4 and Lemma 3.5).

As a first result, we prove that the last component of an efficient point of (MOQIP) must lie in the interval $[\lfloor \alpha(n) \rfloor, \lceil \beta(n) \rceil]$. Thus, in any final node the number of necessary fixings is finite and clearly defined, even while (MOQIP) itself is unconstrained.

Lemma 3.2. *Let $\bar{x} \in \mathbb{Z}^n$ be any efficient point of the multiobjective optimization problem* (MOQIP). *Then, $\bar{x}_n \in [\lfloor \alpha(n) \rfloor, \lceil \beta(n) \rceil]$ (where the interval is determined with respect to the fixing $r = (\bar{x}_1, \dots, \bar{x}_{n-1})^T$).*

Proof. Let $\bar{x} \in \mathbb{Z}^n$ be any efficient point of the multiobjective optimization problem (MOQIP). Let $d = n - 1$ and $r = (\bar{x}_1, \dots, \bar{x}_{n-1})^T$. It holds for all $j \in \{1, \dots, m\}$ that $f_j^r \colon \mathbb{R} \to \mathbb{R}$ is strictly monotonically decreasing on $(-\infty, x^{*j,r})$ and strictly monotonically increasing on $(x^{*j,r}, \infty)$. Thus all functions f_j^r are strictly monotonically decreasing on $(-\infty, \lfloor \alpha(n) \rfloor]$. Hence, we have that $\bar{x}_n \geq \lfloor \alpha(n) \rfloor$, as otherwise $f(r, \lfloor \alpha(n) \rfloor)$ would dominate $f(r, \bar{x}_n) = f(\bar{x})$ in contradiction to $\bar{x}$ being efficient. Analogously, all functions f_j^r are strictly monotonically increasing on $[\lceil \beta(n) \rceil, \infty)$ and thus we have that $\bar{x}_n \leq \lceil \beta(n) \rceil$ as otherwise $f(r, \lceil \beta(n) \rceil)$ would dominate $f(\bar{x})$. □

For the example reported in Figure 1 with three objective functions and one variable, the variable in exam needs to be fixed to $\{-1, 0, 1, \dots, 8\}$.

On the level $d = n$ we have to consider exactly all possible fixings of x_n to the integers between $\lfloor \alpha(n) \rfloor$ and $\lceil \beta(n) \rceil$. On all the other levels $d \leq n - 1$, we first have to fix x_d to values within the interval $[\lfloor \alpha(d) \rfloor, \lceil \beta(d) \rceil]$, but we also need to consider fixings outside this interval, as it will be clarified in the following. We can use a bounding condition to stop the enumeration outside the interval as well as on the deeper nodes. This condition should make use of an easy to calculate lower bound as the one presented in (3) defined by the ideal point. Let LB be this lower bound computed at a certain node of the branch-and-bound tree. In order to decide if the node can be pruned, we need to check whether there exists an upper bound that dominates LB. This is expressed by the following condition

Condition 1.

$$\exists\, z \in \mathcal{L}_{PNS} : \; z \leq LB, \; z \neq LB. \tag{Cond}$$

As stated in the following lemma, (Cond) is a sufficient condition for pruning.

Lemma 3.3. *Let $d \in \{1, \dots, n-1\}$ and let $LB \in \mathbb{R}^m$ be the lower bound computed at the node where the variables $(x_1, \dots, x_d)$ are fixed to $(r_1, \dots, r_d) \in \mathbb{Z}^d$. Let* (Cond) *hold. Then there is no efficient point $\bar{x} \in \mathbb{Z}^n$ of the multiobjective optimization problem (*MOQIP*) with $(\bar{x}_1, \dots, \bar{x}_d) = (r_1, \dots, r_d)$.*

Proof. Let $\bar{x} \in \mathbb{Z}^n$ be an arbitrary feasible point of (MOQIP) with $(\bar{x}_1, \dots, \bar{x}_d) = (r_1, \dots, r_d)$. As LB is a lower bound we have that there exists $\ell \in LB$ (here, as LB is defined in (3), $\ell = LB$) with $\ell \leq f(\bar{x})$. From (Cond), we have that $z \in \mathcal{L}_{PNS}, z \neq \ell$ exists such that $z \leq \ell$. Since $z \in \mathcal{L}_{PNS}$, a point $\tilde{x} \in \mathcal{E} \subseteq \mathbb{Z}^n$ exists with $z = f(\tilde{x})$ and hence

$$f(\tilde{x}) = z \leq \ell \leq f(\bar{x}), \; f(\tilde{x}) \neq f(\bar{x})$$

so that $f(\bar{x})$ is dominated by $f(\tilde{x})$ and thus no efficient point of (MOQIP) is such that $(\bar{x}_1, \dots, \bar{x}_d) = (r_1, \dots, r_d)$. □

Assume that we are at a node where the first d variables are fixed to integer values. By convexity, as soon as condition (Cond) is satisfied at a node where x_d is fixed to a value δ strictly greater than $\lceil \beta(d) \rceil$, the node can be pruned as well as all its outer siblings. In other words, we can prune all the nodes where x_d is fixed to integer values strictly greater than δ, as no efficient point can be obtained fixing x_d to those values. This is stated in Lemma 3.4

The same applies as soon as condition (Cond) is satisfied at a node where x_d is fixed to a value δ strictly smaller than $\lfloor \alpha(d) \rfloor$. Namely, we can prune all the nodes obtained where x_d is fixed to integer values strictly smaller than δ, as no efficient point can be obtained fixing x_d to those values, cf. Lemma 3.5.

Lemma 3.4. *Let $d \in \{1, \ldots, n-1\}$ and let $LB \in \mathbb{R}^m$ be the lower bound computed at the node where $(x_1, \ldots, x_d)$ are fixed to $(r_1, \ldots, r_d) \in \mathbb{Z}^d$, where $r_d = \delta > \lceil \beta(d) \rceil$. Let* (Cond) *hold. Then there is no efficient point $\bar{x} \in \mathbb{Z}^n$ of (*MOQIP*) such that $(\bar{x}_1, \ldots, \bar{x}_{d-1}) = (r_1, \ldots, r_{d-1})$ and $\bar{x}_d > \delta$.*

Proof. By (Cond) there exists $z \in \mathcal{L}_{PNS}$ and thus $\tilde{x} \in \mathcal{E} \subseteq \mathbb{Z}^n$ with $z = f(\tilde{x})$ and $z \leq LB$, $z \neq LB$. Let $\bar{x} \in \mathbb{Z}^n$ be an arbitrary feasible point of (MOQIP) with $(\bar{x}_1, \ldots, \bar{x}_{d-1}) = (r_1, \ldots, r_{d-1})$ and $\bar{x}_d > \delta$. We will show that it holds for $j \in \{1, \ldots, m\}$

$$LB_j \leq f_j(\bar{x}) \tag{5}$$

and thus $f(\tilde{x}) \leq f(\bar{x})$ and $f(\tilde{x}) \neq f(\bar{x})$ and the lemma is proved.

To show (5), let $j \in \{1, \ldots, m\}$ and denote by $\bar{f}_j : \mathbb{R}^{n-d+1} \to \mathbb{R}$ the function

$$\bar{f}_j(z) = f_j(r_1, \ldots, r_{d-1}, z_1, \ldots, z_{n-d+1}).$$

The unique individual minimizers $u^{*j} \in \mathbb{R}^{n-d+1}$ of these functions determine the interval $[\lfloor \alpha(d) \rfloor, \lceil \beta(d) \rceil]$, cf. (2), and we have

$$u_1^{*j} \leq \beta(d) \leq \lceil \beta(d) \rceil. \tag{6}$$

Let $\gamma \geq u_1^{*j}$ and let $P(\gamma)$ be the optimization problem

$$\min_{z \in \mathbb{R}^{n-d+1}} \{\bar{f}_j(z) \mid z_1 \geq \gamma\}. \tag{$P(\gamma)$}$$

We show that it holds for any optimal solution $\bar{z}$ of $P(\gamma)$ that $\bar{z}_1 = \gamma$. For $\gamma = u_1^{*j}$, we simply have that $\bar{z} = u^{*j}$ is the minimzer with $\bar{z}_1 = \gamma$. For $\gamma > u_1^{*j}$ the point u^{*j} is no longer feasible for $P(\gamma)$ and thus we have $\nabla \bar{f}_j(\bar{z}) \neq 0$. Based on the Karush-Kuhn-Tucker conditions this implies that the Lagrange multiplier to the constraint $z_1 \geq \gamma$ has to be positive and thus, due to complementarity slackness, that $\bar{z}_1 = \gamma$.

Now, we define a map $x^{j*} : [u_1^{*j}, \infty) \to \mathbb{R}^{n-d}$ by

$$(\gamma, (x^{j*}(\gamma)) \text{ is the unique minimal solution of } (P(\gamma)).$$

So, for $u_1^{*j} \leq \gamma_1 \leq \gamma_2$ we have that $(\gamma_2, x^{j*}(\gamma_2))$ is feasible for $(P(\gamma_1))$ and thus

$$\bar{f}_j(\gamma_1, x^{j*}(\gamma_1)) \leq \bar{f}_j(\gamma_2, x^{j*}(\gamma_2)).$$

As a consequence, as $r_d > u_1^{*j}$, for $\gamma_1 = r_d = \delta < \bar{x}_d = \gamma_2$ we get

$$\bar{f}_j(r_d, x^{j*}(r_d)) \leq \bar{f}_j(\bar{x}_d, x^{j*}(\bar{x}_d)). \tag{7}$$

Moreover, $(\bar{x}_d, \bar{x}_{d+1}, \ldots, \bar{x}_n)$ is feasible for problem $(P(\bar{x}_d))$ and thus

$$\bar{f}_j(\bar{x}_d, x^{j*}(\bar{x}_d)) \leq \bar{f}_j(\bar{x}_d, \ldots, \bar{x}_n). \tag{8}$$

As $\bar{f}_j(\bar{x}_d, \ldots, \bar{x}_n) = f_j(\bar{x})$, we get from (7) and (8) $\bar{f}_j(r_d, x^{j*}(r_d)) \leq f_j(\bar{x})$.

Finally, recall that LB_j is computed by fixing $(x_1, \ldots, x_d)$ to $(r_1, \ldots, r_d)$ and by (3), i.e.,

$$\begin{aligned} LB_j &= \min\{f_j(r_1, \ldots, r_d, v_1, \ldots, v_{n-d}) \mid v \in \mathbb{R}^{n-d}\} \\ &= \min\{\bar{f}_j(z_1, \ldots, z_{n-d+1}) \mid z_1 = r_d\} \\ &= \min\{\bar{f}_j(z_1, \ldots, z_{n-d+1}) \mid z_1 \geq r_d\} \\ &= \bar{f}_j(r_d, x^{j*}(r_d)) \\ &\leq f_j(\bar{x}), \end{aligned}$$

and (5) is shown. □

Analogously, we can prove that as soon as we can prune a node where x_d is fixed to a value $\delta < \lfloor \alpha(d) \rfloor$, we can prune all its outer siblings as no efficient point will have $x_d < \delta$.

Lemma 3.5. *Let $d \in \{1, \ldots, n-1\}$ and let $LB \in \mathbb{R}^m$ be the lower bound computed at the node where $(x_1, \ldots, x_d)$ are fixed to $(r_1, \ldots, r_d) \in \mathbb{Z}^d$, where $r_d = \delta < \lfloor \alpha(d) \rfloor$. Let* (Cond) *hold. Then there is no efficient point $\bar{x} \in \mathbb{Z}^n$ of (*MOQIP*) such that $(\bar{x}_1, \ldots, \bar{x}_{d-1}) = (r_1, \ldots, r_{d-1})$ and $\bar{x}_d < \delta$.*

Using similar arguments as those used in [3, p. 382] we can show that at every level $d \in \{1, \ldots, n-1\}$, condition (Cond) is satisfied after enumerating a finite number of nodes. Note that we already showed in Lemma 3.2 that fixing x_n to values within the interval $[\lfloor \alpha(n) \rfloor, \lceil \beta(n) \rceil]$ suffices to find all efficient points.

Lemma 3.6. *Let $d \in \{1, \ldots, n-1\}$ and let the variables $(x_1, \ldots, x_{d-1})^T$ be fixed to $(r_1, \ldots, r_{d-1})^T \in \mathbb{Z}^{d-1}$. Then there exists $\gamma \in \mathbb{Z}$ such that condition* (Cond) *is satisfied when x_d is fixed to integer values outside the interval $[\lfloor \alpha(d) \rfloor - \gamma, \lceil \beta(d) \rceil + \gamma]$.*

Proof. Let $z \in \mathcal{L}_{PNS}$ and let $j \in \{1, \ldots, m\}$. Note that $\mathcal{L}_{PNS}$ is not empty as it contains at least the feasible integer point chosen in Algorithm 2. Recall that

$$\alpha(d) \leq x_1^{*j,r} \quad \text{and} \quad \beta(d) \geq x_1^{*j,r}$$

where $x^{*j,r}$ is defined in (1) and is obtained with respect to the fixing of the variables $(x_1, \ldots, x_{d-1})^T$ to $(r_1, \ldots, r_{d-1})^T$. We study now the continuous minima of f_j^r, where $(x_1, \ldots, x_d)^T$ are fixed to $(r_1, \ldots, r_{d-1}, t)^T$ with t fixed to

$$\lceil x_1^{*j,r} \rceil \,,\, \lceil x_1^{*j,r} \rceil + 1 \,,\, \ldots .$$

Recall that the continuous minima of f_j^r are the lower bounds which we consider in Algorithm 3, see also (3). As f_j^r is symmetric w.r.t. $x_1^{*j,r}$ and convex, see Remark 2.1, the continuous minima with respect to these fixings are non-decreasing. See also the arguments in [3, p. 382].

In particular, a value $u^j \in \mathbb{Z}$ exists such that fixing t to $\lceil x_1^{*j,r} \rceil + u^j$ will lead to a lower bound that exceeds z_j. The same holds when fixing t to $\lceil \beta(d) \rceil + u^j$, as $\lceil \beta(d) \rceil + u^j \geq \lceil x_1^{*j,r} \rceil + u^j$.

Consider now t fixed to $\lfloor x_1^{*j,r} \rfloor \,,\, \lfloor x_1^{*j,r} \rfloor - 1 \,, \ldots$. Using the same arguments as before, we have that $l^j \in \mathbb{Z}$ exists such that fixing t to $\lfloor x_1^{*j,r} \rfloor - l^j$ will lead to a lower bound that exceeds z_j. The same holds when fixing t to $\lfloor \alpha(d) \rfloor - l^j$, as $\lfloor x_1^{*j,r} \rfloor - l^j \geq \lfloor \alpha(d) \rfloor - l^j$.

By setting $\gamma := \max\{\max_{j=1,\ldots,m} l^j, \max_{j=1,\ldots,m} u^j\}$, we obtain that fixing x_d outside the interval $[\lfloor \alpha(d) \rfloor - \gamma \,,\, \lceil \beta(d) \rceil + \gamma]$ will lead to a lower bound larger than z_j and thus (Cond) is satisfied. □

We are finally able to state the following

Theorem 3.7. *Algorithm 3 stops returning with $\mathcal{E}$ the efficient set and with $\mathcal{L}_{PNS}$ the set of nondominated points of problem* (MOQIP).

Proof. It is sufficient to notice that Algorithm 3 enumerates the integer solutions in the decision space. Thanks to the pruning strategy and the results stated in Lemma 3.2, Lemma 3.4, Lemma 3.5 and in Lemma 3.6 an infinite enumeration is avoided and no efficient solution is lost. □

3.2. Alternative Lower Bounds

Algorithm 3 reports the scheme of `BB-MOQIP` where at each node the lower bound is given by the ideal point as defined in (3). More precisely, assume that we are at a node of level d where variables $(x_1, \ldots, x_d)^T$ are fixed to $r = (r_1, \ldots, r_d)^T$. The lower bound considered in Algorithm 3 is defined as

$$LB := (f_1^r(x^{*1,r}), \ldots, f_m^r(x^{*m,r}))^T,$$

where $x^{*j,r}$ is the unconstrained minimum of f_j^r, $j = 1, \ldots, m$.

As done in [3], this lower bound can be improved considering suitably defined lattice free ellipsoids. For each f_j^r, we can apply Proposition 1 in [3] and obtain a value $\epsilon_j \geq 0$ such that

$$\begin{array}{rcl} \min\{f_j^r(x) \mid x \in \mathbb{Z}^{n-d}\} & \geq & \min\{f_j^r(x) \mid x \in \mathbb{R}^{n-d}\} + \epsilon_j \\ & \geq & \min\{f_j^r(x) \mid x \in \mathbb{R}^{n-d}\} \\ & = & f_j^r(x^{*j,r}). \end{array}$$

A further improvement can be achieved using ellipsoids having the so called *strong rounding property*, as detailed in [7].

Another possibility in defining lower bounds for problem (MOQIP) is that of refining the outer approximations of $f(\mathbb{Z}^n)$ given by $LB + \mathbb{R}^m$ with LB as defined in (3). This means replacing the ideal point with a set $LB \subseteq \mathbb{R}^m$ which is not made of a singleton only. In this case we have to replace (Cond) by the following condition

Condition 2.

$$\forall \ell \in LB \quad \exists\, z \in \mathcal{L}_{PNS} :\ z \leq \ell,\ z \neq \ell. \tag{Cond2}$$

This condition is equivalent to (Cond) in case the set LB is a singleton. It can easily be checked that Lemma 3.3 still holds for any lower bound set LB, i.e., also (Cond2) is a sufficient condition for pruning.

In order to refine the outer approximation of $f(\mathbb{Z}^n)$, we can look for supporting hyperplanes of $f(\mathbb{R}^n)$. We recall the definition of a supporting hyperplane of a set:

Definition 3.8. *Let $P \subseteq \mathbb{R}^m$ be a nonempty set, let $\lambda \in \mathbb{R}^m \setminus \{0\}$ and $z \in \partial P$, where ∂P is the boundary of the set P. The hyperplane*

$$H^{\lambda,z} := \{y \in \mathbb{R}^m \mid \lambda^T y = \lambda^T z\}$$

is called supporting hyperplane (of P), if $\lambda^T y \geq \lambda^T z$ holds for all $y \in P$.

Let $W := \{y \in \mathbb{R}^m_+ \mid \|y\|_1 = 1\}$ and let $\varphi\colon W \to \mathbb{R}$ be the following map

$$w \mapsto \min_{x \in \mathbb{R}^n} w^T f(x).$$

The optimization problem in the definition of φ is also known as weighted sum scalarization. By choosing positive weights w_i, $i = 1, \ldots, m$, any optimal solution gives an efficient solution of (MOQP). We use it here for adding cutting planes to improve the outer approximation of the set $f(\mathbb{R}^n) + \mathbb{R}^m_+$. This idea of improving the ideal point as lower bound by a lower bound set calculated by using a weighted sum on a relaxed problem was proposed for multiobjective linear integer problems in [12], see also the survey [22].

We get that $\{y \in \mathbb{R}^m \mid w^T y = \varphi(w)\}$ is a supporting hyperplane of the set $f(\mathbb{R}^n)$ for any $w \in W$. Since $f(\mathbb{Z}^n) \subseteq f(\mathbb{R}^n)$ we have that

$$f(\mathbb{Z}^n) \subseteq \{y \in \mathbb{R}^m \mid w^T y \geq \varphi(w)\}$$

for any $w \in W$.

Therefore, at each node of our branch-and-bound algorithm, we can consider a bunch of $w \in W$, compute the corresponding supporting hyperplanes and use the resulting outer approximation of $f^r(\mathbb{Z}^{n-d})$ derived from these additional cuts as lower bound. The expensive computations can again be moved to the preprocessing.

To be more precise, let $w \in W$ and consider the following: $Q_w := \sum_{j=1}^m w_j Q_j$, $c_w := \sum_{j=1}^m w_j c^j$, and $a_w = \sum_{j=1}^m w_j a_j = w^T a$. Since $w \in \mathbb{R}^m_+$, $w \neq 0$, we have that Q_w is positive definite and from the first order optimality conditions we get

$$\varphi(w) = \min_{x \in \mathbb{R}^n} w^T f(x) = a_w - \frac{1}{4}(c_w)^T (Q_w)^{-1} c_w.$$

Assume that we are at a node of level d where variables $(x_1, \ldots, x_d)^T$ are fixed to $r = (r_1, \ldots, r_d)^T$. We can define $\varphi^r \colon W \to \mathbb{R}$ as

$$w \mapsto \min_{x \in \mathbb{R}^{n-d}} w^T f^r(x) = a^r_w - \frac{1}{4}(c^r_w)^T (Q^d_w)^{-1} c^r_w,$$

where $Q^d_w := \sum_{j=1}^m w_j Q^d_j$, $c^r_w := \sum_{j=1}^m w_j \bar{c}^{j,r}$, and $a^r_w = \sum_{j=1}^m w_j \bar{a}_{j,r}$.

Therefore, if we fix a bundle of $w \in W$ in advance, we can compute the inverse of the matrices Q^d_w in the preprocessing phase of `BB-MOQIP`, allowing a fast computation of the desired outer approximation of $f^r(\mathbb{Z}^{n-d})$. More precisely, let $\widetilde{W}$ be a finite subset of W. As we want to refine the outer approximation obtained by the ideal point, we assume $e^j \in \widetilde{W}$, for all $j = 1, \ldots, m$, see (4) on page 10. Then it holds

$$f^r(\mathbb{Z}^{n-d}) \subseteq \bigcap_{w \in \widetilde{W}} \{y \in \mathbb{R}^m \mid w^T y \geq \varphi^r(w)\}, \tag{9}$$

so that we can set the lower bound LB as the boundary of the closed set on the right hand side in (9):

$$LB = \partial\Big(\bigcap_{w \in \widetilde{W}} \{y \in \mathbb{R}^m \mid w^T y \geq \varphi^r(w)\}\Big). \tag{10}$$

In order to properly generalize the scheme of `BB-MOIQP`, we need to adapt the calculation of $\alpha(d+1)$ and $\beta(d+1)$ as follows:

$$\alpha(d+1) := \min\{\min_{j=1,\ldots,m} x_1^{*j,r},\ \min_{w \in \widetilde{W}} x_1^{*w,r}\},\ \beta(d+1) := \max\{\max_{j=1,\ldots,m} x_1^{*j,r},\ \max_{w \in \widetilde{W}} x_1^{*w,r}\}, \tag{11}$$

where

$$x^{*w,r} = -\frac{1}{2}(Q^d_w)^{-1} c^r_w \in \mathbb{R}^{n-d}. \tag{12}$$

are the unconstrained minimizers of $w^T f^r(x)$. Note that the first expressions in the brackets in (12) could be omitted as $e^j \in \widetilde{W}$ for $j = 1, \ldots, m$.

As soon as $LB \subseteq \mathbb{R}^m$ is computed as in (10), we need to use the pruning condition (Cond2). For $m = 2$ the evaluation of condition (Cond2) can still be done easily by using geometrical arguments even if this is numerically more time consuming than just evaluating (Cond). For $m \geq 3$, for an efficient numerical evaluation of (Cond2) the use of the concept of so called local upper bounds (see [19] for its original definition and [9] for an application of the concept within multiobjective

mixed integer programming) is necessary. We do not go into details on this, as it is beyond the scope of the present work.

As already mentioned, Lemma 3.3 still holds when considering condition (Cond2) and $LB \subseteq \mathbb{R}^m$ computed as in (10). As the unit vectors e^j, $j = 1, \dots, m$, are contained in $\widetilde{W}$, (Cond) with LB as the ideal point is a stronger condition than the new (Cond2) with LB as in (10). Then also Lemma 3.6 transfers, and together with the following results we obtain finiteness and exactness as in Theorem 3.7 also for this alternative lower bound. In the following, we state a generalization of Lemma 3.4 (and then of Lemma 3.5), showing that (Cond2) is sufficient for pruning outer siblings when $LB \subseteq \mathbb{R}^m$ is computed as in (10).

Lemma 3.9. *Let $\widetilde{W}$ be a finite subset of W. Let $d \in \{1, \dots, n-1\}$ and let $LB \subseteq \mathbb{R}^m$ be the lower bound computed as in (10) at the node where $(x_1, \dots, x_d)$ are fixed to $(r_1, \dots, r_d) \in \mathbb{Z}^d$, where $r_d = \delta > \lceil \beta(d) \rceil$ and $\beta(d)$ as in (11). Let* (Cond2) *hold. Then there is no efficient point $\bar{x} \in \mathbb{Z}^n$ of (*MOQIP*) such that $(\bar{x}_1, \dots, \bar{x}_{d-1}) = (r_1, \dots, r_{d-1})$ and $\bar{x}_d > \delta$.*

Proof. We denote with $LB^{r_d} := LB$ the lower bound derived from (10) at the node where $(x_1, \dots, x_d)$ are fixed to $(r_1, \dots, r_d) \in \mathbb{Z}^d$. By (Cond2) it holds

$$LB^{r_d} \subseteq \mathcal{L}_{PNS} + \mathbb{R}^m_+.$$

Moreover, let $LB^{\bar{x}_d}$ denote the lower bound derived from (10) at a node where $(\bar{x}_1, \dots, \bar{x}_{d-1}) = (r_1, \dots, r_{d-1})$ and $\bar{x}_d > \delta$. To show the result, we prove that

$$LB^{\bar{x}_d} \subseteq LB^{r_d} + \mathbb{R}^m_+. \tag{13}$$

The inclusion in (13) implies $LB^{\bar{x}_d} \subseteq \mathcal{L}_{PNS} + \mathbb{R}^m_+$ and as a consequence also that node can be pruned by using Lemma 3.3.

For proving (13), we show that given any $w \in \widetilde{W}$, the value $\varphi^r(w)$ which appears in (10) increases with the component r_d. In particular, we show that

$$\varphi^{\delta}(w) \leq \varphi^{\bar{x}_d}(w), \tag{14}$$

where we denote by $\varphi^{\bar{x}_d}(w)$ the value $\varphi^r(w)$ with $r = (r_1, \dots, r_{d-1}, \bar{x}_d) \in \mathbb{R}^d$ and by $\varphi^{\delta}(w)$ the value $\varphi^r(w)$ with $r = (r_1, \dots, r_{d-1}, \delta) \in \mathbb{R}^d$ with $\delta > \lceil \beta(d) \rceil$. Inequality (14) implies that

$$\bigcap_{w \in \widetilde{W}} \{y \in \mathbb{R}^m \mid w^T y \geq \varphi^{\bar{x}_d}(w)\} \subseteq \bigcap_{w \in \widetilde{W}} \{y \in \mathbb{R}^m \mid w^T y \geq \varphi^{\delta}(w)\}$$

and then (13) holds. We fix $w \in \widetilde{W}$. Let $\gamma \in \mathbb{R}$ and consider the parameter optimization problem

$$\begin{array}{ll} \min & w^T(\bar{f}_1(z), \dots, \bar{f}_m(z))^T \\ \text{s.t.} & z_1 = \gamma \\ & z \in \mathbb{R}^{n-d+1} \end{array} \tag{$PW(\gamma)$}$$

where $\bar{f}_j : \mathbb{R}^{n-d+1} \to \mathbb{R}$, $j \in \{1, \ldots, m\}$ is the function

$$\bar{f}_j(z) = f_j(r_1, \ldots, r_{d-1}, z_1, \ldots, z_{n-d+1}).$$

The minimal value of $(PW(\delta))$ equals $\varphi^{\delta}(w)$ and the minimal value of $(PW(\bar{x}_d))$ equals $\varphi^{\bar{x}_d}(w)$. Moreover, let $u^* \in \mathbb{R}^{n-d+1}$ be the unconstrained minimizer of the objective function of $(PW(\cdot))$, i.e.,

$$u^* = \operatorname{argmin}\{w^T(\bar{f}_1(z), \ldots, \bar{f}_m(z))^T \mid z \in \mathbb{R}^{n-d+1}\}.$$

As $w^T\bar{f}$ is a strictly convex function, the minimizer is unique. As a consequence of the calculation of $\beta(d)$ according to (11) we have that $u_1^* \leq \beta(d)$ and thus $u_1^* \leq \lceil \beta(d) \rceil$.

Now we look at

$$\begin{array}{ll} \min & w^T(\bar{f}_1(z), \ldots, \bar{f}_m(z))^T \\ \text{s.t.} & z_1 \geq \gamma \\ & z \in \mathbb{R}^{n-d+1} \end{array} \qquad (PW'(\gamma))$$

for $\gamma \geq u_1^*$. With the same arguments as in the proof of Lemma 3.4, for any optimal solution of $(PW'(\gamma))$ it holds that the inequality constraint is active in the minimal solution. We define a map $z^* : [u_1^*, \infty) \to \mathbb{R}^{n-d}$ by

$$(\gamma, (z^*(\gamma)) \text{ is the unique minimal solution of } (PW'(\gamma)).$$

So, for $u_1^* \leq \gamma_1 \leq \gamma_2$ we have that $(\gamma_2, z^*(\gamma_2))$ is feasible for $(PW'(\gamma_1))$ and thus

$$w^T\bar{f}(\gamma_1, z^*(\gamma_1)) \leq w^T\bar{f}(\gamma_2, z^*(\gamma_2)).$$

As a consequence, as $r_d = \delta > u_1^*$, for $\gamma_1 = r_d = \delta < \bar{x}_d = \gamma_2$ we get

$$\varphi^{\delta}(w) = w^T\bar{f}(r_d, z^*(r_d)) \leq w^T\bar{f}(\bar{x}_d, z^*(\bar{x}_d)) = \varphi^{\bar{x}_d}(w)$$

and (14) is shown. □

With the same arguments one can prove:

Lemma 3.10. *Let $\widetilde{W}$ be a finite subset of W. Let $d \in \{1, \ldots, n-1\}$ and let $LB \subseteq \mathbb{R}^m$ be the lower bound computed as in (10) at the node where $(x_1, \ldots, x_d)$ are fixed to $(r_1, \ldots, r_d) \in \mathbb{Z}^d$, where $r_d = \delta < \lfloor \alpha(d) \rfloor$ and $\alpha(d)$ as in (11). Let* (Cond2) *hold. Then there is no efficient point $\bar{x} \in \mathbb{Z}^n$ of (*MOQIP*) such that $(\bar{x}_1, \ldots, \bar{x}_{d-1}) = (r_1, \ldots, r_{d-1})$ and $\bar{x}_d < \delta$.*

3.3. Bound constrained problems

In case the formulation of Problem (MOQIP) includes bounds on the variables, Algorithm 3 still works, as soon as the bounds are taken into account in the enumeration of the nodes and the computation of $\alpha(d)$ and $\beta(d)$ is properly modified.

Assume that each variable $x_i, i = 1, \dots, n$ is constrained to be within the interval $[l_i, u_i]$. Assume that we are at a node of level d where the variables $(x_1, \dots, x_d)^T$ are fixed to $r = (r_1, \dots, r_d)^T$ and $x^{*j,r}$ is obtained as in (1). Then, the values $\alpha(d+1)$ and $\beta(d+1)$ should be computed as follows:

$$\begin{aligned} \alpha(d+1) &:= \min\left\{u_{d+1}, \max\{l_{d+1}, \min_{j=1,\dots,m} x_1^{*j,r}\}\right\}, \\ \beta(d+1) &:= \max\left\{l_{d+1}, \min\{u_{d+1}, \max_{j=1,\dots,m} x_1^{*j,r}\}\right\}. \end{aligned}$$

Clearly, we do not have to enumerate those nodes for which x_d is fixed to values outside the interval $[l_d, u_d]$. Thus also the pruning condition (Cond) as well as the updates of $\mathcal{E}$ and $\mathcal{L}_{PNS}$ should include a feasibility check.

4. Toy example

In this section we show the behavior of the branch-and-bound algorithm `BB-MOQIP` on the following simple example

$$\begin{array}{ll} \min & (f_1(x), f_2(x))^T \\ \text{s.t.} & x \in \mathbb{Z}^2, \end{array} \tag{15}$$

where

$$f_1(x) = x_1^2 + x_1x_2 + x_2^2, \text{ so that } Q_1 = \begin{pmatrix} 1 & 0.5 \\ 0.5 & 1 \end{pmatrix}, \quad c^1 = (0,0)^T, \quad a_1 = 0;$$

$$f_2(x) = x_1^2 + x_2^2 - 2x_1 - 2x_2, \text{ so that } Q_2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad c^2 = (-2,-2)^T, \quad a_2 = 0.$$

In Figure 2, we plot the decision and the criterion space of Problem (15), where we restrict the plot in the decision space to $[-1, 2] \times [-1, 1]$. In the subfigures, the efficient and nondominated solutions of Problem (15) are circled. The efficient set of Problem (15) is $\{(0,0)^T,\ (0,1)^T,\ (1,0)^T,\ (1,1)^T\}$ corresponding to the nondominated points $\{(0,0)^T,\ (1,-1)^T,\ (3,-2)^T\}$.

Our branch-and-bound algorithm starts by choosing an order in which the variables are fixed. Let us choose the natural order x_1, x_2. In the preprocessing phase Q_j^d as well as $(Q_j^d)^{-1}$ are computed, for $j = 1, \dots, m$ and $d = 1, \dots, n$. In our example we have $Q_1^1 = (Q_1^1)^{-1} = Q_2^1 = (Q_2^1)^{-1} = (1)$, and the inverse matrices of Q_1 and Q_2. As initial feasible solution we take $r^* = (0,0)^T \in \mathbb{Z}^2$, so that $\mathcal{E} \leftarrow \{r^*\}$ and $\mathcal{L}_{PNS} \leftarrow \{f(r^*)\} = \{z^1\}$ with $z^1 := (0,0)^T$. We enter in the while loop with $d = 0$. The resulting branch-and-bound tree with all visited nodes is shown in Figure 3 on page 24.

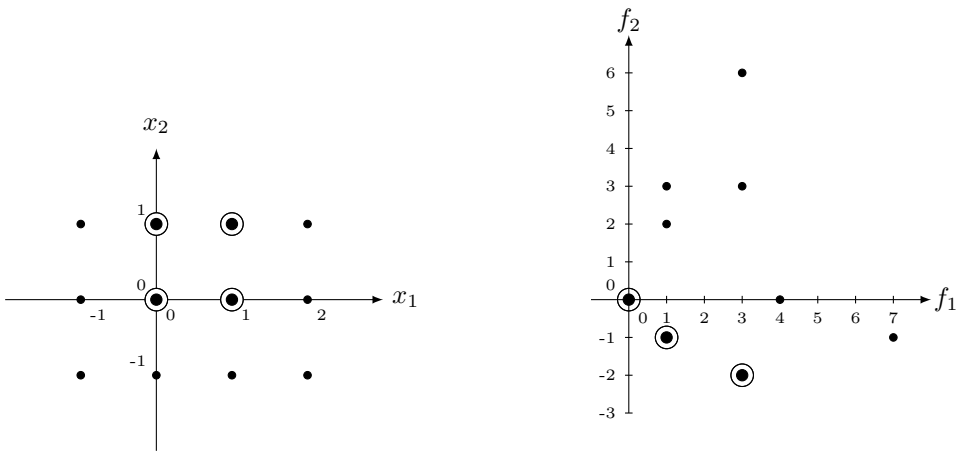

Figure 2: The decision (left subfigure) and the criterion space (right subfigure) of Problem (15).

1st iteration.. The unconstrained minimizers of f_1 and f_2 are $x^{*1} = (0,0)^T$ and $x^{*2} = (1,1)^T$, respectively, so that we set

$$LB := (f_1(x^{*1}), f_2(x^{*2}))^T = (0,-2)^T, \qquad \alpha(1) = 0, \quad \beta(1) = 1.$$

Condition (Cond) is not satisfied as $LB_2 < 0 = z_2^1$. We then set $d = 1$, $r_1 = \lfloor \alpha(1) \rfloor = 0$ and fix x_1 to r_1, namely $x_1 = 0$.

2nd iteration.. The unconstrained minimizers of f_1^r and f_2^r, taking into account $r_1 = 0$ are $x^{*1,r} = 0$ and $x^{*2,r} = 1$, respectively, so that we set

$$LB := (f_1^r(x^{*1,r}), f_2^r(x^{*2,r}))^T = (0,-1)^T, \qquad \alpha(2) = 0, \quad \beta(2) = 1.$$

Condition (Cond) is not satisfied as $LB_2 < 0 = z_2^1$. We then set $d = 2$, $r_2 = \lfloor \alpha(2) \rfloor = 0$ and fix x_2 to r_2, namely $x_2 = 0$.

3rd iteration.. We have $r = (0,0)^T$, already contained in $\mathcal{E}$. Since $r_2 \in [\lfloor \alpha(2) \rfloor, \lceil \beta(2) \rceil) = [0,1)$, we fix $x_2 = r_2 + 1 = 1$ and set $r_2 = x_2 = 1$.

4th iteration.. We have $r = (0,1)^T$ and $f(r) = (1,-1)^T =: z^2$, so that $\mathcal{E}$ and $\mathcal{L}_{PNS}$ are updated as

$$\mathcal{E} = \{(0,0)^T, (0,1)^T\}, \quad \mathcal{L}_{PNS} = \{z^1, z^2\} = \{(0,0)^T, (1,-1)^T\}.$$

Since $r_2 = 1 = \lceil \beta(2) \rceil$, we set $d = d - 1 = 1$ and update r_1 according to Algorithm 1, so that we fix $x_1 = r_1 + 1 = 1$ and $r_1 = x_1 = 1$.

5th iteration.. The unconstrained minimizers of f_1^r and f_2^r, taking into account $x_1 = r_1 = 1$ are $x^{*1,r} = -0.5$ and $x^{*2,r} = 1$, respectively, so that we set

$$LB := (f_1^r(x^{*1,r}), f_2^r(x^{*2,r}))^T = (0.75, -2)^T, \qquad \alpha(2) = -0.5, \quad \beta(2) = 1.$$

Condition (Cond) is not satisfied as $LB_2 < -1 = z_2^2 < z_2^1$. We then set $d = 2$, $r_2 = \lfloor \alpha(2) \rfloor = -1$ and fix x_2 to r_2, namely $x_2 = -1$.

6th iteration.. We have $r = (1, -1)^T$, but since $f(r) = (1, 2)^T$ is dominated by z^1 and z^2, the sets $\mathcal{E}$ and $\mathcal{L}_{PNS}$ are not updated. Since $r_2 \in [\lfloor \alpha(2) \rfloor, \lceil \beta(2) \rceil) = [-1, 1)$, we fix $x_2 = r_2 + 1 = 0$ and set $r_2 = x_2 = 0$.

7th iteration.. We have $r = (1, 0)^T$ and $f(r) = (1, -1)^T$. While $f(r)$ is already contained in $\mathcal{L}_{PNS}$, we set $\mathcal{E} = \{(0, 0)^T, (0, 1)^T, (1, 0)^T\}$. As still $r_2 \in [\lfloor \alpha(2) \rfloor, \lceil \beta(2) \rceil) = [-1, 1)$, we fix $x_2 = r_2 + 1 = 1$ and set $r_2 = x_2 = 1$.

8th iteration.. We have $r = (1, 1)^T$ and $f(r) = (3, -2)^T$, so that $\mathcal{E}$ and $\mathcal{L}_{PNS}$ are updated using $z^3 := (3, -2)^T$ as

$$\mathcal{E} = \{(0, 0)^T, (0, 1)^T, (1, 0)^T, (1, 1)^T\}, \quad \mathcal{L}_{PNS} = \{z^1, z^2, z^3\}.$$

Since $r_2 = 1 = \lceil \beta(2) \rceil$, we set $d = d - 1 = 1$ and update r_1 according to Algorithm 1. Since $r_1 = 1 = \lceil \beta(1) \rceil$, we fix $x_1 = \lceil \beta(1) \rceil + 1 = 2$ and $r_1 = x_1 = 2$.

9th iteration.. The unconstrained minimizers of f_1^r and f_2^r, taking into account $x_1 = 2$ are $x^{*1,r} = -1$ and $x^{*2,r} = 1$, respectively, so that we set

$$LB := (f_1^r(x^{*1,r}), f_2^r(x^{*2,r}))^T = (3, -1)^T, \qquad \alpha(2) = -1, \quad \beta(2) = 1.$$

Condition (Cond) is satisfied as LB is dominated by z^3. Since $r_1 = 2 > 1 = \lceil \beta(1) \rceil$ we can prune this node and all its outer siblings and we fix $x_1 = \lfloor \alpha(1) \rfloor - 1 = -1$ and $r_1 = x_1 = -1$.

10th iteration.. The unconstrained minimizers of f_1^r and f_2^r, taking into account $x_1 = -1$ are $x^{*1,r} = 0.5$ and $x^{*2,r} = 1$, respectively, so that we set

$$LB := (f_1^r(x^{*1,r}), f_2^r(x^{*2,r}))^T = (0.75, 2)^T, \qquad \alpha(2) = 0.5, \quad \beta(2) = 1.$$

Condition (Cond) is satisfied as LB is dominated by z^1. Since $r_1 = -1 < 0 = \lfloor \alpha(1) \rfloor$ we set $d = d - 1 = 0$ and break the while loop.

Hence, the branch-and-bound algorithm stops with

$$\mathcal{L}_{PNS} = \{z^1, z^2, z^3\} = \{(0, 0)^T, (1, -1)^T, (3, -2)^T\}$$

after having enumerated 10 nodes.

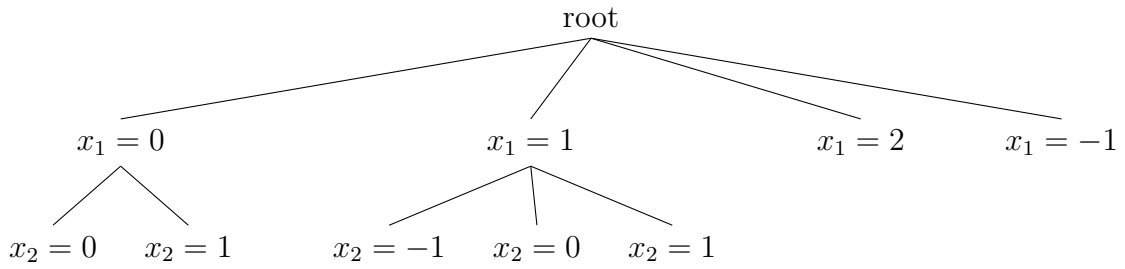

Figure 3: Illustration of the branch-and-bound tree of the toy example.

5. Numerical results

In this section, we present our numerical experience on different instances of (MOQIP). Next to some results on biobjective instances, we show results on instances with $m = 3$ and $m = 4$. All the algorithms considered have been implemented in MATLAB R2019a. In the preprocessing of `BB-MOQIP`, see Algorithm 2, we have always chosen $r^* = 0$ for initialization.

All experiments have been performed on an Intel Core i7 processor running at 3.1 GHz under Linux. An implementation of `BB-MOQIP`, together with the instances used in this numerical experience, is available at [1].

As done in [3], the running time to process a node at level d, can be decreased from $O(n^2)$ to $O(n - d)$, using some adjustments in the preprocessing and in the computation of the ideal point. We implemented these adjustments. However, since additional memory is required and the number of variables in the tests we consider is relatively small, we did not notice an improvement in terms of computational time. Therefore, we show the results obtained by `BB-MOQIP` implemented as reported in the Algorithm 3, where a quadratic time per node is achieved.

5.1. Results on a scalable biobjective instance

In this section, we show results on a biobjective instance of (MOQIP), scalable in the number of variables. As there are no other algorithms which are specifically taylored for unconstrained convex quadratic integer multiobjective problems, we compare `BB-MOQIP` with `MOMIX` [9], with a naïve enumeration procedure (`Enum`) and with the ε-constraint method (ε-`const`), as explained in the following. Thereby, `MOMIX` and `Enum` require bounds on the variables in order to be applicable. For unconstrained test instances we obtain these bounds from the results of `BB-MOQIP`. We clarify this procedure below.

Algorithm `MOMIX` [9] is a decision space algorithm for solving multiobjective convex mixed integer programming problems, based on linear outer approximations of the image set (see [9] for further details). We choose to compare `BB-MOQIP` with `MOMIX` in order to have a comparison between two deterministic decision space search algorithms, as also `MOMIX` gives a guarantee to find all nondominated points.

The enumeration procedure `Enum` simply evaluates the objective functions over the integers belonging to a box $[l, u] \subseteq \mathbb{R}^n$ and extract the efficient points out of them. We choose to compare `BB-MOQIP` with `Enum` to show the power of our pruning strategy and to show that our branch-and-bound is clearly superior to a full enumeration strategy.

The ε-constraint method for biobjective instances of (MOQIP) solves a sequence of parameter-dependent single-objective optimization problems of the following form:

$$\begin{array}{ll} \min & f_2(x) \\ \text{s.t.} & f_1(x) \leq \varepsilon \\ & x \in \mathbb{Z}^n. \end{array} \qquad (\mathrm{P}_\varepsilon)$$

The parameter ε is chosen from the interval $[f_1(x^{*1}), f_1(x^{*2})]$, starting with $\varepsilon = f_1(x^{*2})$, where x^{*j} denotes the minimizer of f_j over $\mathbb{Z}^n$, $j = 1, 2$. Starting from a step size $\delta > 0$, the choice of the parameters can be adapted along the iterations by setting $\varepsilon_{k+1} := f_1(x^k) - \delta$ where x^k denotes a minimal solution of $(\mathrm{P}_{\varepsilon_k})$. In our implementation we considered $\delta = 0.1$. In Table 1 we report both the results obtained considering P_ε (ε-`const` $(f1)$) and the results obtained considering P_ε, where the roles of f_1 and f_2 are exchanged (ε-`const` $(f2)$), namely f_1 is the objective function and f_2 defines the constraint. We choose to compare `BB-MOQIP` with the ε-constraint method to have a comparison with a scalarization method. In our implementation, we consider GUROBI [15] as solver for the convex quadratic integer optimization subproblems (P_ε). We would like to underline that GUROBI [15] is a commercial software and is among the best solvers for mixed integer quadratic optimization. Moreover, the ε-constraint method with the above chosen strategy of δ-steps of the parameter, only calculates representations of the nondominated set, with no guarantee to find the complete efficient set as `BB-MOQIP`. In addition to that, the found points might be weakly efficient points only.

While `BB-MOQIP` does not necessarily need bounds on the variables, some of the other methods in comparison need them as inputs. We proceed as follows. Once the instance is solved by `BB-MOQIP`, we compute the minimum and the maximum value of the components of the efficient points found. These values can be used as bounds for the variables both in `MOMIX`, in `Enum` and have also been used in the ε-constraint method. This is clearly not in favor of `BB-MOQIP`.

The scalable biobjective instance that we considered is the following.

Test instance 1. *Let $Q_1, Q_2 \in \mathbb{R}^{n\times n}$ be the symmetric matrices defined as follows:*

$$(Q_1)_{i,j} = \begin{cases} 7.9 & \text{if } i=j \\ -0.1 & \text{else} \end{cases} \qquad \text{and } (Q_2)_{i,j} = \begin{cases} 0.3 & \text{if } i=j \\ 0 & \text{else,} \end{cases}$$

with $i, j = 1, \ldots, n$. *Then, the instance of* (MOQIP) *is stated as*

$$\begin{aligned} \min \quad & \begin{pmatrix} x^T Q_1^T Q_1 x + (1, 2, 2, \ldots, 2, 2, 1)x \\ x^T Q_2^T Q_2 x + (-1, -2, -2, \ldots, -2, -2, 5)x \end{pmatrix} \\ \text{s.t.} \quad & x \in \mathbb{Z}^n. \end{aligned} \tag{Inst1}$$

Thus $c_j^1 = 2$, $c_j^2 = -2$ for $j = 2, \ldots, n-1$. In our tests, we considered instance (Inst1) with $n = 2, \ldots, 10$. Note that for these choices of n, the matrices Q_1 and Q_2 are positive definite.

We report in Figure 4 the criterion space of the instance, plotting $(f_1(x), f_2(x))$ for all $x \in [-8, 3] \times [-8, 3] \cap \mathbb{Z}^2$. We highlight the efficient points in the decision space (left subfigure) and the corresponding nondominated points in the criterion space. Note that the nondominated points do not all belong to the convex hull of the image points.

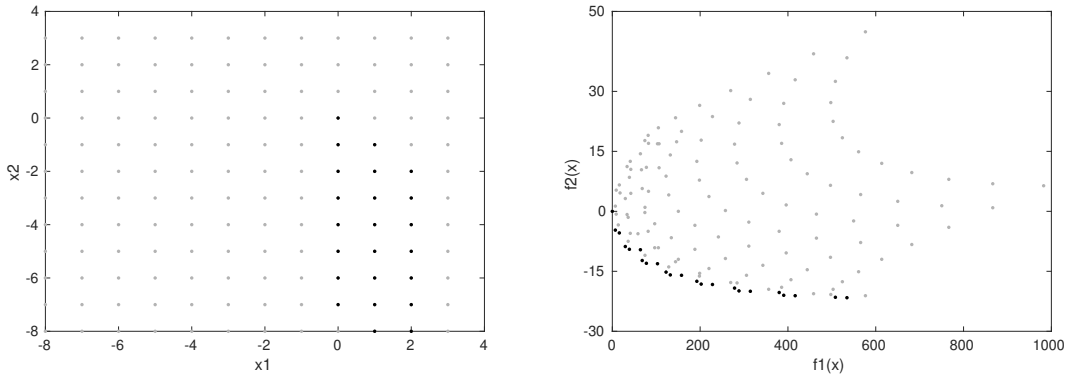

Figure 4: The decision (left subfigure) and the criterion space (right subfigure) of instance (Inst1), with $n = 2$.

The numerical results are shown in Table 1. In the first and the second column of the table, we report the number of integer variables (n) and the number of nondominated points of the instance, respectively. Note that $|\mathcal{L}_{PNS}|$ is by Theorem 3.7 exactly the number of nondominated points. Then, for each method, we report (#nod) and the total computational time in seconds (time), where #nod has a different meaning depending on the method chosen:

For `BB-MOQIP` the number (#nod) represents the number of nodes explored, for `MOMIX` the number (#nod) is the number of considered boxes in the branching tree, for `Enum` it is the total number of objective function evaluations done, for ε`-const` it is the number of subproblems solved. Failures, i.e., instances for which the time limit of 1800 seconds was exceeded, are marked with "-".

As expected, the ε-constraint method is much faster compared to the other approaches. We even only used a basic implementation here (but at least using the commercial solver GUROBI) such that one can expect that these results can

		BB-MOQIP		MOMIX		Enum		ε-const $(f1)$		ε-const $(f2)$	
n	$\lvert\mathcal{L}_{PNS}\rvert$	#nod	time	#nod	time	#nod	time	#nod	time	#nod	time
2	23	102	0.06	87	7.18	144	0.03	22	1.95	22	3.23
3	42	796	0.09	267	36.43	1,728	0.08	41	3.35	39	2.58
4	51	5,069	0.29	631	79.27	20,736	0.55	50	2.87	47	4.01
5	56	29,388	1.30	1,371	254.51	248,832	5.75	56	3.09	53	3.76
6	65	157,019	4.70	2,973	509.62	2,985,984	73.63	67	3.29	59	3.27
7	66	786,374	24.23	5,931	1,106.07	35,831,808	936.95	73	3.77	65	2.13
8	74	3,699,585	118.61	-	-	-	-	80	4.36	71	4.88
9	79	16,702,420	582.07	-	-	-	-	88	5.91	77	11.05
10	89	-	-	-	-	-	-	97	29.69	89	39.56

Table 1: Numerical results for test instances (Inst1).

even be improved. Unlike the other methods, ε-const is a criterion space search algorithm. However, no covering results as those obtained for BB-MOQIP and MOMIX are guaranteed and we can notice that the number of solutions found by ε-const is often less than $|\mathcal{L}_{PNS}|$, i.e., with the procedure as applied here with a chosen step size in advance, it cannot be guaranteed to find all nondominated points. We also want to mention that the ε-constraint method performs particularly well on biobjective instances, while this is not always true when considering more than two objective functions, as the choice of the parameters ε gets more difficult, and that the choice of the stepsize δ is also not clear. However, there exist extensions of the ε-constrained method that try to overcome these issues (see, e.g., [18, 26]).

When looking at the comparison among BB-MOQIP, MOMIX and Enum, we notice the following. We observe that BB-MOQIP outperforms MOMIX in terms of computational time for all $n \leq 9$ while MOMIX is superior with respect to (#nod). This shows how fast our approach enumerates the nodes: thanks to the preprocessing phase, the computation of the lower bounds requires a negligible computational effort.

For $n = 2, 3$ Enum is faster than BB-MOQIP, but then we notice the opposite situation, as the number of function evaluations needed by Enum is at least one order of magnitude greater than the number of nodes enumerated by BB-MOQIP.

With the aim of improving the performance of BB-MOQIP, we have implemented the use of alternative lower bounds by refining the outer approximation of $f(\mathbb{Z}^n)$, as explained in Section 3.2. In the following, we denote by BB-MOQIP-3 and by BB-MOQIP-5 the versions of BB-MOQIP where, at every node, lower bounds are computed according to (10), considering 3 and 5 supporting hyperplanes of $f(\mathbb{R}^n)$, respectively. Note that since BB-MOQIP uses the ideal point as lower bound, it considers the hyperplanes defined using $w = (1, 0)^T$ and $w = (0, 1)^T$. In BB-MOQIP-3, we refine the outer approximation of $f(\mathbb{Z}^n)$ used in BB-MOQIP by considering the hyperplanes defined by

$$w \in \widetilde{W} = \left\{ \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 0.5 \\ 0.5 \end{pmatrix} \right\}.$$

In BB-MOQIP-5, we refine the outer approximation of $f(\mathbb{Z}^n)$ used in BB-MOQIP by considering the hyperplanes defined using

$$w \in \widetilde{W} = \left\{ \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 0.5 \\ 0.5 \end{pmatrix}, \begin{pmatrix} 0.25 \\ 0.75 \end{pmatrix}, \begin{pmatrix} 0.75 \\ 0.25 \end{pmatrix} \right\}.$$

We report in Table 2 the comparison between BB-MOQIP, BB-MOQIP-3 and BB-MOQIP-5 on the test instance (Inst1). Despite the fact that the evaluation of (Cond2) imple-

		BB-MOQIP		BB-MOQIP-3		BB-MOQIP-5	
n	$\lvert\mathcal{L}_{PNS}\rvert$	#nod	time	#nod	time	#nod	time
2	23	102	0.06	102	0.10	102	0.12
3	42	796	0.09	742	0.13	656	0.15
4	51	5,069	0.29	4,724	0.23	4,027	0.33
5	56	29,388	1.30	26,996	0.91	21,878	1.15
6	65	157,019	4.70	140,451	4.67	109,616	3.42
7	66	786,374	24.23	681,332	21.62	510,438	15.32
8	74	3,699,585	118.61	3,114,010	100.07	2,235,497	69.92
9	79	16,702,420	582.07	13,392,984	449.76	9,265,958	300.61
10	89	-	-	-	-	37,226,723	1,280.24

Table 2: Use of alternative lower bounds - test instances (Inst1).

mented in BB-MOQIP-3 and BB-MOQIP-5 is numerically more time consuming than the evaluation of (Cond) as implemented in BB-MOQIP, we can notice for increasing n an improvement in both the number of nodes and the computational time required. As expected, improving the quality of the lower bounds leads to an overall improvement of the algorithm. The motivation is that this improvement in the quality of the lower bounds comes with a small additional computational effort, as the most expensive operations are again performed in the preprocessing phase, which requires 0.01 seconds at most.

5.2. Results on randomly generated multi-objective instances

In this section, we report numerical results on instances of (MOQIP), where the matrices Q_j are randomly generated.

We report in Table 3 the comparison between BB-MOQIP and MOMIX where the number m of objective functions is $2, 3, 4$. We considered instances with a number $n = 2, 5, 10$ of integer variables. For each combination of n and m we produced 10 different instances, where the matrices Q_j are obtained using the MATLAB operator *rand* that generates uniformly distributed pseudorandom numbers. We used $rng(k)$, $k = 1, \ldots, 10$ as seed for the random number generator.

The four positive definite matrices are obtained as follows. Starting from a randomly generated matrix A_j, $j = 1, \ldots, 4$, the matrix Q_j is set to $A_j A_j^T$ in case

the latter is positive definite, otherwise it is set to $A_j A_j^T + I$, where I is the identity matrix. The matrices A_j are computed as follows:

$$A_1 = \texttt{rand}(n), \quad A_2 = 5 * \texttt{rand}(n), \quad A_3 = 2 * \texttt{rand}(n), \quad A_4 = 8 * \texttt{rand}(n).$$

For the linear terms c^i, $i = 1, \dots 4$, we considered the following:

$$\begin{array}{ll} c^1 = (1, 2, \dots, 2, 1)^T, & c^2 = (-1, -2, \dots, -2, 5)^T, \\ c^3 = (-1, \dots, -1, 5)^T, & c^4 = (1, \dots, 1)^T. \end{array}$$

To clarify, we have $c_j^1 = 2$, $c_j^2 = -2$, $c_j^3 = -1$, $c_j^4 = 1$ for $j = 2, \dots, n-1$.

The time-limit was set to 1800 seconds for each individual instance. As already mentioned, `MOMIX` needs bounds on the variables $[l_i, u_i]$, $i = 1, \dots, n$ as input. Therefore, we set $l_i = -100$ and $u_i = 100$, for all $i = 1, \dots, n$, for all instances, in order to have a box large enough to contain the efficient sets.

Table 3 includes the following data for each algorithm: the number of instances solved within the time limit (#sol), the average number of nodes (#nod) and the average running time in seconds (time). We also report the minimum and the maximum number of nondominated points considering the instances solved within the time limit ($\min |\mathcal{L}_{PNS}|$ and $\max |\mathcal{L}_{PNS}|$, respectively).

				BB-MOQIP			MOMIX		
n	m	$\min \lvert\mathcal{L}_{PNS}\rvert$	$\max \lvert\mathcal{L}_{PNS}\rvert$	#sol	#nod	time	#sol	#nod	time
2	2	3	37	10	85.5	0.01	10	111.8	33.07
5	2	4	318	10	661317.6	17.89	5	1722.6	1083.36
10	2	25	197	7	7846846.9	379.27	0	-	-
2	3	9	2048	10	221930.7	48.52	10	109.8	22.04
5	3	23	496	7	841648.3	70.29	4	1334.5	987.99
10	3	46	336	5	9765977.8	697.61	0	-	-
2	4	15	3880	10	274016.1	95.41	10	121.0	27.17
5	4	27	563	7	1047007.0	70.99	4	1508.5	1308.19
10	4	49	460	4	9731083.3	885.56	0	-	-

Table 3: Numerical results on 90 randomly generated instances.

We observe that `BB-MOQIP` and `MOMIX` solved within the time limit 70 and 43 instances out of 90, respectively. Again we can notice how fast the nodes are enumerated by `BB-MOQIP`: when $n = 5, 10$ `BB-MOQIP` strongly outperforms `MOMIX` in terms of computational time. With respect to (#nod), we see that `MOMIX` needs to consider a much smaller number of boxes in the branching tree in comparison to the nodes enumerated by `BB-MOQIP`.

6. Conclusions

We devised a deterministic decision space search algorithm for solving multiobjective strictly convex quadratic integer programming problems. The main feature

of the method consists in a preprocessing phase that enables a fast computation of lower bounds, obtained as the ideal points of the restricted objective functions. Also alternative lover bounds using hyperplanes are examined to improve the results obtained by the ideal points. Theoretical results related to the correctness of the algorithm are provided: the algorithm guarantees to find all efficient and all nondominated points. This property is for instance important in case one wants to optimize another function over the efficient set. It also presents a significant difference to representations generated by scalarization approaches. Numerical examples on biobjective instances as well as on instances with three and four objectives are reported, showing the ability of the algorithm to detect the efficient set of multiobjective strictly convex quadratic integer programming problems.

We are convinced that our theoretical achievements can be used for further developments in the context of multiobjective nonlinear integer optimization. In particular, if combined with the approaches proposed in [5, 6] for computing lower bounds of nonconvex quadratic integer minimization problems, as well as with the active set algorithm developed in [4] for computing lower bounds of convex quadratic integer minimization problems with linear constraints, they can be used to devise decision space algorithms for multiobjective quadratic integer optimization problems.

7. Acknowledgments

The first author acknowledges support within the project No RP1181641D22304F which has received funding from Sapienza, University of Rome. The authors wish to thank the two anonymous referees for their valuable comments and remarks on an earlier version of this manuscript.

[1] `https://github.com/mariannadesantis/BBMOQIP.git`.

[2] V. Blanco and J. Puerto. Some algebraic methods for solving multiobjective polynomial integer programs. *Journal of Symbolic Computation*, 46(5):511–533, 2011.

[3] C. Buchheim, A. Caprara, and A. Lodi. An effective branch-and-bound algorithm for convex quadratic integer programming. *Math. progr.*, 135(1-2):369–395, 2012.

[4] C. Buchheim, M. De Santis, S. Lucidi, F. Rinaldi, and L. Trieu. A feasible active set method with reoptimization for convex quadratic mixed-integer programming. *SIAM J. Optim.*, 26(3):1695–1714, 2016.

[5] C. Buchheim, M. De Santis, and L. Palagi. A fast branch-and-bound algorithm for non-convex quadratic integer optimization subject to linear constraints using ellipsoidal relaxations. *Operations Research Letters*, 43(4):384–388, 2015.

[6] C. Buchheim, M. De Santis, L. Palagi, and M. Piacentini. An exact algorithm for nonconvex quadratic integer minimization using ellipsoidal relaxations. *SIAM J. Optim.*, 23(3):1867–1889, 2013.

[7] C. Buchheim, R. Hübner, and A. Schöbel. Ellipsoid bounds for convex quadratic integer programming. *SIAM J. Optim.*, 25(2):741769, 2015.

[8] C. Buchheim and L. Trieu. Active set methods with reoptimization for convex quadratic integer programming. In *International Symposium on Combinatorial Optimization*, pages 125–136. Springer, 2014.

[9] M. De Santis, G. Eichfelder, J. Niebling, and S. Rocktäschel. Solving multiobjective mixed integer convex optimization problems. *SIAM J. Optim.*, 30(4):3122–3145, 2020.

[10] M. De Santis, G. Grani, and L. Palagi. Branching with hyperplanes in the criterion space: The frontier partitioner algorithm for biobjective integer programming. *Eur. J. Oper. Res.*, 283(1):57–69, 2020.

[11] S. Dogan, O. Karsu, and F. Ulus. An exact algorithm for biobjective integer programming problems. arXiv 1905.07428, 2019.

[12] M. Ehrgott and X. Gandibleux. Bound sets for biobjective combinatorial optimization problems. *Computers & Operations Research*, 34(9):2674 – 2694, 2007.

[13] J. Fernández and B. Tóth. Obtaining the efficient set of nonlinear biobjective optimization problems via interval branch-and-bound methods. *Computational Optimization and Applications*, 42(3):393–419, 2009.

[14] Sune Gadegaard, Lars Nielsen, and Matthias Ehrgott. Bi-objective branch-and-cut algorithms based on lp relaxation and bound sets. *INFORMS Journal on Computing*, 31, 06 2019.

[15] LLC Gurobi Optimization. Gurobi optimizer reference manual, 2020.

[16] Tim Holzmann and J Cole Smith. Solving discrete multi-objective optimization problems using modified augmented weighted tchebychev scalarizations. *European Journal of Operational Research*, 271(2):436–449, 2018.

[17] Parragh, S.N. and F. Tricoire. Branch-and-bound for bi-objective integer programming. *INFORMS J. Comput.*, 31(4):805–822, 2019.

[18] Gokhan Kirlik and Serpil Sayın. A new algorithm for generating all nondominated solutions of multiobjective discrete optimization problems. *European Journal of Operational Research*, 232(3):479–488, 2014.

[19] N. Klamroth, R. Lacour, and D. Vanderpooten. On the representation of the search region in multi-objective optimization. *Eur. J. Oper. Res.*, 245(3):767–778, 2015.

[20] A. Liefooghe, S. Verel, and J-K Hao. A hybrid metaheuristic for multiobjective unconstrained binary quadratic programming. *Applied Soft Computing*, 16:10–19, 2014.

[21] J. Niebling and G. Eichfelder. A branch-and-bound-based algorithm for nonconvex multiobjective optimization. *SIAM J. Optim.*, 29(1):794–821, 2019.

[22] Anthony Przybylski and Xavier Gandibleux. Multi-objective branch and bound. *European Journal of Operational Research*, 260, 01 2017.

[23] D. Quadri, E. Soutif, and P. Tolla. Exact solution method to solve large scale integer quadratic multidimensional knapsack problems. *J. Comb. Optim.*, 17:157–167, 2009.

[24] D. Scholz. *Deterministic global optimization.* Springer, 2012.

[25] L. Song, R. Zeng, Y. Wang, and M. Shang. Solving bi-objective unconstrained binary quadratic programming problem with multi-objective path relinking algorithm. In *2016 12th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD)*, pages 289–293, 2016.

[26] Satya Tamby and Daniel Vanderpooten. Enumeration of the nondominated set of multiobjective discrete optimization problems. *INFORMS Journal on Computing*, 33(1):72–85, 2021.

[27] L. Trieu. *Continuous Optimization Methods for Convex Mixed-Integer Nonlinear Programming.* PhD thesis, Technischen Universität Dortmund, 2015.

[28] Ozgu Turgut, Evrim Dalkiran, and Alper E Murat. An exact parallel objective space decomposition algorithm for solving multi-objective integer programming problems. *Journal of Global Optimization*, 75(1):35–62, 2019.

[29] P. Van Emde Boas. Another np-complete problem and the complexity of computing short vectors in a lattice. Technical report, Technical report, University of Amsterdam, Department of Mathematics, 1981.

[30] Y. Zhou. A decomposition-based multi-objective tabu search algorithm for triobjective unconstrained binary quadratic programming problem. In *2017 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC)*, volume 1, pages 101–107, 2017.