On the exacteness of the ε-constraint method for bi-objective integer nonlinear programming

Marianna De Santis[*] Daniele Patria[*]

July 27, 2021

Abstract

The ε-constraint method is a well-known scalarization technique used for multiobjective optimization. We explore how to properly define the step size parameter of the method in order to guarantee its exactness when dealing with problems having two nonlinear objective functions and integrality constraints on the variables. Under specific assumptions, we prove that the number of nonlinear integer subproblems that the method needs to address to detect the complete Pareto front is finite. We report numerical results on portfolio optimization instances built on real-world data and compare the ε-constraint method with an existing criterion space algorithm for bi-objective nonlinear integer programming.

Key Words: Multiobjective Optimization, Integer Programming, ε-constraint Algorithm, Critetion Space Algorithms.

Mathematics subject classifications (MSC 2010): 90C10, 90C29

1 Introduction

There is a growing interest in devising exact methods for multi-objective integer programming (MOIP) as it is underlined by recent contributions in this respect (see, e.g. [2, 15, 9, 5, 6, 7]). This is partly due to the fact that MOIPs represent a flexible tool to model real-world applications. Such models appear in works on finance, management, transportation, design of water distribution networks, biology [13, 17, 18, 19, 20]. MOIPs are intrinsically nonconvex, implying that the

[*]Dipartimento di Ingegneria Informatica Automatica e Gestionale, Sapienza Università di Roma, Via Ariosto, 25, 00185 Roma, Italy, (marianna.desantis@uniroma1.it, patria.1743074@studenti.uniroma1.it)

design of exact and efficient solution methods is particularly challenging and requires global optimization techniques [10]. In this paper, we focus on bi-objective nonlinear integer programming problems of the following form

$$\begin{array}{ll} \min & (f_1(x), f_2(x))^T \\ \text{s.t.} & x \in \mathcal{X} \cap \mathbb{Z}^n. \end{array} \tag{BOIP}$$

where $\mathcal{X} \subseteq \mathbb{R}^n$ and $f_1, f_2 : \mathbb{R}^n \to \mathbb{R}$ are continuous functions. The image of the feasible set $\mathcal{X} \cap \mathbb{Z}^n$ under the vector-valued function $f : \mathbb{R}^n \to \mathbb{R}^2$ represents the feasible set in the *criterion space*, or the *image set*. When dealing with problem (BOIP), one wants to detect the so called *efficient solutions* $x^* \in \mathcal{X} \cap \mathbb{Z}^n$. Those are feasible points such that there exists no other feasible point $x \in \mathcal{X} \cap \mathbb{Z}^n$ for which $f_j(x) \leq f_j(x^*)$, $j = 1, 2$ and $f(x) \neq f(x^*)$. The images $f(x)$ of efficient points $x \in \mathcal{X} \cap \mathbb{Z}^n$ are called *non-dominated points*. Furthermore, point $\bar{x} \in \mathcal{X} \cap \mathbb{Z}^n$ is called a *weakly efficient* point of (BOIP) if there is no $x \in \mathcal{X} \cap \mathbb{Z}^n$ with $f(x) < f(\bar{x})$. Solving (BOIP) exactly means to detect its complete set of non-dominated points, also called the *non-dominated set* or *Pareto front* and its complete set of efficient points, also called the *efficient set*. In the following, we will denote the non-dominated set by $\mathcal{Y}_N$.

Regarding general purpose methods able to give correctness guarantees, the focus is most of all on multiobjective integer *linear* problems and we refer to [14] for a survey. A class of algorithms developed is the so called *criterion space search algorithms*, i.e., algorithms that work in the space of the objective functions (see, e.g. [12, 15, 16]) and focus in detecting only the Pareto front of the problem. Such algorithms find non-dominated points by solving a sequence of single-objective integer linear programming problems. After computing a non-dominated point, these algorithms remove the dominated parts of the criterion space (based on the obtained non-dominated point) and look for not-yet-found non-dominated points in the remaining parts.

Criterion space algorithms usually rely on *scalarization* techniques. This means that the multiobjective problem is replaced with a parameter-dependent single-objective integer optimization problem. In order to find several non-dominated solutions, a sequence of single-objective integer optimization problems has to be handled considering different values of the parameters. A typical issue is how to choose the parameters so that the method can guarantee the detection of the complete non-dominated set. Criterion space algorithms have been extended to deal with nonlinear problems, even if this clearly adds difficulties both from a theoretical and a numerical point of view. In case of (BOIPs), an approach that can be followed to deal with nonlinearities is the one proposed in [5], where the Frontier Partitioner Algorithm (FPA), relying on the use of the weighted-sum scalarization, has been proposed.

It is the purpose of this paper to use the ideas developed in [5] to define an exact criterion space algorithm based on another scalarization technique, the ε-constraint method. The ε-constraint method produces single-objective subproblems adding further constraints to the original feasible set. More specifically, given (BOIP), the ε-constraint method minimizes single-objective optimization problems of the following form:

$$\begin{array}{rl} \min & f_2(x) \\ \text{s.t.} & f_1(x) \leq \varepsilon \\ & x \in \mathcal{X} \cap \mathbb{Z}^n. \end{array} \tag{P$_\varepsilon$}$$

The parameter ε varies between $\min\{f_1(x) \mid x \in \mathcal{X} \cap \mathbb{Z}^n\}$ and $f_1(\hat{x}) + \delta$ with $\hat{x} \in \operatorname{argmin}\{f_2(x) \mid x \in \mathcal{X} \cap \mathbb{Z}^n\}$ and δ is a positive step size. As it will be clarified in the next section, the role of the step size δ is crucial to detect the complete non-dominated set of a (BOIP). The role of $f_1(x)$ and $f_2(x)$ in the definition of Problem (P$_\varepsilon$) can be interchanged.

The paper is organized as follows. In Section 2, we recall the γ-positivity assumption introduced in [5] and we establish our main result. Under specific assumptions, we prove that the ε-constraint method is able to detect the complete Pareto front of a (BOIP) after having addressed a finite number of single-objective problems. In Section 3, we report our numerical experience, comparing the performance of FPA* [5] with the ε-constraint method devised on portfolio optimization problems. Section 4 concludes.

2 The γ-positivity assumption and the exacteness of the ε-constraint method

The scheme of the ε-constraint method applied to (BOIP) is reported in Algorithm 1. As already mentioned, at every iteration k of the algorithm, the following single-objective integer subproblem needs to be addressed

$$\begin{array}{rl} \min & f_2(x) \\ \text{s.t.} & f_1(x) \leq \varepsilon^k \\ & x \in \mathcal{X} \cap \mathbb{Z}^n, \end{array} \tag{P$^k_\varepsilon$}$$

with $\varepsilon^k \in \mathbb{R}$ properly set. As a first assumption, we ask the availability of a solver for (P$^k_\varepsilon$).

Assumption 2.1. *There exists an oracle that either returns an optimal solution of* (P$^k_\varepsilon$) *or certifies its infeasibility.*

In the definition of FPA (and FPA*) [5] some basic assumptions on Problem (BOIP) had to be made. Here we make the same assumptions, reported in the following.

Definition 2.2. *The* ideal point *of Problem* (BOIP) *is the vector $f^{id} \in \mathbb{R}^2$ defined component-wise as*

$$f_i^{id} = \min_{\mathcal{X} \cap \mathbb{Z}^n} f_i(x), \quad i = 1, 2. \tag{1}$$

Algorithm 1: Scheme of the ε-constraint method.

```
**Input:** (BOIP), $\mathcal{Y}_N = \emptyset$, $\delta > 0$, k = 0;
**Output:** the Pareto front $\mathcal{Y}_N$ of $(BOIP)$;
**Compute** $x^* \in \operatorname{argmin}_{x \in \mathcal{X} \cap \mathbb{Z}^n} f_1(x)$
**Set** $\varepsilon^0 = f_1(\hat{x})$, where $\hat{x} \in \operatorname{argmin}_{x \in \mathcal{X} \cap \mathbb{Z}^n} f_2(x)$
**while** $\varepsilon^k \geq f_1(x^*)$ **do**
    **Compute** $x^k \in \operatorname{argmin}_{x \in \mathcal{X}^k \cap \mathbb{Z}^n} f_2(x)$, with
     $\mathcal{X}^k = \mathcal{X} \cap \{x \in \mathbb{R}^n : f_1(x) \leq \varepsilon^k\}$
    **Set** $\mathcal{Y}_N = \mathcal{Y}_N \cup \{f(x^k)\}$
    **Set** $\varepsilon^{k+1} = f_1(x^k) - \delta$
    **Set** $k = k + 1$
**end**
**Apply** a filtering procedure to $\mathcal{Y}_N$
**Return** $\mathcal{Y}_N$
```

Assumption 2.3 (Existence of the ideal point)**.** *We assume that the ideal objective values f_i^{id}, $i = 1, 2$ exist.*

The crucial assumption that we make in order to prove the exactness of the ε-constraint method is the so called *positive gap value* assumption. We need to assume that a positive value exists that underestimates the distance between the image of two integer feasible points of (BOIP), componentwise.

Definition 2.4 (Positive γ-function)**.** *A function $g : \mathcal{X} \to \mathbb{R}$ is a positive γ-function over $\mathcal{X} \cap \mathbb{Z}^n$ if there exists a positive $\gamma \in \mathbb{R}$ such that $|g(x) - g(z)| \geq \gamma$, for all $x, z \in \mathcal{X} \cap \mathbb{Z}^n$ with $g(x) \neq g(z)$.*

Assumption 2.5. *The functions $f_i : \mathbb{R}^n \to \mathbb{R}$, $i = 1, 2$ in Problem* (BOIP) *are positive γ-function as in definition 2.4.*

Assumptions 2.3 and 2.5 imply that the non-dominated set $\mathcal{Y}_N$ of (BOIP) is finite (see Proposition 2.7 in [5]). Note that there exist a number of classes of functions that easily satisfy Assumption 2.1 and Assumption 2.5 (see Section 4.3 in [5]), such as linear or quadratic functions defined over $\mathbb{Q}^n$.

Let Assumption 2.5 hold for $f_1(x)$ and $f_2(x)$ with $\gamma > 0$. Let $\delta > 0$ be the input parameter for Algorithm 1. Two different scenarios occur:

- $\delta > \gamma$: in this case the ε-constraint method could miss some points of the Pareto front $\mathcal{Y}_N$, as the step size δ may be wider than the distance between two non-dominated points;

- $\delta \leq \gamma$: the ε-constraint algorithm is able to detect the complete Pareto front $\mathcal{Y}_N$ as shown in the following.

In order to prove Theorem 2.8, we first show that the ε-constraint method can find, at every iteration, a not-yet detected weakly efficient point of (BOIP).

Lemma 2.6. *Let Assumption 2.5 hold with $\gamma > 0$. If* (BOIP) *is not infeasible, the solution of* P^0_ε *of Algorithm 1 is weakly efficient for* (BOIP).

Proof. Let $\hat{x} \in \mathrm{argmin}_{x \in \mathcal{X} \cap \mathbb{Z}^n} f_2(x)$ and let $\varepsilon^0 = f_1(\hat{x})$. Let x^0 be the solution of (P^0_ε), namely $x^0 \in \mathrm{argmin}_{x \in \mathcal{X}^0 \cap \mathbb{Z}^n} f_2(x)$ where $\mathcal{X}^0 = \mathcal{X} \cap \{x \in \mathbb{R}^n : f_1(x) \leq \varepsilon^0\}$. Assume by contradiction that x^0 is not a weakly efficient solution for (BOIP). Then, $\tilde{x} \in \mathcal{X} \cap \mathbb{Z}^n$ exists such that $f_1(\tilde{x}) < f_1(x^0)$ and $f_2(\tilde{x}) < f_2(x^0)$. Then, we get a contradiction with respect to the optimality of x^0, as $\tilde{x}$ would be a feasible point for (P^0_ε) with $f_2(\tilde{x}) < f_2(x^0)$. □

Lemma 2.7. *Let Assumption 2.5 hold with $\gamma > 0$. Assume that $\delta \leq \gamma$ in Algorithm 1 and that the point x^{k-1} detected at iteration $k-1$ is a weakly efficient solution for* (BOIP). *Then, at step k, Algorithm 1 finds a weakly efficient solution $x^k \neq x^{k-1}$ and no other weakly efficient point exists with $f_1(x^k) \leq f_1(x) \leq f_1(x^{k-1}) - \delta$.*

Proof. Let $x^k \in \mathrm{argmin}_{x \in \mathcal{X}^k \cap \mathbb{Z}^n} f_2(x)$ where $\mathcal{X}^k = \mathcal{X} \cap \{x \in \mathbb{R}^n : f_1(x) \leq \varepsilon^k\}$ and with $\varepsilon^k = f_1(x^{k-1}) - \delta$. Assume by contradiction that x^k is not a weakly efficient solution for (BOIP). Then, $\tilde{x} \in \mathcal{X}^k \cap \mathbb{Z}^n$ exists such that $f_1(\tilde{x}) < f_1(x^k)$ and $f_2(\tilde{x}) < f_2(x^k)$. This is a contradiction to x^k being an optimal solution for (P^k_ε), as $\tilde{x}$ would be a feasible point for P^k_ε with $f_2(\tilde{x}) < f_2(x^k)$. The fact that Assumption 2.5 holds and that $\delta \leq \gamma$ implies that $x^k \neq x^{k-1}$ and that no weakly efficient point exists with $f_1(x^k) \leq f_1(x) \leq f_1(x^{k-1}) - \delta$. □

Note that we cannot prove that the solution of (P^k_ε) is efficient, as a point $\tilde{x} \in \mathcal{X}^k \cap \mathbb{Z}^n$ may exist such that $f_1(\tilde{x}) < f_1(x^k)$ and $f_2(\tilde{x}) = f_2(x^k)$. From Lemma 2.6 and Lemma 2.7 we are able to prove the following result

Theorem 2.8. *Let Assumptions 2.5, 2.3 and 2.1 hold. Let $\delta \leq \gamma$. Algorithm 1 finds the complete Pareto front $\mathcal{Y}_N$ of* (BOIP) *after having addressed a finite number of single objective integer programs.*

Proof. According to Lemma 2.6 and Lemma 2.7, at each step of the `while` loop in Algorithm (1), we solve a problem that finds a not-yet detected weakly efficient solution of (BOIP). Thanks to Assumption 2.3, we have that the `while` loop will take at most $m = \lceil f_1(\hat{x}) - f_1(x^*)/\delta \rceil$ iterations. Then, considering the two single-objective integer programs tackled at the beginning of Algorithm 1 for the computation of x^* and $\hat{x}$, the total number of single objective integer programs addressed by Algorithm 1 is $m + 2$. The Pareto front $\mathcal{Y}_N$ of (BOIP) can be detected considering the set of the images of the weakly efficient solutions found and applying a filtering procedure to this set, keeping only the non-dominated solutions. □

3 Numerical results

In our computational experience, we consider bi-objective nonlinear integer instances arising from portfolio selection problems. Let $\mu \in \mathbb{R}^n$ be the expected return and

$Q \in \mathbb{R}^{n\times n}$ be the covariance matrix with respect to a specific set of assets. We consider the following model

$$\begin{array}{ll} \min & (x^TQx,\ -\mu^Tx) \\ \text{s.t.} & a^Tx \leq b \\ & x \geq 0 \\ & x \in \mathbb{Z}^n, \end{array}$$

where the elements of $a \in \mathbb{R}^n$ are the prices of the financial securities, $b \in \mathbb{R}$ is the budget of the investor and the non-negativity constraint rules out short sales. The decision variable $x_i \in \mathbb{Z},\ i = 1, \dots, n$ stands for the amount of unit of a certain asset the investor is buying.

As benchmark data set, we used historical real-data capital market indices from the Eurostoxx50 index that were used in [3, 4] and are publicy available at `https://host.uniroma3.it/docenti/cesarone/DataSets.htm`. This data set was used for solving a *Limited Asset Markowitz (LAM)* model. For each of the 48 stocks the authors obtained 264 weakly price data, adjusted for dividends, from Eurostoxx50 for the period from March 2003 to March 2008. Stocks with more than two consecutive missing values were disregarded. Logarithmic weekly returns, expected returns and covariance matrices were computed based on the period March 2003 to March 2007. By choosing stocks at random from the 48 available ones, we built portfolio optimization instances of different sizes. We decided to generate 60 different instances by considering $n = 5, 10, 25, 30$ stocks. For every n, 15 different instances have been generated. Hence, we got the covariances matrices, the expected returns and the prices for every combination by picking the proper information from the files provided. As in [1], for every instance, we set $b = 10\ \sum_{i=1}^n a_i$, representing the budget of the investor. In order to run FPA* and the exact version of the ε-constraint method, we need a proper value $\gamma > 0$ so that the γ-positivity assumption is satisfied. Therefore, we had to pre-process the data, trimming the number of decimal digits to four and multiplying the entries by 10^3, ending with $\gamma \geq 0.1$. Note that, since the entries of the matrices are in $\mathbb{Q}$, the value γ can be defined as $1/r$, where $r \in \mathbb{N}$ is the least common multiple of the denominators of the rational coefficients (see [5], Proposition 4.14).

Both in the implementation of FPA* and of the ε-constraint method, we considered the linear objective $-\mu^Tx$ as the function defining the additional constraint in the single-objective integer subproblems. Consequently, both FPA* and the ε-constraint method have to deal with a sequence of single-objective convex quadratic integer problems with linear constraints. In our Python implementation of the two algorithms, we used the MIQP solver of GUROBI [11].

All experiments have been executed on an Intel Core i5-6300U CPU running at 2.40 GHz and all running times were measured in cpu seconds.

3.1 Comparison with FPA*

In Table 1 and Table 2 we report, for each instance, the CPU time needed by FPA* and the ε-constraint method to detect the non-dominated set, whose cardinality is reported in the column N. For instances with $n = 5$ variables, we report the results obtained by running FPA, too. Note that FPA and FPA* are proved to detect the full Pareto front after the solution of $2N + 1$ and $N + 2$ integer programs, respectively [5].

As expected, FPA may take more than two times the CPU time needed by FPA*. On the other hand, FPA* and the ε-constraint method have very similar performances, as the number of subproblem addressed by the two algorithm resulted to be very close in practice, even if the theoretical bound on the number of subproblems solved by the ε-constraint method can be much larger than $N + 2$.

Instance	$n = 5$				$n = 10$		
	FPA	FPA*	ε-const	N	FPA*	ε-const	N
p1	58.622	24.987	24.599	7380	238.654	234.754	34760
p2	29.307	6.693	6.551	1464	78.236	76.971	10379
p3	113.066	34.132	32.889	6104	60.345	60.553	6050
p4	6.836	1.887	1.809	401	47.166	46.357	6974
p5	24.248	6.017	5.863	1207	182.296	180.577	31190
p6	11.449	3.094	3.034	708	50.235	49.399	6270
p7	30.845	8.768	8.558	1676	17.275	17.167	2374
p8	13.172	4.609	4.195	845	230.237	220.498	23660
p9	4.771	1.726	1.593	412	102.458	96.957	13546
p10	6.048	2.09	1.949	429	93.249	88.336	8470
p11	25.777	7.798	7.285	1549	103.547	98.216	14985
p12	12.897	4.442	4.125	838	69.709	66.241	7783
p13	137.997	40.044	37.409	7260	57.354	54.447	5057
p14	15.706	5.454	5.057	1258	305.462	291.578	26335
p15	36.565	10.727	9.99	1873	48.465	46.495	4994

Table 1: Results on instances with $n = 5$ and $n = 10$ variables

We further compare FPA* and the ε-constraint algorithm using performance profiles as proposed by Dolan and Moré [8]. Given a set of solvers $\mathcal{S}$ and a set of problems $\mathcal{P}$, the performance of a solver $s \in \mathcal{S}$ on problem $p \in \mathcal{P}$ is compared against the best performance obtained by any solver in $\mathcal{S}$ on the same problem. The performance ratio is defined as $r_{p,s} = t_{p,s}/\min\{t_{p,s'} \mid s' \in \mathcal{S}\}$, where $t_{p,s}$ is the measure we want to compare, and we consider a cumulative distribution function $\rho_s(\tau) = |\{p \in \mathcal{P} \mid r_{p,s} \leq \tau\}|/|\mathcal{P}|$. The performance profile for $s \in S$ is the plot of the function ρ_s. We report in Figure 1 the performance profiles of FPA* and the ε-constraint with respect to the CPU time considering all the 60 instances. Note

Instance	$n = 25$			$n = 30$		
	FPA*	ε-const	N	FPA*	ε-const	N
p1	1300.581	1265.896	63415	10068.922	10137.358	162308
p2	3111.565	3145.79	91306	3684.094	3682.686	56587
p3	974.068	989.094	77020	5381.171	5416.535	83742
p4	543.683	549.171	35837	9712.737	9906.551	132464
p5	807.287	821.02	46343	5226.969	5311.91	86458
p6	737.876	747.852	36424	7284.134	7465.468	109712
p7	1723.602	1733.165	123166	7036.658	7121.332	125773
p8	954.394	948.633	39078	6923.088	6908.466	110547
p9	899.327	884.5	38348	9984.137	9973.19	137072
p10	631.116	619.971	28737	7683.148	7682.36	119352
p11	2326.067	2305.429	84277	7489.533	7480.726	120466
p12	1882.476	1864.43	66071	4452.296	4396.061	68003
p13	2234.691	2201.693	103374	7281.698	7164.956	125586
p14	1551.685	1539.967	87819	4510.21	4500.241	67150
p15	2224.76	2182.219	109878	5657.234	5647.937	85597

Table 2: Results on instances with $n = 25$ and $n = 30$ variables

that the value τ needed to have both $\rho_a(\tau) = 1$ is very small ($\tau = 1.1$), confirming that the two algorithm share very similar performance.

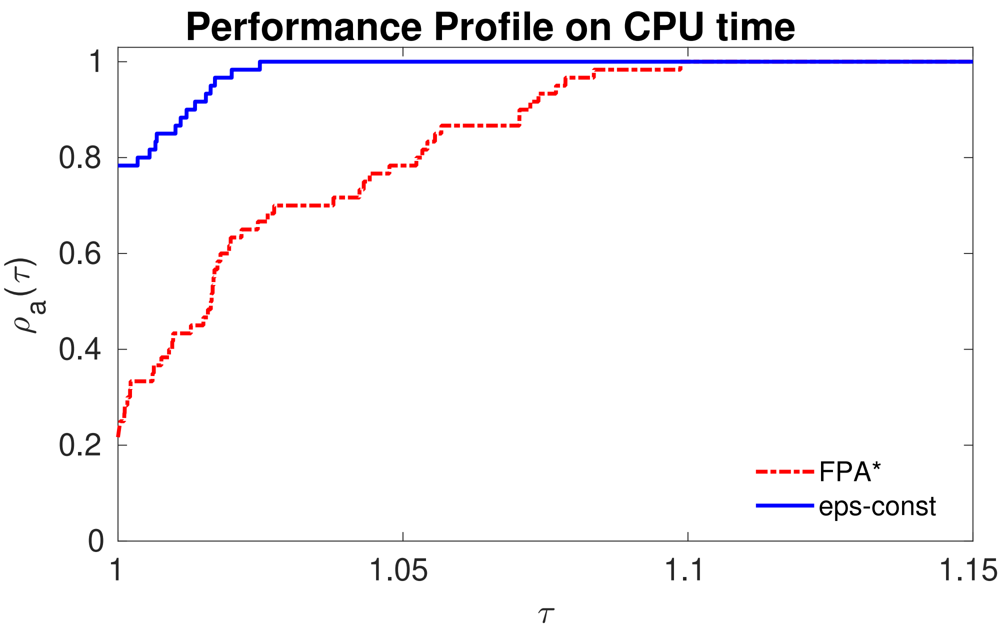

Figure 1: Comparison between FPA* and the ε-constraint method on all the 60 instances.

4 Conclusions

We focus on the definition of the ε-constraint method for bi-objective nonlinear integer programming problems. We give sufficient conditions able to guarantee that the ε-constraint method detects the full Pareto front of a (BOIP) after having addressed a finite number of single-objective integer problems. The method have been numerically compared with an existing criterion space algorithm on real-world portfolio instances, showing very similar performances.

References

[1] Christoph Buchheim, Marianna De Santis, Francesco Rinaldi, and Long Trieu. A frank–wolfe based branch-and-bound algorithm for mean-risk optimization. *Journal of Global Optimization*, 70(3):625–644, 2018.

[2] Guillermo Cabrera-Guerrero, Matthias Ehrgott, Andrew J Mason, and Andrea Raith. Bi-objective optimisation over a set of convex sub-problems. *Annals of Operations Research*, pages 1–26, 2021.

[3] F. Cesarone, A. Scozzari, and F. Tardella. A new method for mean-variance portfolio optimization with cardinality constraints. *Annals of Operations Research*, 205(1):213–234, 2013.

[4] Francesco Cesarone and Fabio Tardella. Equal risk bounding is better than risk parity for portfolio selection. *Journal of Global Optimization*, 68(2):439–461, 2017.

[5] 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.

[6] Marianna De Santis and Gabriele Eichfelder. A decision space algorithm for multiobjective convex quadratic integer optimization. *Computers & Operations Research*, page 105396, 2021.

[7] Marianna De Santis, Gabriele Eichfelder, Julia Niebling, and Stefan Rocktäschel. Solving multiobjective mixed integer convex optimization problems. *SIAM Journal on Optimization*, 30(4):3122–3145, 2020.

[8] Elizabeth D Dolan and Jorge J Moré. Benchmarking optimization software with performance profiles. 91(2):201–213.

[9] Saliha Ferda Doğan, Özlem Karsu, and Firdevs Ulus. An exact algorithm for biobjective integer programming problems. *Computers & Operations Research*, 132:105298, 2021.

[10] Gabriele Eichfelder. Twenty years of continuous multiobjective optimization in the twenty-first century. 2021.

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

[12] 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.

[13] A Legendre, E. Angel, and F. Tahi. Bi-objective integer programming for rna secondary structure prediction with pseudoknots. *BMC Bioinformatics*, 19 (13), 2018.

[14] Anthony Przybylski and Xavier Gandibleux. Multi-objective branch and bound. *European Journal of Operational Research*, 260(3):856–872, 2017.

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

[16] 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.

[17] Aly-Joy Ulusoy, Filippo Pecci, and Ivan Stoianov. Bi-objective design-for-control of water distribution networks with global bounds. *Optimization and Engineering*, pages 1–51, 2021.

[18] Panagiotis Xidonas, George Mavrotas, and John Psarras. Equity portfolio construction and selection using multiobjective mathematical programming. *Journal of Global Optimization*, 47(2):185–209, 2010.

[19] Mehmet Mutlu Yenisey and Betul Yagmahan. Multi-objective permutation flow shop scheduling problem: Literature review, classification and current trends. *Omega*, 45:119–135, 2014.

[20] Han Zhong, Wei Guan, Wenyi Zhang, Shixiong Jiang, and Lingling Fan. A bi-objective integer programming model for partly-restricted flight departure scheduling. *Plos one*, 13(5):e0196146, 2018.