Optimising a nonlinear utility function in multi-objective integer programming

Melih Ozlen, Meral Azizoğlu, Benjamin A. Burton

Abstract

In this paper we develop an algorithm to optimise a nonlinear utility function of multiple objectives over the integer nondominated set. Our approach is based on identifying and updating bounds on the individual objectives as well as the optimal utility value. This is done using already known solutions, linear programming relaxations, utility function inversion, and integer programming. We develop a general optimisation algorithm for use with k objectives, and we illustrate our approach using a tri-objective assignment problem.

Keywords: Multiple objective optimisation, integer programming, nonlinear utility function

1 Introduction

The majority of studies reported in the optimisation literature consider a single objective, such as minimising cost or maximising profit. However, in practice, there are usually many criteria that need to be considered simultaneously. In particular, the increasing effect of globalisation brings safety, environmental impact and sustainability issues, and hence their related performance measures, into consideration. The practical aim is to find solutions that are not only economically profitable but also safe, green and sustainable.

Multi-Objective Integer Programming (MOIP) considers discrete representations by integer variables. The main focus on MOIP in the literature has been on enumerating the entire integer nondominated set, or on optimising a single linear utility function. However, many practical situations require the optimisation of a nonlinear utility function that combines multiple objectives. Prominent applications of such problems include, but are not limited to, pricing, routing, production planning, resource allocation, portfolio selection, capital budgeting, computer networks and reliability networks.

Besides their practical importance, these optimisation problems are theoretically challenging as they—even their simpler versions—fall into the class of NP-hard problems. Despite their practical and theoretical importance, there is no reported research on them, at least to the best of our knowledge. Recognising this gap in the literature, in this paper we address the optimisation of an explicitly defined nonlinear utility function of multiple objectives over the integer nondominated set, and we provide a general framework for its solution.

A MOIP problem defines a feasible nondominated set, and the optimal value of our nonlinear utility function can always be found in this nondominated set. A naïve solution to our optimisation problem could therefore be to generate all nondominated solutions of the MOIP problem, and to evaluate the utility function at each point. However, this naïve approach would be highly impractical, since the number of nondominated solutions can be extremely large in general.

Recognising this, we develop a more sophisticated approach to optimise a nonlinear utility function over the integer nondominated set, in which we generate only a smaller subset of all nondominated solutions. To avoid the generation of non-promising solutions, we compute objective optimality bounds by combining the best solution found so far with the nonlinear utility function. To generate the subset of promising nondominated solutions, we use an algorithm by Özlen and Azizoğlu [2009] that recursively solves MOIP problems with fewer objectives.

The rest of the paper is organised as follows. Section 2 reviews the related literature. In Section 3 we explain this algorithm in full and prove its correctness. Section 4 offers a detailed illustration of the workings of the algorithm, using an instance of a tri-objective assignment problem. We conclude in Section 5.

2 Literature

The best-studied cases of MOIP are Multi-Objective Combinatorial Optimisation (MOCO) problems. These are special cases of MOIP that have special constraint set structures. Ehrgott and Gandibleux [2000, 2004] provide rich surveys of MOCO studies that use exact and approximate approaches, respectively. They address some special problem structures and discuss their solution methodologies. Ehrgott and Gandibleux [2002] survey other MOCO problems including, but not limited to, nonlinear programming, scheduling and multi-level programming. These recent studies have led to a considerable growth in the MOCO literature. However, research on generating the nondominated solution set of MOIP, or on optimisation over that set, is still scarce.

Klein and Hannan [1982], Sylva and Crema [2004], Ehrgott and Ruzika [2008] study both MOIP and Multi-Objective Mixed Integer Programming (MOMIP) models, a generalisation of MOIP. Klein and Hannan [1982] develop an approach based on the sequential solutions of the Single-Objective models. Their algorithm generates a subset, but not necessarily the whole set, of all nondominated solutions. Sylva and Crema [2004] improve the approach of Klein and Hannan [1982] by defining a weighted combination of all objectives, and their approach guarantees to generate all nondominated solutions. Ehrgott and Ruzika [2008] suggest modifications to the ε-constraint method by first including slack variables, and then elasticising the constraints and including surplus variables.

Klamroth et al. [2004], Ehrgott [2006] study the general MOIP problem. Klamroth et al. [2004] discuss the importance of using upper bounds on the objective function values when generating the nondominated set, and define composite functions to obtain such bounds. To form the composite functions, they propose classical optimisation methods such as cutting plane and branch and bound. Ehrgott [2006] presents some properties of the nondominated solutions and proposes a scalarisation technique that can identify some nondominated solutions. Özlen and Azizoğlu [2009] develop a general approach to generate all nondominated solutions of the MOIP problem. Their method recursively identifies objective efficiency ranges using problems with fewer objectives, and is an improvement over the classical ε-constraint method. Dhaenens et al. [2010] present a similar but parallel algorithm for MOCO problems, that again involves recursively solving problems with fewer objectives. Przybylski et al. [2010a]

extend the two phase method—first developed for solving MOCO problems—to generate nondominated solutions for the MOIP problem.

There are few studies dealing with MOIP problems in which the aim is to optimise a function. Przybylski et al. [2010b] develop an algorithm to identify all extreme nondominated solutions that can optimise a linear function of multiple objectives. Abbas and Chaabane [2006], Jorge [2009] deal with optimising a linear function over the efficient set of a MOIP problem. Alves and Clímaco [2007] review some interactive approaches for MOIP and MOMIP problems, where no explicit utility function is defined, and where the decision maker must be consulted on a regular basis.

3 The algorithm

In its general form, our algorithm optimises a nonlinear utility function of k objectives over the integer programming nondominated set. This problem can be defined precisely as:

Min $G(f_1(x), f_2(x), \dots, f_k(x))$
s.t. $x \in X$,
where X is the set of feasible solutions with $x_j \geq 0$ and $x_j \in \mathbb{Z}$ for all $j \in \{1, 2, \dots, n\}$.

The individual objectives are defined as $f_1(x) = \sum_{j=1}^{n} c_{1j}x_j$, $f_2(x) = \sum_{j=1}^{n} c_{2j}x_j$, ..., and $f_k(x) = \sum_{j=1}^{n} c_{kj}x_j$, where $c_{ij} \in \mathbb{Z}$ for all $i \in \{1, 2, \dots, k\}$ and $j \in \{1, 2, \dots, n\}$. The nonlinear utility function[1] $G(f_1(x), f_2(x), \dots, f_k(x))$ is assumed to be strictly increasing in each individual objective function $f_1(x)$, $f_2(x)$, ..., and $f_k(x)$.

A solution $x' \in X$ is called *k-objective efficient* if and only if there is no $x \in X$ such that $f_i(x) \leq f_i(x')$ for each $i \in \{1, 2, \dots, k\}$ and $f_i(x) < f_i(x')$ for at least one i. The resulting objective vector $(f_1(x'), f_2(x'), \dots, f_k(x'))$ is said to be *k-objective nondominated*.

A solution $x' \in X$ is called *k-objective optimal* if and only if there is no $x \in X$ for which $G(f_1(x), f_2(x), \dots, f_k(x)) < G(f_1(x'), f_2(x'), \dots, f_k(x'))$. Because our utility function is strictly increasing, any k-objective optimal solution must also be k-objective nondominated.

Our proposed method of finding a k-objective optimal solution is based on a shrinking set of bounds for the k individual objectives. We update the lower bounds using linear programming relaxations. Where possible, we update the upper bounds by using the lower bounds and inverting the utility function; where necessary, we update the upper bounds using the algorithm of Özlen and Azizoğlu [2009]. The shrinking bounds allow us to avoid non-promising portions of the search space, and the method of updating these bounds is designed to solve integer programs only when absolutely necessary.

Algorithm 1 gives the stepwise description of our procedure to find the optimal solution for the k-objective nonlinear utility function $G(f_1(x), f_2(x), \dots, f_k(x))$. Key variables that we use in this algorithm include:

- G^{BEST}, the best known value of the utility function;
- f_i^{LB} and f_i^{UB}, the current lower and upper bounds on the values of the individual objective functions.

Algorithm 1 terminates with a G^{BEST} value that is the minimum of $G(f_1(x), f_2(x), \dots, f_k(x))$ among all nondominated solutions. Stated formally:

[1] We refer to G as a *utility function* because it combines the multiple objectives $f_1, \dots, f_k$. However, we *minimise* G for consistency with other authors such as Klein and Hannan [1982], Przybylski et al. [2010a].

Algorithm 1 Optimising G over the nondominated set of a MOIP problem

Step 0. Find some initial set of solutions X^K.
Initialise $G^{BEST} = \min_{x \in X^K} G(f_1(x), f_2(x), \ldots, f_k(x))$.
Solve Min $f_i(x)$ s.t $x \in X$ for each $i \in \{1, 2, \ldots, k\}$.
Set each f_i^{LB} to the corresponding optimal objective value.
Set each f_i^{UB} to ∞.

Step 1. If $G(f_1^{LB}, f_2^{LB}, \ldots, f_k^{LB}) \geq G^{BEST}$ then STOP.
For each objective $i \in \{1, 2, \ldots, k\}$, find f_i^A that solves
$G(f_1^{LB}, \ldots, f_{i-1}^{LB}, f_i^A, f_{i+1}^{LB}, \ldots, f_k^{LB}) = G^{BEST}$.
Set $f_i^{UB} = \min(\lfloor f_i^A \rfloor, f_i^{UB})$ for each $i \in \{1, 2, \ldots, k\}$.

Step 2. For each objective $i \in \{1, 2, \ldots, k\}$:
Solve the LP relaxation of
Min $f_i(x)$ s.t. $x \in X$, $f_1(x) \leq f_1^{UB}$, $f_2(x) \leq f_2^{UB}$, …, and $f_k(x) \leq f_k^{UB}$.
Let $f_i^{LP} = f_i(x^*)$ be the optimal objective value, and set $f_i^{LB} = \lceil f_i^{LP} \rceil$.
If x^* is integer and $G(f_1(x^*), f_2(x^*), \ldots, f_k(x^*)) < G^{BEST}$,
set $G^{BEST} = G(f_1(x^*), f_2(x^*), \ldots, f_k(x^*))$.
If the lower bound f_i^{LB} is updated for any objective i then go to Step 1.

Step 3. Use Özlen and Azizoğlu [2009] to update the upper bound f_k^{UB}. Specifically:
Generate all nondominated solutions for the $(k-1)$-objective MOIP problem
Min $f_1(x)$, $f_2(x)$, …, $f_{k-2}(x)$, $f_{k-1}(x) + \varepsilon f_k(x)$
s.t. $x \in X$, $f_1(x) \leq f_1^{UB}$, $f_2(x) \leq f_2^{UB}$, …, and $f_k(x) \leq f_k^{UB}$.
If no feasible solution exists then STOP.
Each time we generate a nondominated solution x^*,
test whether $G(f_1(x^*), f_2(x^*), \ldots, f_k(x^*)) < G^{BEST}$.
If true, set $G^{BEST} = G(f_1(x^*), f_2(x^*), \ldots, f_k(x^*))$ and go to Step 1.
Let ME be the set of all $(k-1)$-objective nondominated solutions.
Set $f_k^{UB} = \max_{x \in ME} f_k(x) - 1$ and go to Step 2.

Theorem 1. *Algorithm 1 finds the minimum value of $G(f_1(x), f_2(x), \ldots, f_k(x))$ among all nondominated solutions to the MOIP problem.*

Proof. As the algorithm runs we maintain the following invariants:

- The utility G^{BEST} is obtainable; that is, $G^{BEST} = G(f_1(x), f_2(x), \ldots, f_k(x))$ for some feasible integer solution $x \in X$.
- Either G^{BEST} is already equal to the optimal utility, or else the optimal utility can be achieved for some solution $x \in X$ with $f_i^{LB} \leq f_i(x) \leq f_i^{UB}$ for each objective $i \in \{1, \ldots, k\}$.

In essence, the lower and upper bounds f_i^{LB} and f_i^{UB} are used to bound the region of the search space that remains to be examined. It is easy to see that these invariants hold:

- In Step 0, each f_i^{UB} is ∞, each f_i^{LB} is the global minimum for $f_i(x)$, and G^{BEST} is obtained from a known solution.

- In Step 1, any solution with $f_i(x) > f_i^A$ must have a utility worse than G^{BEST}. The revised upper bound of $\lfloor f_i^A \rfloor$ is valid because each $f_i(x) \in \mathbb{Z}$.
- In Step 2, the revised lower bounds are valid because each optimal LP value f_i^{LP} is equal to or better than the corresponding optimal IP value.
- In Step 3, any revision to G^{BEST} is valid because it comes from a nondominated solution: in particular, Özlen and Azizoğlu [2009] show that each solution to the $(k-1)$-objective MOIP in Step 3 is k-objective nondominated also. The revision to f_k^{UB} is valid because Özlen and Azizoğlu [2009] show there cannot exist any other nondominated solution having objective function value $f_k(x)$ between $\max_{x \in ME} f_k(x)$ and the previous value of f_k^{UB}.

To prove that the algorithm terminates: Even if no bounds are updated in Steps 1 or 2, the procedure of Özlen and Azizoğlu [2009] will reduce the bound f_k^{UB} in Step 3. This ensures that the bounds shrink during every loop through the algorithm, and because all bounds are integers we must terminate after finitely many steps.

To prove that the algorithm gives the optimal utility: Upon termination, either G^{BEST} is at least as good as anything obtainable within our bounds $f_i^{LB} \le f_i(x) \le f_i^{UB}$ (if we STOP in Step 1 or 2), or else these bounds have been reduced so far that the remaining search space is empty (if we STOP in Step 3). Either way, we know from our invariants that G^{BEST} is obtainable and that no better utility is possible. □

4 An example problem

In this section we illustrate our approach on a concrete example with $k = 3$. This example is an instance of a Tri-Objective Assignment Problem (TOAP) with five assignments, whose data is taken from Özlen and Azizoğlu [2009]. The objective function coefficients are taken from a discrete uniform distribution between 1 and 100, and are tabulated in Table 1.

c_1	1	2	3	4	5
1	99	19	74	55	41
2	23	81	93	39	49
3	66	21	63	24	38
4	65	41	7	39	66
5	93	30	5	4	13

c_2	1	2	3	4	5
1	28	39	19	42	7
2	66	98	49	83	42
3	73	26	42	13	54
4	46	42	28	27	99
5	80	17	99	59	68

c_3	1	2	3	4	5
1	29	67	2	90	7
2	84	37	64	64	87
3	54	11	100	83	61
4	75	63	69	96	3
5	66	99	34	33	21

Tab. 1: Coefficients for the three objectives in the example problem instance

Each solution (i.e., assignment) is represented by a sequence of column indices assigned to rows 1 through 5. For instance, in the solution sequence 2-4-5-1-3, we assign row 1 to column 2 and row 2 to column 4. By optimising single objectives (Step 0 of the algorithm), the initial lower bounds on the individual objectives are:

$$
\begin{aligned}
f_1^{LB} &= 86 \quad \text{from sequence 2-1-4-3-5;} \\
f_2^{LB} &= 128 \quad \text{from sequence 1-5-4-3-2;} \\
f_3^{LB} &= 129 \quad \text{from sequence 3-2-1-5-4.}
\end{aligned}
$$

For this example we set our nonlinear utility function to $G(f_1(x), f_2(x), f_3(x)) = f_1(x)^3 + f_2(x)^3 + f_3(x)^3$. The iterations of Algorithm 1 are summarised in Table 2. Most of the columns in this table show the variables that appear in Algorithm 1; the column G^{LB} shows the value of $G(f_1^{LB}, f_2^{LB}, f_3^{LB})$ as computed in Step 1. A value of $*$ indicates that the value has not changed from the line above.

For the initialisation in Step 0, the procedure solves three IPs and finds three nondominated solutions that provide lower bounds on the individual objective functions. Step 1 then sets (and later updates) upper bounds on the individual objective functions based on the current best solution, G^{BEST}. Step 2 updates the lower bounds by solving the linear programming relaxations with the upper bound constraints on the individual objective function values. Steps 1 and 2 are iterated for as long as they continue to update these lower and upper bounds. When the bounds cannot be updated further, Step 3 generates tri-objective nondominated solutions by generating a bi-objective nondominated set based on the upper bounds f_1^{UB}, f_2^{UB} and f_3^{UB}. If Step 3 is able to update the best utility value G^{BEST}, it returns to Step 1; otherwise it updates f_3^{UB} and returns to Step 2. In the final iteration, where Step 3 fails to find any feasible tri-objective nondominated solutions within the current bounds, the entire algorithm terminates.

We see from Table 2 that our shrinking bounds perform very well for this example: Algorithm 1 requires the solution of just eight IPs to find the optimal utility value. If we were to use the naïve method from the introduction and generate all nondominated solutions, we would require a total of 56 IPs to solve, using the generation algorithm of Özlen and Azizoğlu [2009]. This illustrates the way in which many IPs can be avoided (by eliminating nondominated solutions without explicitly generating them) using the shrinking bound techniques of Algorithm 1.

5 Conclusion

In this study we propose a general algorithm to optimise a nonlinear utility function of multiple objectives over the integer nondominated set. As an alternative to the naïve method of generating and evaluating all nondominated solutions, we restrict our search to a promising subset of nondominated solutions by computing and updating bounds on the individual objectives. The nondominated solutions within this promising subset are generated using the algorithm of Özlen and Azizoğlu [2009]. As illustrated by the example in Section 4, these bounding techniques can significantly reduce the total number of IPs to be solved. Because solving IPs is the most computationally expensive part of the algorithm, we expect these bounding techniques to yield a significant performance benefit for the algorithm as a whole.

For larger problems that remain too difficult to solve, Algorithm 1 can be used as an *approximation algorithm*. We can terminate the algorithm at any time, whereupon G^{BEST} and $G(f_1^{LB}, f_2^{LB}, \ldots, f_k^{LB})$ will give upper and lower bounds for the optimal utility, and we will have a feasible solution $x \in X$ for which $G(f_1(x), f_2(x), \ldots, f_k(x)) = G^{BEST}$.

We hope that this study stimulates future work in the field of multi-objective optimisation. One promising direction for future research may be to apply our algorithm to specific families of MOCO or MOIP problems. The special structure of the constraints in these families might help to improve the efficiency of our algorithm for arbitrary nonlinear utility functions. Another topic for future research might be the development of branch-and-cut techniques for solving the constrained IP models that appear in Algorithm 1. In particular, an IP solved at a

Step	#IP	$f_1(x)$	$f_2(x)$	$f_3(x)$	G^{BEST}	f_1^{LB}	f_2^{LB}	f_3^{LB}	G^{LB}	f_1^{UB}	f_2^{UB}	f_3^{UB}	$\lfloor f_1^A \rfloor$	$\lfloor f_2^A \rfloor$	$\lfloor f_3^A \rfloor$
0	1 IP Min $f_1(x)$	86	214	324	44,448,624	86			636,056						
0	2 IP Min $f_2(x)$	209	128	367	*	*	128		2,733,208						
0	3 IP Min $f_3(x)$	291	348	129	*	*	*	129	4,879,897						
1	*Find* $f_1^A, f_2^{A,} f_3^A$				*	*	*	*	*	342	346	346	342	346	346
2	Solve LR Min $fi(x)$ i=1,2,3	86	130.2	129.1	*	*	131	130	5,081,147	*	*	*			
1	*Find* $f_1^A, f_2^{A,} f_3^A$				*	*	*	*	*	*	*	*	342	346	346
3	4 IP Min $f_1(x) f_2(x)$	86	214	324	*	*	*	*	*	*	*	*			
3	5 IP Min $f_1(x) f_2(x)$	96	186	204	15,809,256	*	*	*	*	*	*	*			
1	*Find* $f_1^A, f_2^{A,} f_3^A$				*	*	*	*	*	224	234	234	224	234	234
2	Solve LR Min $fi(x)$ i=1,2,3	93.5	169.8	157.3	*	94	170	158	9,687,896	*	*	*			
1	*Find* $f_1^A, f_2^{A,} f_3^A$				*	*	*	*	*	190	222	215	190	222	215
2	Solve LR Min $fi(x)$ i=1,2,3	95.1	178.4	167.6	*	96	179	168	11,361,707	*	*	*			
1	*Find* $f_1^A, f_2^{A,} f_3^A$				*	*	*	*	*	174	216	209	174	216	209
2	Solve LR Min $fi(x)$ i=1,2,3	95.6	181.3	173.7	*	*	182	174	12,181,328	*	*	*			
1	*Find* $f_1^A, f_2^{A,} f_3^A$				*	*	*	*	*	165	212	207	165	212	207
2	Solve LR Min $fi(x)$ i=1,2,3	95.8	182.4	177.6	*	*	183	178	12,652,975	*	*	*			
1	*Find* $f_1^A, f_2^{A,} f_3^A$				*	*	*	*	*	159	210	206	159	210	206
2	Solve LR Min $fi(x)$ i=1,2,3	95.8	183.1	179.7	*	*	184	180	12,946,240	*	*	*			
1	*Find* $f_1^A, f_2^{A,} f_3^A$				*	*	*	*	*	155	208	*	155	208	205
2	Solve LR Min $fi(x)$ i=1,2,3	95.9	183.5	181.6	*	*	*	182	13,142,808	*	*	*			
1	*Find* f_1^A, f_2^A				*	*	*	*	*	152	207	*	152	207	
2	Solve LR Min $fi(x)$ i=1,2,3	95.9	183.7	182.6	*	*	*	183	13,242,727	*	*	*			
1	*Find* f_1^A, f_2^A				*	*	*	*	*	151	206	*	151	206	
2	Solve LR Min $fi(x)$ i=1,2,3	95.9	183.7	183.5	*	*	*	184	13,343,744	*	*	*			
1	*Find* f_1^A, f_2^A				*	*	*	*	*	149	205	*	149	205	
2	Solve LR Min $fi(x)$ i=1,2,3	95.9	183.7	184.4	*	*	*	185	13,445,865	*	*	*			
1	*Find* f_1^A, f_2^A				*	*	*	*	*	148	204	*	148	204	
2	Solve LR Min $fi(x)$ i=1,2,3	95.9	183.8	185.3	*	*	*	186	13,549,096	*	*	*			
1	*Find* f_1^A, f_2^A				*	*	*	*	*	146	*	*	146	204	
2	Solve LR Min $fi(x)$ i=2,3		183.9	185.5	*	*	*	*	*	*	*	*			
3	6 IP Min $f_1(x) f_2(x)$	96	186	204	*	*	*	*	*	*	*	203			
3	7 IP Min $f_1(x) f_2(x)$	no more solutions			*	*	*	*	*	*	*	*			
2	Solve LR Min $fi(x)$ i=1,2	97.3	184.6		*	98	185	*	13,707,673	*	*	*			
1	*Find* $f_1^A, f_2^{A,} f_3^A$				*	*	*	*	*	144	203	*	144	203	204
2	Solve LR Min $fi(x)$ i=1,2,3	97.3	184.7	186.4	*	*	*	187	13,812,020	*	*	*			
1	*Find* f_1^A, f_2^A				*	*	*	*	*	143	202	*	143	202	
2	Solve LR Min $fi(x)$ i=1,2,3	97.3	184.7	187.3	*	*	*	188	13,917,489	*	*	*			
1	*Find* f_1^A, f_2^A				*	*	*	*	*	141	201	*	141	201	
2	Solve LR Min $fi(x)$ i=1,2,3	97.3	184.8	188.3	*	*	*	189	14,024,086	*	*	*			
1	*Find* f_1^A, f_2^A				*	*	*	*	*	139	200	*	139	200	
2	Solve LR Min $fi(x)$ i=1,2,3	97.3	184.9	189.2	*	*	*	190	14,131,817	*	*	*			
1	*Find* f_1^A, f_2^A				*	*	*	*	*	137	*	*	137	200	
2	Solve LR Min $fi(x)$ i=2,3		184.9	189.4	*	*	*	*	*	*	*	*			
3	8 IP Min $f_1(x) f_2(x)$	infeasible, STOP			15,809,256	98	185	190	14,131,817	137	200	203			

Tab. 2: Iteration details of Algorithm 1 on the example problem instance

particular iteration might be used as an initial feasible solution for the IP models that appear in subsequent iterations.

Acknowledgements

The third author is supported by the Australian Research Council under the Discovery Projects funding scheme (project DP1094516).

References

Moncef Abbas and Djamal Chaabane. Optimizing a linear function over an integer efficient set. *European J. Oper. Res.*, 174(2):1140–1161, 2006.

Maria João Alves and João Clímaco. A review of interactive methods for multiobjective integer and mixed-integer programming. *European J. Oper. Res.*, 180(1):99–115, 2007.

C. Dhaenens, J. Lemesre, and E. G. Talbi. K-PPM: a new exact method to solve multi-objective combinatorial optimization problems. *European J. Oper. Res.*, 200(1):45–53, 2010.

M. Ehrgott and S. Ruzika. Improved ϵ-constraint method for multiobjective programming. *J. Optim. Theory Appl.*, 138(3):375–396, 2008.

Matthias Ehrgott. A discussion of scalarization techniques for multiple objective integer programming. *Ann. Oper. Res.*, 147:343–360, 2006.

Matthias Ehrgott and Xavier Gandibleux. A survey and annotated bibliography of multiobjective combinatorial optimization. *OR Spektrum*, 22(4):425–460, 2000.

Matthias Ehrgott and Xavier Gandibleux, editors. *Multiple criteria optimization: state of the art annotated bibliographic surveys*. International Series in Operations Research & Management Science, 52. Kluwer Academic Publishers, Boston, MA, 2002.

Matthias Ehrgott and Xavier Gandibleux. Approximative solution methods for multiobjective combinatorial optimization. *Top*, 12(1):1–89, 2004.

Jesús M. Jorge. An algorithm for optimizing a linear function over an integer efficient set. *European J. Oper. Res.*, 195(1):98–103, 2009.

Kathrin Klamroth, Jørgen Tind, and Sibylle Zust. Integer programming duality in multiple objective programming. *J. Global Optim.*, 29(1):1–18, 2004.

Dieter Klein and Edward Hannan. An algorithm for the multiple objective integer linear programming problem. *European J. Oper. Res.*, 9(4):378–385, 1982.

Melih Özlen and Meral Azizoğlu. Multi-objective integer programming: a general approach for generating all non-dominated solutions. *European J. Oper. Res.*, 199(1):25–35, 2009.

Anthony Przybylski, Xavier Gandibleux, and Matthias Ehrgott. A two phase method for multi-objective integer programming and its application to the assignment problem with three objectives. *Discrete Optim.*, 7(3):149–165, 2010a.

Anthony Przybylski, Xavier Gandibleux, and Matthias Ehrgott. A recursive algorithm for finding all nondominated extreme points in the outcome set of a multiobjective integer programme. *INFORMS J. Comput.*, 22(3):371–386, 2010b.

John Sylva and Alejandro Crema. A method for finding the set of non-dominated vectors for multiple objective integer linear programs. *European J. Oper. Res.*, 158(1):46–55, 2004.

Melih Ozlen
School of Mathematical and Geospatial Sciences, RMIT University
GPO Box 2476V, Melbourne VIC 3001, Australia
(melih.ozlen@rmit.edu.au)

Meral Azizoğlu
Department of Industrial Engineering, Middle East Technical University
Ankara 06531, Turkey
(meral@ie.metu.edu.tr)

Benjamin A. Burton
School of Mathematics and Physics, The University of Queensland
Brisbane QLD 4072, Australia
(bab@maths.uq.edu.au)