Optimally Solving Multi-Objective MILP Problems with Part-Wise Continuous Pareto Fronts

Thiago Cantos Lopes, Nadia Brauner and Leandro Magatão

February 5, 2020

Optimally solving multi-objective MILP problems with part-wise continuous Pareto fronts

Thiago Cantos Lopes[1,2], Nadia Brauner[1], Leandro Magatão[2]

[1] Univ. Grenoble Alpes, CNRS, Grenoble INP, G-SCOP, F-38000 Grenoble, France
thiago.cantos-lopes@grenoble-inp.fr, nadia.brauner@grenoble-inp.fr

[2] Universidade Tecnológica Federal do Paraná (UTFPR), CPGEI, 80230-901, Brazil
magatao@utfpr.edu.br

Keywords : *mixed-integer linear programming, multi-objective optimization, part-wise continuous Pareto front*

1 Introduction

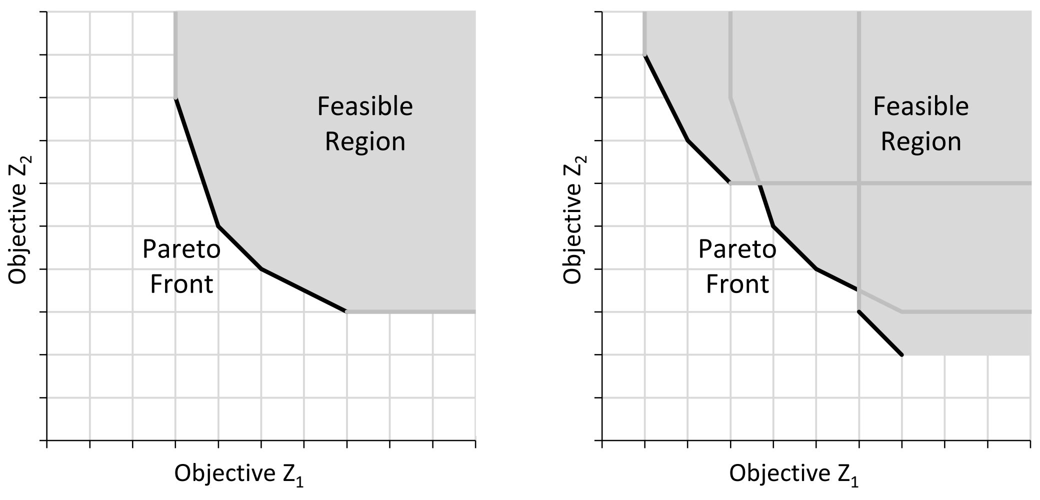
Frequently, in various practical environments, decision makers have to consider multiple objectives at the same time. These are often in conflict, meaning it is not possible to simultaneously reach the best score in all of them. This context defines multi-objective optimization, which commonly employ one of two main strategies: (i) using weight factors for each objective, transforming the problem back into a single objective one; and (ii) obtaining the set of Pareto efficient solutions and allowing the decision maker to choose *a posteriori* [6, 12]. This article focuses on the latter case. For the sake of simplicity, this paper considers a problem definition with two minimization objectives.

Multi-objective problems can relate to integer programing, which implies each integer solution leads to one value for each objective. Furthermore, solution methods are commonly problem-specific [14]. However, in the case of mixed-integer linear programming (MILP) problems, fixing the integer variables and optimizing the continuous ones of a solution can lead to a Pareto front made of continuous segments, as illustrated by Figure 1a. By combining the Pareto fronts of various integer solutions, a part-wise continuous Pareto front is obtained, as depicted by Figure 1b.

A recent work [8] provides an example of the aforementioned multi-objective disputes in the context of assembly line balancing problems [13]. In particular, cycle time (tied to production rate) and line length (tied to implementation costs) display a mixed-integer linear multi-objective dispute, when treated as minimization objectives. A recent work [3] on simultaneous line balancing-sequencing presents a mathematical model that can be easily adapted to test that dispute.

While some methods have been developed to solve multi-objective MILP problems, they are rather limited in terms of the size of problems that are treatable. Recent branch and bound methods [10, 15], for instance, were only tested on instances with up to 70 variables and constraints, which were considered "large instances". More commonly [6, 12], two main approaches are applied: the multiple-weighted sums method and the epsilon-constrained method. The former consists on solving the problem using weighted sums for each objective, gradually changing the weights to reach new solutions in the Pareto front. The later consists in minimizing one objective and using bounds on the others. These limits can then be gradually increased in further executions to generate the Pareto front. These techniques are conceptually illustrated by Figure 2.

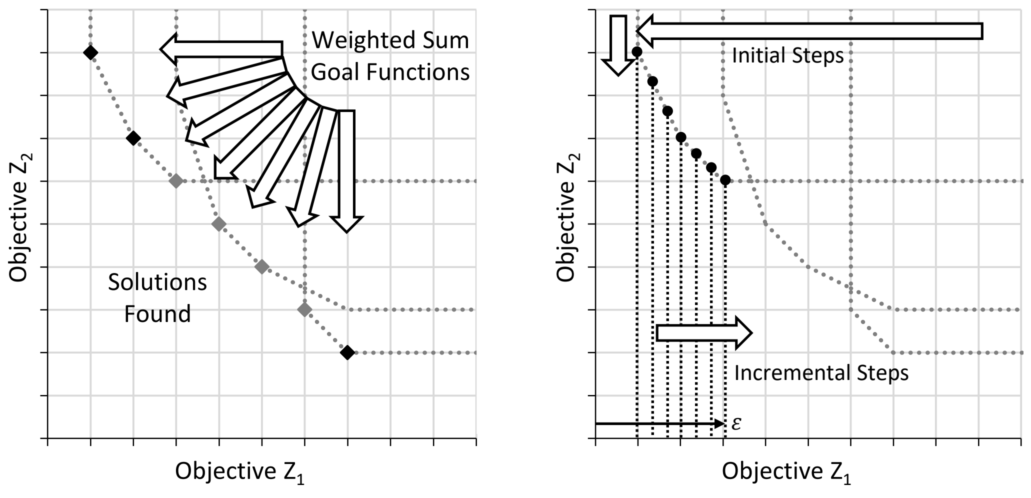
These two methods share the strategy of solving multiple single objective MILP problems to solve a multi-objective one. The inherent advantage of this approach is that the multi-objective problem is tractable whenever multiple executions of the single objective one is.



(a) Single Integer Solution (b) Multiple Integer Solutions

FIG. 1: Combining feasible regions of integer solutions

Given the increasing strength of commercial MILP solvers, it is reasonable to expect that exact solutions are attainable for problems ranging from hundreds to a few thousands variables and constraints. However, Figure 2 illustrates some limitations of these methods: In Figure 2a it is clear that using weighted goal functions only allows some extreme point solutions to be found, for instance, the points highlighted in gray would tend not to be reached by any weighted goal function. Figure 2b, on the other hand, illustrates that the epsilon method might find the same integer solution multiple times, leading to redundancies. The aforementioned capacity to solve MILP instances does not address these limitations, which motivates the development of a new method that overcomes them.



(a) Multiple weighted sums method (b) Epsilon constrained method

FIG. 2: Typical methods based on reduction to single objectives

2 Problem Definition

Consider a mixed-integer linear problem with two minimization objectives Z_1 and Z_2, as stated in Expression (1). Let these goals be bounded by functions of two sets of decision variables $\mathbf{x}$ and $\mathbf{y}$, as presented in Expression (2). Finally, let $\mathbf{x}$ represent integer/binary decisions and $\mathbf{y}$ be the set of continuous decision variables, as stated in Expression (3).

$$\text{Minimize} \quad Z_1 \quad \text{and} \quad Z_2 \tag{1}$$

Subject to:

$$\begin{aligned} Z_1 &\geq \mathbf{f_1}(\mathbf{x}, \mathbf{y}) \\ Z_2 &\geq \mathbf{f_2}(\mathbf{x}, \mathbf{y}) \end{aligned} \tag{2}$$

$$\mathbf{x} \in \mathbb{Z}^+, \ \mathbf{y} \in \mathbb{R}^+ \tag{3}$$

By fixing all the integer variables $\mathbf{x}$ to a specific set $\overline{\mathbf{x}}$, an integer solution is defined. Notice that the residual problem defined by the $\mathbf{y}$ variables is linear, meaning its feasible region is convex. By defining a plane such that the goals Z_1 and Z_2 correspond to its axis, it is possible to project integer solutions to it, defining convex feasible regions as illustrated by Figure 1a: an area bounded by Pareto-efficient continuous solution segments and lines for the minimum possible values of each objective. Naturally, by combining multiple integer solutions, a global part-wise continuous Pareto front is defined for the studied problem (Figure 1b).

3 Proposed Method

In order to define the optimal Pareto front for this problem, a method composed of two parts is presented hereafter: First, it is necessary to extract the full continuous set of solutions for a given integer solution. This is achieved using a duality-based method described in Section 3.1. Second, a procedure for generating an initial global Pareto front and then refining it to optimality is required to determine which set of integer solutions is Pareto efficient. This is done by exploiting a second MILP-based model, and is described in Section 3.2. In these sections, an overline is used to distinguish variables/goal functions from reference values they can be set to. For instance, $\mathbf{x}$ represents the variables, and $\overline{\mathbf{x}}$ refers to values they can be set to.

3.1 Extracting the Pareto front of each Integer Solution

Consider a given integer solution $\mathbf{x} = \overline{\mathbf{x}}$. The residual linear problem ties the goals Z_1 and Z_2 to the set of continuous variables $\mathbf{y}$. An iterative three-step procedure is employed to extract the Pareto-efficient solution segments:

1. Determine the solution wise minimum-value of Z_2, set $\overline{Z}_2$ as this value (Z_2^*);
2. Determine the minimum value of Z_1, using $\overline{Z}_2$ as a limit value of Z_2, name the dual variable tied to this resource constraint π, let $\overline{\pi}$ state its value;
3. If $\overline{\pi} < 0$, determine the maximal value of Z_2 for which the shadow-price $\overline{\pi}$ remains valid, update $\overline{Z}_2$ to that value. Return to Step 2.

Similarly to other works [5, 11] on multi-objective problems, this method exploits duality theory to explore the feasible region of the residual linear problem. However, contrary for instance to reference [4], the analysis is based on the objective space (Z_1-Z_2 plane) rather than the decision space (defined by the $\mathbf{y}$ variables).

The first step is the most straightforward, and consists in simply using Z_2 as the only goal function and solving the linear problem to optimality. The optimal value of the goal function in this step is used to define a resource constraint, as stated by Expression (4).

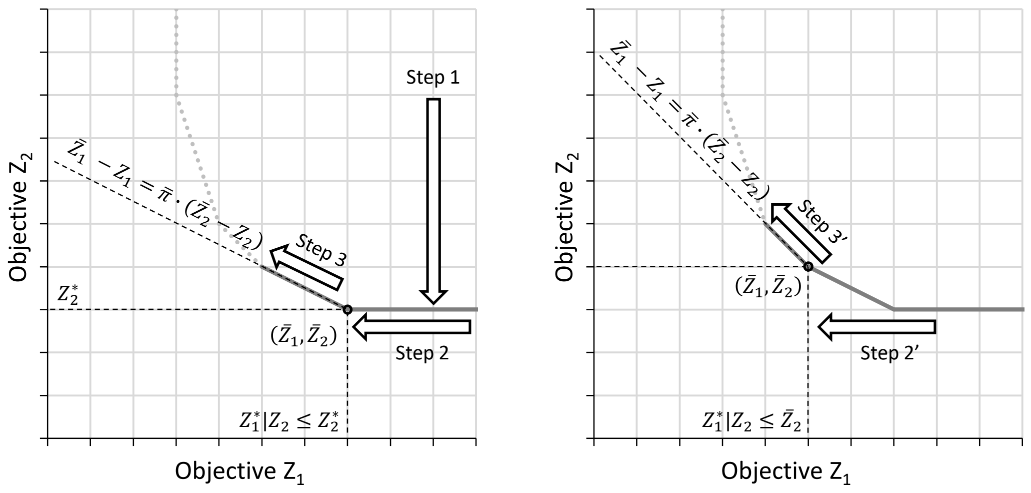
$$Z_2 \leq \overline{Z}_2 + \varepsilon \qquad \pi = \text{dual variable tied to constraint} \tag{4}$$

In the second step, a model that incorporates this resource constraint is solved, and its objective is set to minimize Z_1. The dual variable π stores the shadow price of $\overline{Z}_2$ as a resource, i.e. what is the relative reduction of Z_1 enabled by a marginal increase of $\overline{Z}_2$. A small value (ε) can be added to the RHS (Right Hand Side) of this constraint in order to ensure that its slack is zero and that a valid shadow price π is generated. Adequately defining this ε value may require problem-specific knowledge. The minimal value of Z_1 obtained in this step defines the first Pareto efficient point for the integer solution, namely ($\overline{Z}_1$, $\overline{Z}_2$). The shadow-price π can be combined with that information to define the line that contains this point and its associated Pareto efficient segment, as stated by Equation (5).

$$\overline{Z}_1 - Z_1 = \overline{\pi} \cdot (\overline{Z}_2 - Z_2) \tag{5}$$

Assuming a non-zero shadow price, in the third step, Z_1 is minimized again. This time, Equation (5) is incorporated by the base model rather than Expression (4). The resulting value of Z_2 is then used to update $\overline{Z}_2$ and iterate step 2.

This process is illustrated conceptually by Figure 3: the binary solution depicted by Figure 1a is considered. Figure 3a illustrates the first iteration of the three steps, and Figure 3b illustrates the second iteration. After a third one, $\overline{\pi}$ would be set to zero, meaning the vertical line is reached and no further reduction of Z_1 is possible.



(a) First Iteration (b) Following Iterations

FIG. 3: Generating the feasible region of an integer solution

3.2 Defining and refining the global Pareto front

The previous section detailed how to extract the Pareto front associated to an integer solution $\mathbf{x} = \overline{\mathbf{x}}$. In order to determine the set of efficient $\overline{\mathbf{x}}$ solutions, the following procedure is followed:

1. Solve the problem using one objective function at a time;

2. Combine their solutions into an initial Pareto front, use artificial segments if needed;

3. For each segment in the pareto front, solve the associated maximization problem. This will either:

i) Confirm the optimality of the segment; or

ii) Provide an integer solution that improves the incumbent Pareto front. In this case it should be updated accordingly.

Repeat step 3 until all segments in the incumbent Pareto front are optimal.

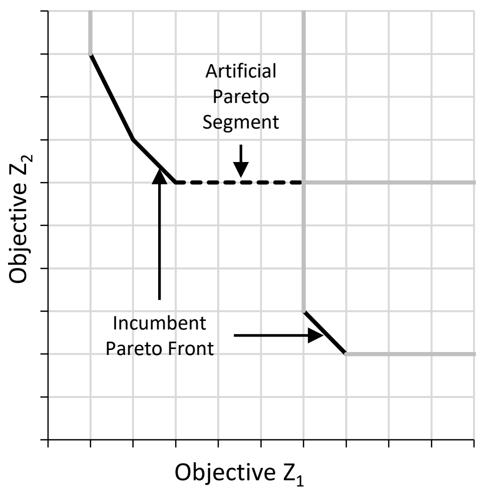
The first step determines the global minimal values of Z_1 and Z_2. It is possible, however, that their combined Pareto fronts do not cover all possible ranges of Z_1. Figure 4a illustrates how to tie them using artificial Pareto segments such that all values of Z_1 are covered for Step 2. This initial front covers all potentially efficient values of Z_1, but is approximative and it is not guaranteed to contain solutions of the problem's true Pareto front.

For the third step, for each segment in the incumbent Pareto set, a maximization model is defined: Let Δ be a continuous variable that measures how much bellow the tested segment a given solution is. This variable is added to the original problem, and set as the maximization objective. Constraints (6) and (7) are also added. In Expression (6), Z_1^- and Z_1^+ are the minimal and maximal values for Z_1 in the tested segment. By solving the resulting model, an optimality test is defined for the line segment: if the resulting maximum value of Δ is zero, then there is no solution bellow the line segment and it is a Pareto efficient. Otherwise, a new integer solution is found and guaranteed to be Pareto efficient for at least one value of Z_1.

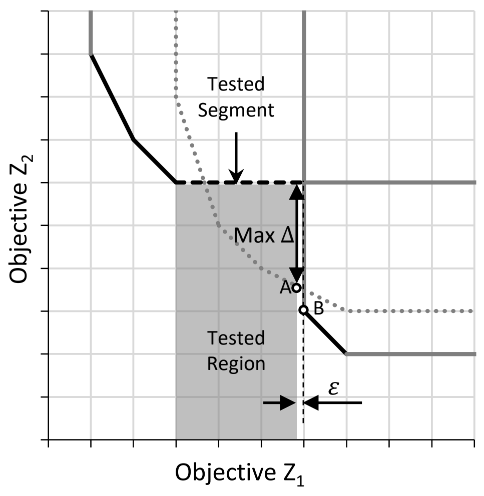
$$Z_1^- \leq Z_1 \leq Z_1^+ - \varepsilon \tag{6}$$

$$\overline{Z}_1 - Z_1 = \overline{\pi} \cdot (\overline{Z}_2 - (Z_2 + \Delta)) \tag{7}$$

Figure 4b illustrates such optimality test for a line segment. This highlights two particularities: First, in the case of an artificial segment, Constraint (7) can be replaced by: $Z_2 + \Delta = \overline{Z}_2$. Second, a small value ε must be subtracted from Z_1^+ in Constraint (6) so that the model does not consider as feasible a solution that belongs to the next line segment. That is, so that the best solution found under the tested region is the one marked with an "A" rather than the one marked with a "B" in Figure 4b. Similarly to in Constraint (4), adequately defining the ε value may require problem specific knowledge.



(a) Initial incumbent Pareto front



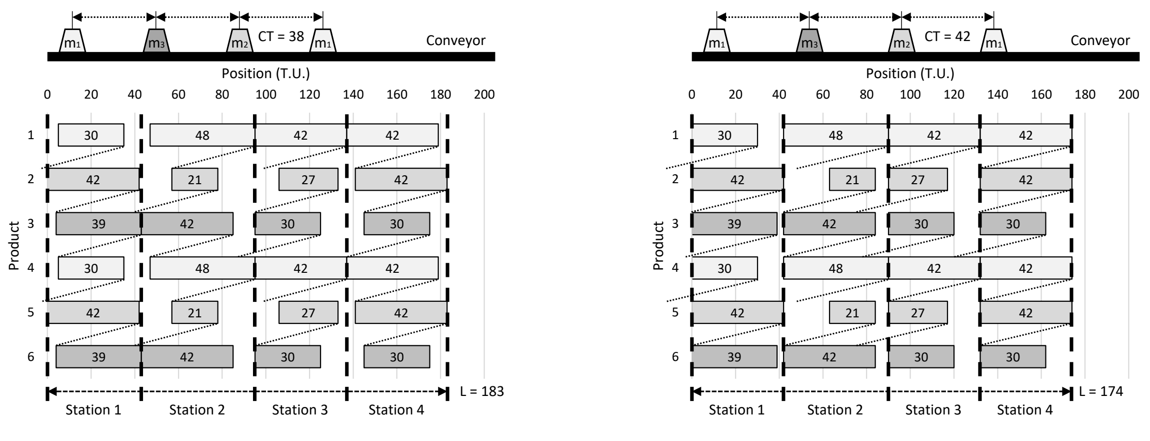
(b) Delta test for new solutions

FIG. 4: Determining the optimal Pareto front

4 Experiments

In order to test the proposed method, six instances of the aforementioned balancing-sequencing problem [8, 13] were defined based on literature datasets [7] and on a recently introduced model [3]. The recently described [8] multi-objective dispute between cycle time (CT) and line length (L) is explored as these are set as minimization objectives ($Z_1 = CT$, $Z_2 = L$). This problem is briefly explained hereafter, so that the reason for the existence of the dispute is introduced by the reader. In a mixed-model assembly line [1, 2], multiple products are manufactured, and the tasks required to produce them are split between sequential stations. This problem combines two key binary (variable set $\mathbf{x}$) decisions: (i) balancing, i.e. assignment of tasks to stations, and (ii) sequencing, i.e. defining an order for product models to enter the line. These two degrees of freedom combined define the duration and order of processing time "blocks" at each station. In the studied formulation [3], the sequence is cyclical and the line is paced, i.e. a conveyor constantly moves products forward while tasks are performed. One of the minimization objectives is the cycle time (CT), which equals the time between two consecutive product launches, and the other is line length (L), i.e. the sum of the lengths of stations, which are not allowed to overlap. These two objectives are, however, affected by continuous scheduling variables (variable set $\mathbf{y}$).

Figure 5 presents two solutions for a fixed balancing-sequencing solution ($\mathbf{x} = \overline{\mathbf{x}}$): the vertical dashed lines represent station boundaries, each horizontal line represents a product being manufactured, the numbers within blocks represent their processing time, the blocks' colors represent the product model, and the diagonal dotted lines represent the worker's upstream movement after finishing work on a product. Notice that the size and sequence of processing time blocks is the same in both Figure 5a and 5b.



(a) Schedule for $CT = 38$ and $L = 183$ (b) Schedule for $CT = 42$ and $L = 174$

FIG. 5: Different schedules for the same balancing-sequencing solution.

Figure 5 illustrates why this mixed-integer multi-objective dispute exists for this problem: two different cyclical scheduling solutions are presented for the same balancing-sequencing decision variables. Therefore, the observed differences for the minimization objectives (CT and L) are necessarily due to the problem's continuous decision variables ($\mathbf{y}$), namely the starting position of each block and the boundaries of each station. This means that each solution for the integer variables ($\mathbf{x}$) defines a residual linear problem for the remaining continuous ones ($\mathbf{y}$), as illustrated by Figure 1a.

Experiments were run on an Intel i7-3610QM CPU with 8GB RAM, and MILP models were solved using Gurobi 8.1. Instance parameters were varied so that instances of multiple sizes (S, M and L for small, medium, and large) were defined. Furthermore, two levels of flexibility were allowed to the problems (1 and 2), with the higher level meaning that the search field is broader in regard to the binary decision variables, i.e. more feasible integer solutions exist.

These instances were solved to optimality by the proposed method, i.e. their true Pareto front was fully determined. Table 1 lists the six instances' parameters, namely number of constraints (Ncon), total variables (Nvar) and binary variables (Nbin).

Instance information				Solution information				
I	Ncon	Nvar	Nbin	Time	π Time	Δ Time	Δ Count	Δ T/C
L-1	673	494	205	300	0.5	285	33	8.6
L-2	673	494	205	260	0.3	195	10	19.5
M-1	537	390	165	11.1	0.1	9.1	7	1.3
M-2	537	390	165	62.4	0.2	49.2	10	4.9
S-1	401	286	125	9.2	0.1	7.5	7	1.1
S-2	401	286	125	16.6	0.1	13.5	8	1.7

TAB. 1: Tests on balancing-sequencing instances

Table 1 presents the results of applying the proposed method to the instances, in particular the total time required to solve them (generate the full Pareto front). This time is mainly consumed by solving linear and mixed-integer linear problems associated to the instance, being mainly divided in three components: time required to generate the Pareto front for given fixed integer variables (π Time - Section 3.1), time required to test incumbent Pareto segments (Δ Time - Section 3.2), with the remaining time required to solve the single objective instances required to generate the initial Pareto front. Lastly, Table 1 lists the number of tested Pareto segments (Δ Count) as well as the average time required to solve the mixed-integer problems associated to these segments (Δ T/C).

As expected, most of the required time is spent solving mixed-integer linear problems rather than linear ones. Furthermore, larger instances require more computing time to be solved. However, the additional flexibility did not always imply more total computing cost: while S-2 and M-2 required more time than S-1 and M-1, the reverse happened between instances L-2 and L-1 (260 vs 300). This can be explained by the number of Pareto segments that required testing for each instance: L-2 required more time per test in average than L-1 (19.5 vs 8.6), however, L-1 required more segment tests than L-2 (33 vs 10).

5 Conclusions and Perspectives

The newly introduced method allows to optimally solve multi-objective mixed-integer linear programing problems with part-wise continuous Pareto fronts. This is achieved by solving a finite number of single objective linear and mixed-integer linear problems. The former are employed to obtain the Pareto front of the residual problem defined by fixing the integer variables $\mathbf{x}$, and the later is used to verify whether incumbent segments in the objective space are part of the true Pareto front. By doing so, the proposed method converts the exact solution of this class of multi-objective MILP problems to multiple, but finitely many, solutions single-objective MILP instances of similar size.

This method addresses some of the major drawbacks of common alternatives that are based on reductions to single objective problems: the weighted sum and ε-constrained methods. The former tends not to find all efficient solutions and the latter might find the same solution multiple times without being able to guarantee that the Pareto front has been fully defined. Assuming the instance is tractable, the proposed method is guaranteed to require a finite number of single-objective MILP model solutions to find the full Pareto front. Further works should compare in more detail the proposed method's performance to that of previously described strategies, as well as evaluate the specific contexts [9] in which it performs best.

Furthermore, the proposed methodology can be generalized for problems with more objectives: the method described in Section 3.1 can be broaden to define Pareto fronts composed of (hyper)-planes for each integer solution; and so can the optimality test described in Section 3.2, by replacing its Δ measure by a normal vector to the tested (hyper)-plane and its ε constraint by distances to the (hyper)-planes that bound the region for which the tested plane is currently being evaluated in terms of Pareto efficiency. Lastly, a general adequate definition for the ε values for Constraints (4) and (6) can be challenging and should be addressed by further works to render the proposed method less dependent on problem specific knowledge.

Acknowledgments

The authors thank the financial support from CNPq (Grants 406507/2016-3 and 307211/2017-7). This study was also financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

References

[1] Battaïa, O., and Dolgui, A. A taxonomy of line balancing problems and their solution approaches. *International Journal of Production Economics 142*, 2 (2013), 259–277.

[2] Boysen, N., Fliedner, M., and Scholl, A. Assembly line balancing: Which model to use when? *International Journal of Production Economics 111*, 2 (2008), 509–528.

[3] Defersha, F. M., and Mohebalizadehgashti, F. Simultaneous balancing, sequencing, and workstation planning for a mixed model manual assembly line using hybrid genetic algorithm. *Computers and Industrial Engineering 119* (2018), 370–387.

[4] Ecker, J. G., and Kouada, I. A. Finding all efficient extreme points for multiple objective linear programs. *Mathematical Programming 14* (1978), 249–261.

[5] Ignizio, J. P. An Algorithm for Solving the Linear Goal Programming Problem by Solving its Dual. *Journal of Operational Research Society 36*, 6 (1985), 507–515.

[6] Keller, A. A. *Multi-Objective Optimization in Theory and Practice: I. Classical Methods.* No. I, Classical methods in Multi-objective Optimization in Theory and Practice. Bentham Science Publishers Ltd, 2017.

[7] Lopes, T. C., Michels, A. S., Sikora, C. G. S., Molina, R. G., and Magatão, L. Balancing and cyclically sequencing synchronous, asynchronous, and hybrid unpaced assembly lines. *International Journal of Production Economics 203* (2018), 216–224.

[8] Lopes, T. C., Pastre, G. V., Michels, A. S., and Magatão, L. Flexible multi-manned assembly line balancing problem: Model, heuristic procedure, and lower bounds for line length minimization. *Omega* (2019).

[9] Mavrotas, G. Effective implementation of the e-constraint method in Multi-Objective Mathematical Programming problems. *Applied Mathematics and Computation 213*, 2 (2009), 455–465.

[10] Mavrotas, G., and Diakoulaki, D. Multi-criteria branch and bound: A vector maximization algorithm for Mixed 0-1 Multiple Objective Linear Programming. *Applied Mathematics and Computation 171* (2005), 53–71.

[11] Pourkarimi, L., and Zarepisheh, M. A dual-based algorithm for solving lexicographic multiple objective programs. *European Journal of Operational Research 176* (2007), 1348–1356.

[12] Rangaiah, G. P. *Multi-Objective Optimization Techniques And Applications In Chemical Engineering.* No. 1 in Advances in Process Systems Engineering. World Scientific, 2009.

[13] Scholl, A. *Balancing and sequencing assembly lines*, 2nd ed. Physica, Heidelberg, 1999.

[14] Ungulu, E. L., and Teghem, J. Multi-objective Combinatorial Optimization Problems : A Survey. *Journal of Multi-Criteria Decision Analysis 3* (1994), 83–104.

[15] Vincent, T., Seipp, F., Ruzika, S., Przybylski, A., and Gandibleux, X. Multiple objective branch and bound for mixed 0-1 linear programming: Corrections and improvements for the biobjective case. *Computers and Operation Research 40*, 1 (2013), 498–509.