Computers and Operations Research 93 (2018) 27–40

Contents lists available at ScienceDirect

Computers and Operations Research

journal homepage: www.elsevier.com/locate/cor

The robust (minmax regret) assembly line worker assignment and balancing problem

Jordi Pereira

Department of Engineering and Sciences, Universidad Adolfo Ibáñez, Av. Padre Hurtado 750, Office C216, Viña del Mar, Chile

ARTICLE INFO

Article history:
Received 18 March 2017
Revised 12 December 2017
Accepted 11 January 2018
Available online 12 January 2018

Keywords:
Production
Line balancing
Robust optimization
Minmax regret

ABSTRACT

Line balancing aims to assign the assembly tasks to the stations that compose the assembly line. A recent body of literature has been devoted to heterogeneity in the assembly process introduced by different workers. In such an environment, task times depend on the worker performing the operation and the problem aims at assigning tasks and workers to stations in order to maximize the throughput of the line. In this work, we consider an interval data version of the assembly line worker assignment and balancing problem (ALWABP) in which it is assumed that lower and upper bounds for the task times are known, and the objective is to find an assignment of tasks and workers to the workstations such that the absolute maximum regret among all of the possible scenarios is minimized. The relationship with other interval data minmax regret (IDMR) problems is investigated, the inapplicability of previous approximation methods is studied, regret evaluation is considered, and exact and heuristic solution methods are proposed and analyzed. The results of the proposed methods are compared in a computational experiment, showing the applicability of the method and the theoretical results to solve the problem under study. Additionally, these results are not only applicable to the problem in hand, but also to a more general class of problems.

1. Introduction

An assembly line is a manufacturing system widely used in the production of homogeneous, mass-produced goods. It consists of several production units, known as stations or workstations, which are linked by a transportation device that moves unfinished products and components from one station to the next. Each station has an assigned worker with an identical allotted time, known as cycle time, to perform the elementary operations (the tasks) assigned to the station. Moreover, these tasks show precedence relations that force order constraints in their respective station assignments, that is, some tasks have to be performed in the same or a preceding station (e.g., the seats of a car must be assembled before assembling the doors).

Balancing an assembly line aims at maximizing the efficiency (minimizing the unproductive time) of an assignment of tasks to stations. The basic version of the problem is known as the SALBP (Simple Assembly Line Balancing Problem) and only considers the above objective and constraints (Scholl and Becker, 2006; Sewell and Jacobson, 2012), while alternative formulations that take into account other issues or have different objectives are collectively known as GALBP (General Assembly Line Balancing Problem), see Battaïa and Dolgui (2013) for a review.

GALBP formulations usually build upon the SALBP by incorporating additional characteristics to the SALBP model. The basic (deterministic) version of the problem studied in this paper builds upon the SALBP by incorporating heterogeneity among workers as observed in sheltered work centers for the disabled (SWD) (Miralles et al., 2007). The problem, known as the assembly line worker assignment and balancing problem (ALWABP), differs from the SALBP by considering that the processing time of the tasks depends on the worker who performs the task. Assuming a fixed workforce, as in a SWD, the number of stations is a parameter and maximizing the efficiency corresponds to minimizing the cycle time allotted to each of the stations. This objective is commonly known as type-2 in the line balancing literature, in contrast to type-1 problems, in which the cycle time (work throughput) is a parameter and efficiency maximization is achieved by minimizing the number of stations.

While the heterogeneity among workers reflects part of the task time variability that the SALBP model does not capture, the ALWABP does not take into account variability in the processing times of each task between different cycles, see Becker and Scholl (2006) for further details. This work builds upon the ALWABP including uncertainty within the task times to reflect this

E-mail address: jorge.pereira@uai.cl

https://doi.org/10.1016/j.cor.2018.01.009
0305-0548/

additional source of heterogeneity, see Moreira et al. (2015b,a) for further discussion. The introduction of uncertainty within the formulation requires a change in the optimization objective, which will now aim at finding an assignment of tasks and workers to stations such that the solution is robust, that is, it offers a good performance under different task time scenarios, see Ben-Tal et al. (2009). We now proceed to review the literature on worker heterogeneity and robustness within assembly line balancing.

1.1. Related work

Line balancing with heterogeneity among workers has been an active research topic in recent years. Some early work considered the problem of choosing and assigning the best set of robots (workers) to each workstation among a group of available options (Levitin et al., 2006; Rubinovitz and Bukchin, 1993). This problem, known as the robotic assembly line balancing problem (RALBP), portrays events in which it is possible to configure the machinery available to each station in accordance to the tasks assigned to it.

While in the RALBP workforce (or machinery) has to be selected, in the assembly line and worker assignment balancing problem (ALWABP) the workforce is fixed to some initial conditions. The ALWABP was first formulated in Miralles et al. (2007), and requires the assignment of both tasks and workers to the workstations in order to maximize line throughput.

The literature for the ALWABP covers different solution approaches. Several heuristic and exact solutions procedures have been proposed. Among the heuristic procedures, we highlight the branch-and-bound-based heuristic described in Miralles et al. (2008); several genetic algorithms (GA), see Gao et al. (2009), Moreira et al. (2012) and Mutlu et al. (2013); two Beam Search (BS) based methods, see Blum and Miralles (2011) and Borba and Ritt (2014); and, more recently, a two-phase variable neighborhood search, see Polat et al. (2016). Among the exact solutions methods, two alternative branch-and-bound procedures (Borba and Ritt, 2014; Vilà and Pereira, 2014) have shown to constitute the state of the art solution method. The procedure described in Borba and Ritt (2014) is a task oriented branching scheme in which a branching decision constitutes the assignment of a task to one worker, while the procedure described in Vilà and Pereira (2014) is a station-oriented branching scheme in which a branching decision constitutes the assignment of tasks and a worker to a station. Even if based on very different approaches, no procedure dominates the other when their performance is compared using the benchmark set of instances and, combining their results, they are able to solve all of the instances from the benchmark set (Blum and Miralles, 2011).

An alternative formulation to line balancing with heterogeneity among workers is the assembly line and worker integration balancing problem (ALWIBP), see Moreira et al. (2015b). In the ALWIBP both regular and special (disabled) workers are available, and the objective is to find an assignment of workers to stations such that all special workers are assigned and the number of regular workers is minimized.

Previous work on robustness issues within assembly line balancing is reviewed in the following lines. Several works explore robustness issues on the SALBP by considering the stability of the solutions obtained by other resolution methods, see Gurevsky et al. (2012) and Sotskov et al. (2006). In Sotskov et al. (2006), the authors solve the type-1 SALBP and investigate the robustness of the solutions by determining the maximum variation of processing times that ensures that a solution remains optimal. In Gurevsky et al. (2012), the stability of the solutions obtained by means of a heuristic method is investigated.

A similar approach is followed in Dolgui and Kovalev (2012), in which the robustness, minmax solution, of type-2 SALBP solutions is considered in a scenario-based situation, and the computational complexity of some specific cases is analyzed.

Two classical optimization approaches to deal with uncertainty are stochastic optimization and robust optimization, both of which have been previously used within the assembly line balancing literature. In stochastic optimization, Cakir et al. (2011), Gamberini et al. (2009) and Sarin et al. (1999), task times are considered to follow statistical distributions which are known in advance. Any realization of task times leads to a scenario, an instance of the deterministic problem, and the stochastic problem tries to optimize the expected outcome of the solution. In robust optimization, the task times are considered to be uncertain but to belong to a given interval without any statistical assumption. The objective is then to find a solution that performs reasonably well for any possible scenario.

Among the alternative robust optimization approaches, the assembly line balancing literature covers several applications of the robust Bertsimas and Sim framework, see Bertsimas and Sim (2003). In this approach a solution is sought under the assumption that a maximum of Γ parameters are allowed to change their baseline, nominal, values. This framework has been investigated within the line balancing literature in a number of papers (Hazir and Dolgui, 2013, 2015; Moreira et al., 2015a; Pereira and Álvarez-Miranda, 2017). In Hazir and Dolgui (2013) the type-2 SALBP is studied and a Benders' decomposition procedure is put forward. The same authors also studied the use of the same framework and solution methodology for a U-shaped type-2 assembly line balancing problem, see Hazir and Dolgui (2015). For line balancing problems with heterogeneity among workers, Moreira et al. (2015a) provides two alternative formulations for type-1 heterogeneous line balancing problems and a fast heuristic for their resolution.

Robust (Bertsimas and Sim) formulations of both the ALWABP and ALWIBP have also been considered in the literature, see Moreira et al. (2015a). This work provides formulations for both the ALWABP and the ALWIBP and proposes an insertion-based heuristic used to solve these problems. The quality of the proposed solution method is then compared to those of a general-purpose mixed integer programming solver.

Yet, the present work differs from the Bertsimas and Sim approach by considering a robustness framework known in the literature as interval data minmax regret (IDMR) (Aissi et al., 2009). In the IDMR framework some parameters are uncertain but bounded to take values within an interval. Instead of considering a number of parameters that may differ from their nominal values, in the IDMR framework all parameters may take any value from the interval, and the problem is to find a solution that performs reasonably well for all the possible scenarios. Specifically, the problem aims at finding a solution minimizing the maximum absolute deviation (regret) between the objective value of the solution and the optimal objective value under any possible scenario (Conde, 2017). The usefulness of minmax regret formulations for line balancing was pointed out in Dolgui and Kovalev (2012, p. 1963). IDMR formulations have been considered to be as pessimistic (Hazir and Dolgui, 2013, p. 262) when compared to other methods. Nonetheless, the framework provides a robust solution with limited information, and these solutions are expected to perform well under many different scenarios.

IDMR problems have been thoroughly studied in the literature, but to the best of the author's knowledge they have never been considered within an assembly line balancing problem. Early results on absolute minmax regret objectives can be found in Aissi et al. (2009), Averbakh and Lebedev (2004) and Kasperski and Zieliński (2006), and a review on the applica-

tion and motivation of both minmax and minmax regret objectives to different combinatorial optimization problems can be found in Aissi et al. (2009). IDMR problems have been studied for multiple combinatorial optimization problems, such as the shortest path (Assunção et al., 2017), quadratic assignment problem (Feizollahi and Averbakh, 2014), the knapsack problem (Furini et al., 2015), the traveling salesman problem (Montemanni et al., 2007), the assignment problem (Pereira and Averbakh, 2011), or the set covering problem (Assunção et al., 2017; Pereira and Averbakh, 2013), among others.

1.2. Contributions of this work

In this work, we consider the IDMR version of the ALWABP-2, which will be denoted by MMR-ALWABP (min max regret assembly line and worker assignment balancing problem). We introduce the problem and consider the applicability of theoretical results for IDMR problems found in the literature. We prove that the 2-approximation algorithm and the evaluation method given in the literature for some absolute IDMR problems are not applicable to the MMR-ALWABP. Consequently, a new regret evaluation method is proposed and new and adapted solution methods are presented.

The results show that a novel heuristic is a viable solution method and that medium-sized instances, with up to 50 tasks, can be optimally solved within reasonable running times. In addition to the solution methods provided for the ALWABP, the applicability of the methods to other IDMR problems is also discussed in the conclusions section.

1.3. Paper outline

The remainder of the paper is structured as follows. Section 2 describes the ALWABP-2 in detail and provides a mathematical formulation of the problem. Section 3 introduces the IDMR framework, discusses the implications of the IDMR framework for the ALWABP, provides the MMR-ALWABP formulation and puts forward the main theoretical results of this work, that is, an alternative method to evaluate regret and a proof that the midpoint-scenario heuristic does not provide an approximation algorithm for the problem. Section 4 describes in detail the proposed solution methods. The results of a computational experiment are reported in Section 5, and some final conclusions and directions of work are given in Section 6.

2. Problem description

In this section, we consider the deterministic version of the ALWABP. A concise description a formulation of the deterministic problem is put forward.

The ALWABP-2 can be formally stated as follows: we are given a set T of elementary tasks (operations) in which the assembly of a product has been divided, a set W of workers, and a set S of stations to perform these tasks. Some of the workers cannot perform some tasks because they lack the specific skill or have some disability. Let $W_t \subseteq W$ be the subset of workers that can perform task t, and let $T_w \subseteq T$ be the subset of tasks that can be performed by worker $w \in W$. For each task t that can be performed by worker w, p_{wt} denotes the time required by the worker to perform the task (indistinctly referred to as the processing time, the task time or the operation time of the task). Some tasks feature precedence constraints among them, that is, they must be performed before other tasks. These precedence constraints can be represented using a directed graph $G(T, A)$, in which an arc (t, t') states that task t has to be performed before task t'. Note that the set of workers and stations is considered to have the same cardinality: $|W| = |S|$. The objective is to find an assignment of tasks and workers to the stations such that 1) the worker assigned to each station can perform all of the tasks assigned to the station, 2) precedence constraints among tasks as fulfilled and 3) the cycle time, i.e. the maximum of the sum of processing times among stations, is minimized.

Among the different available formulations for the problem, we introduce the two-index formulation given in Borba and Ritt (2014), since it is later used in the exact resolution method for the MMR-ALWABP proposed in Section 4. This formulation uses a set of binary variables x_{wt} to identify whether worker $w \in W$ performs task $t \in T$.

Moreover, the formulation also uses an auxiliary real variable C to represent the cycle time, and a set of binary variables d_{vw} that take value 1 if worker $v \in W$ is assigned to an earlier station than worker $w \in W$. This formulation was selected over previous ones (Miralles et al., 2007), because it is more compact and requires fewer variables, see discussion in Borba and Ritt (2014).

According to the above notation, Eqs. (1)–(9) constitute a valid model for the ALWABP-2.

$$\text{minimize } C, \tag{1}$$

subject to:

$$\sum_{t \in A_w} p_{wt} x_{wt} \leq C \qquad \forall w \in W, \tag{2}$$

$$\sum_{w \in A_t} x_{wt} = 1 \qquad \forall t \in T, \tag{3}$$

$$d_{vw} \geq x_{vt} + x_{wt'} - 1 \qquad \forall (t, t') \in A,\ v \in W_t,\ w \in W_{t'} \setminus \{v\}, \tag{4}$$

$$d_{uw} \geq d_{uv} + d_{vw} - 1 \qquad \forall \{u, v, w\} \subseteq W, |\{u, v, w\}| = 3, \tag{5}$$

$$d_{vw} + d_{wv} = 1 \qquad \forall v \in W,\ w \in W \setminus \{v\}, \tag{6}$$

$$d_{vw} \in \{0, 1\} \qquad \forall v \in W,\ w \in W \setminus \{v\}, \tag{7}$$

$$C \in \mathbb{R}^+. \tag{8}$$

$$x_{wt} \in \{0, 1\} \qquad \forall w \in W, t \in T_w, \tag{9}$$

Eq. (1) states that the objective is to minimize the cycle time. Constraint set (2) calculates the cycle time. Constraint set (3) ensures that each task is assigned to a station, while constraint sets (4)–(6) ensure precedence constraints as well as a valid assignment of workers to stations. Consequently, the model does not directly take into account the assignment of workers to stations, but constraint sets (4)–(6) guarantee that such an assignment is possible (that is, workers can always be ordered linearly while complying with the precedence constraints). Constraint set (4) transfers precedence constraints from tasks to their assigned workers. Constraint sets (5) and (6) enforce transitivity and avoid symmetry of worker dependencies. Specifically, constraint set (5) ensures that if worker u precedes worker v and worker v precedes worker w, worker u precedes worker w, while constraint set (6) ensures that each worker precedes or follows each other worker. Finally, constraint sets (7)–(9) define the domain of the variables.

Formulation (1)–(9) highlights the relationship between the ALWABP and other classical combinatorial optimization problems. For example, if all of the workers have identical task times ($p_{1t} = p_{wt}$, $w \in W,\ t \in T$) and all of the tasks can be performed by each worker ($A_t = W,\ t \in T$ and $A_w = T,\ w \in W$), the problem becomes a SALBP-2, a simple ALBP with type-2 objective function. If precedence

constraints are disregarded, constraint set (4)–(6), then the problem corresponds to an unrelated parallel machine scheduling problem (Martello et al., 1997). If workers have identical task times and precedence constraints are disregarded, then the problem becomes a bin-packing problem. Notice that all of these problems are known to be NP-hard (Garey and Johnson, 1979). Moreover, the model is versatile as additional characteristics of real-life assembly lines can be incorporated to it, see Battaïa and Dolgui (2013) for a thorough analysis of different characteristics found in several assembly line environments.

Throughout the paper, we refer to the set of feasible solutions $\mathcal{X}$ (an assignment of tasks to workers) in the following compact form

$$\mathcal{X} = \{\mathbf{x} \in \{0,1\}^{|W|\times|T|} \mid \text{satisfies } (2)-(8)\}. \tag{10}$$

Notice that a feasible solution $\mathbf{x}$ does not explicitly consider the assignment of workers to stations, but if $\mathbf{x}$ fulfills the constraints, a valid assignment exists, and it can be retrieved according to the relative ordering provided by the auxiliary d_{vw} variables.

3. The robust (minmax regret) assembly line and worker assignment line balancing problem

3.1. Formulation of the MMR-ALWABP and regret evaluation

The ALWABP model described in Section 2 considers that the processing times of the tasks are deterministic and known. A more realistic model would consider task time variability intrinsic to manual operations. In the IDMR framework, task times of each task $t \in T$ and worker $w \in W$ are modeled as intervals $= [p^-_{tw}, p^+_{tw}]$ and no information or assumption is made about how it is distributed.

Note that the optimal solution, the one that minimizes the cycle time, depends on the scenario, a realization of processing times for each task and worker. Due to the uncertainty, the objective becomes one of obtaining a solution that remains favorable for all of the possible scenarios, and in the IDMR framework this is achieved by considering the absolute maximum regret of any solution. For a given scenario, the regret of a solution is defined as the difference between the cycle time of the solution and the optimal cycle time for the said scenario, and we aim to minimize the absolute regret in the most adverse scenario.

Aiming at a solution that remains favorable even in adverse scenarios, makes the solution robust against different situations. Given that the proposed robust model is based on a high degree of uncertainty regarding the conditions of the workers, a minmax regret approach is likely to provide a satisfactory solution until additional information is available.

In order to formally introduce the minmax regret version of the problem, some additional notation is required. Let P be the set of all possible scenarios, i.e., a realization of processing times for each task and worker. Any scenario in which either $p_{tw} = p^-_{tw}$ or $p_{tw} = p^+_{tw}$ holds for every task ($t \in T$) and worker ($w \in W$) is referred to as an extreme scenario. We denote the set of extreme scenarios by $P^\xi \subseteq P$. The value of regret for a feasible solution $\mathbf{x} \in \mathcal{X}$ under scenario f is denoted by $c(\mathbf{x}, f)$, and $\mathbf{x}^*_f$ denotes the optimal assignment of tasks to workstations under scenario $f \in P$. Then, the regret of $\mathbf{x}$ under scenario f can be computed as (11).

$$c(\mathbf{x}, f) - c(\mathbf{x}^*_f, f) \tag{11}$$

The scenario f that provides the maximum value of regret for any given solution $\mathbf{x}$ is known as its worst-case scenario, and the optimal solution of its worst-case scenario is known as the worst-case alternative. Hence, the objective of the MMR-ALWABP is to find an assignment of tasks to workers $\mathbf{x} \in \mathcal{X}$ that minimizes the regret for its worst-case scenario, see (12).

$$\min_{\mathbf{x}\in\mathcal{X}} \left\{ \max_{f\in P} \{c(\mathbf{x}, f) - c(\mathbf{x}^*_f, f)\} \right\} \tag{12}$$

Note that the ALWABP-2 can be seen as a special case of the MMR-ALWABP with intervals $p^+_{tw} = p^-_{tw} = p_{tw}$. In other cases, the problem is not easier than the discrete version, because evaluating an MMR-ALWABP requires to find the optimal solution $\mathbf{x}^*_f \in \mathcal{X}$ for at least one scenario (its worst-case scenario).

A classical result which provides a useful method to evaluate the maximum regret of a solution for a large class of combinatorial optimization problems follows:

Lemma 1. *(see Aissi et al., 2009; Kasperski, 2008; Kasperski and Zieliński, 2006; Pereira and Averbakh, 2011, among others) Consider a problem defined as:*

$$\begin{cases} \min \sum_{i=1}^n c_i x_i & c_i \in \mathbb{N} \\ \mathbf{x} \in \mathcal{X} \subset \{0,1\}^n. \end{cases} \tag{13}$$

The regret of a solution $\mathbf{x} \in \mathcal{X}$ of problem (13) in which each coefficient c_i can take any value in the interval $[c_i^-, c_i^+]$ is maximized for scenario $\bar{c}(\mathbf{x})$ defined as:

$$\bar{c}_i(\mathbf{x}) = \begin{cases} c_i^+ & \text{if } x_i = 1 \\ c_i^- & \text{if } x_i = 0 \end{cases} \qquad i = 1, \ldots, n$$

Proof. See (Aissi et al., 2009, Proposition 7). □

The results from Lemma 1 apply to many problems in which uncertainty corresponds to the cost coefficients of binary variables, like in the shortest path, the spanning tree, the assignment, or the traveling salesman problem, among others. In such a case, the worst-case scenario of a solution corresponds to a scenario in which the costs of all variables included in the solution are set to their maximum while the costs of the remaining variables are set to their minimum. Since the costs of the worst-case scenario are known, evaluating (12) for any given solution $\mathbf{x}$ reduces to optimally solving a discrete version of the problem under the costs given by Lemma 1.

The above result does not hold for the MMR-ALWABP, as its objective function, see Eq. (1), corresponds to an auxiliary variable, and the uncertainty corresponds to coefficients used to calculate this auxiliary variable. Therefore, an alternative method to evaluate regret is required. The method builds upon the observation that extreme scenarios are regret maximizers.

Lemma 2. *Let $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$, be two solutions. There is an extreme scenario $f \in P$ that maximizes the difference between their objective values, $c(\mathbf{x}, f) - c(\mathbf{x}', f)$.*

Proof. Let $\Omega, \Omega' \subseteq W$ be the sets of workers that, respectively, define $c(\mathbf{x}, f)$ and $c(\mathbf{x}', f)$. In other words, the set of workers that satisfy the cycle time constraint (2) in equality. For each task $t \in T$, let ω and ω' be the workers that performs task t in solutions $\mathbf{x}$ and $\mathbf{x}'$, respectively.

Note that the task times associated to the worker performing the task in $\mathbf{x}$ may contribute to its objective value only if $\omega \in \Omega$ (for $\mathbf{x}'$, $\omega' \in \Omega'$ must hold). Hence, if $\omega \notin \Omega$ (similarly, $\omega' \notin \Omega'$), we can safely set its task times to an extreme value without affecting the evaluation (we assume that $p_{\omega t}$ is set to $p^-_{\omega t}$, similarly we set $p_{\omega' t}$ to $p^-_{\omega' t}$).

We have to consider the cases in which either $\omega \in \Omega$, $\omega' \in \Omega'$ or both hold.

If $\omega \in \Omega$ and $\omega' \notin \Omega'$, $p_{\omega t}$ contributes to $c(\mathbf{x}, f)$ and setting its value to $p^+_{\omega t}$ maximizes the difference by minimizing $c(\mathbf{x}, f)$. Similarly, if $\omega \notin \Omega$ and $\omega' \in \Omega'$ $p^-_{\omega' t}$, setting $p_{\omega' t}$ to $p^-_{\omega' t}$ maximizes the difference by minimizing $c(\mathbf{x}', f)$.

If $\omega \in \Omega$ and $\omega \in \Omega'$ we distinguish two cases:

1. if $\omega = \omega'$, then the value of the task time is irrelevant as it is added to and subtracted from $c(\mathbf{x}, f) - c(\mathbf{x}', f)$ (we will later assume that its value is set to $p_{\omega t} = p^+_{\omega t}$).
2. if $\omega \neq \omega'$ then the value of $c(\mathbf{x}, f)$ reaches its maximum when $p_{\omega t} = p^+_{\omega t}$, while the minimum of $c(\mathbf{x}', f)$ corresponds to $p_{\omega' t} = p^-_{\omega' t}$, which can be independently optimized. □

Notice that Lemma 2 does not state which worker (or workers) becomes critical, that is the worker (or workers) that defines the cycle time of the solution, and bases its logic on the existence of such a worker (or workers) for each solution $\mathbf{x}$ and $\mathbf{x}'$. As a corollary of Lemma 2, Corollary 1 states that the worst-case scenario of any solution can be found within the subset of extreme scenarios.

Corollary 1. *For any given solution $\mathbf{x} \in \mathcal{X}$, there exists a maximum regret scenario corresponding to an extreme scenario.*

Proof. Lemma 2 must hold for any pair of solutions $\mathbf{x}$, $\mathbf{x}'$ and specifically for $\mathbf{x}$ and its worst-case alternative, $\mathbf{x}^*_f$. □

According to Lemma 2, we can narrow the set of scenarios down to the subset P^ξ of extreme scenarios. Moreover, notice that unlike P, P^ξ is a finite set, and thus the optimal solution, $c(\mathbf{x}^*_f, f)$, for each scenario in P^ξ can be calculated beforehand. Consequently, objective (14) as well as constraint sets (15)–(16) provide a valid formulation for the MMR-ALWABP.

$$\text{minimize } \theta, \tag{14}$$

subject to:

$$\theta \geq c(\mathbf{x}, f) - c(\mathbf{x}^*_f, f) \qquad \forall f \in P^\xi, \tag{15}$$

$$\mathbf{x} \in \mathcal{X} \tag{16}$$

The formulation is amenable to cutting plane approaches by limiting the attention to a subset of the scenarios from P^ξ, and including additional scenarios when required. This approach is usually used in the literature to optimally solve MMR problems (Furini et al., 2015; Montemanni et al., 2007; Pereira and Averbakh, 2011, among others).

In order to avoid enumerating the complete set of scenarios, the cutting plane approach tries to identify the worst-case scenario of incumbent solutions and adds them to P^ξ. Note that for the MMR-ALWABP, the worst-case scenario cannot be ascertained using Lemma 1. Therefore, an alternative approach is required, and a new result, Lemma 3, is used.

Lemma 3. *For any given solution $\mathbf{x} \in \mathcal{X}$, a scenario in which $p_{tw} = p^+_{tw}$ for every task t assigned to worker w, and $p_{tw} = p^-_{tw}$ for every other task and worker, is the maximum regret scenario.*

Proof. A similar argumentation to the one expounded for Lemma 2 is given. First notice that c($\mathbf{x}$, f) corresponds to (17).

$$c(\mathbf{x}, f) = max_{w \in W} \sum_{t \in A_w} p^f_{tw} x_{wt} \tag{17}$$

Substitute (17) for $\mathbf{x}$ and $\mathbf{x}^*_f$ in (12). Notice that the inner maximization in (12) is maximized when $p^f_{tw} = p^+_{tw}$ if they correspond to the evaluation of $\mathbf{x}$, and maximized when $p^f_{tw} = p^-_{tw}$ if they correspond to the evaluation of $\mathbf{x}^*_f$. In case it belongs to the evaluation of both solutions, according to Lemma 2, we can safely choose p^+_{tw} without affecting the total regret and we can set all the processing times that do not participate in the calculation of c($\mathbf{x}$, f) and $c(\mathbf{x}^*_f, f)$ to p^-_{tw}. □

Note that Lemma 3 does not identify the worker creating the scenario that maximizes the regret, it only states that given the worker, the condition holds. As a critical worker must exist, Lemma 3 ensures that an extreme scenario in which $p_{tw} = p^+_{tw}$ for every task t assigned to the critical worker ω in the solution $\mathbf{x}$, and $p_{tw} = p^-_{tw}$ for every other task worker pair, is the worst-case scenario.

Corollary 2. *To evaluate the maximum regret of any solution $\mathbf{x}$, $|W|$ ALWABP-2 instances need to be solved.*

Proof. Suppose now that worker ω is the worker who maximizes the regret. If ω is known, the value of c($\mathbf{x}^*$, f) is the optimal solution of the ALWABP-2 instance in which the tasks assigned to worker ω have task times equal to their upper limit $p^+_{t\omega}$ and the rest of the pairs task-worker have task times equal to their lower limit p^-_{wt}, as shown in Lemma 3. Since ω is not known, all possible workers are to be tested, and their corresponding ALWABP-2 problem needs to be solved. □

Note that the results from Lemma 3 and Corollary 2 differ from the classical results given in Lemma 1, in which the evaluation of the regret is more straightforward, since every solution defines only one scenario that needs to be optimality solved to obtain the maximum regret. The evaluation of the regret for the MMR-ALWABP is more elaborated, as highlighted by Corollary 2. While Lemma 1 holds for most of the previously studied IDMR problems, the literature also covers a few cases in which Lemma 1 does not hold, see Mausser and Laguna (1999) and Pereira (2016b).

3.2. Applicability of the midpoint scenario as an approximation

Before proceeding to the description of the proposed solution methods, we consider the classical midpoint scenario heuristic, which provides a 2-approximation, that is, it ensures that its cost is at most twice the cost of the optimal solution, for several MMR combinatorial optimization problems, including all of the problems covered by Lemma 1, see Aissi et al. (2009), Kasperski and Zieliński (2006), Pereira (2016b), among others.

A scenario-based heuristic builds upon the optimal solution of a specific discrete version of the IDMR problem in which each uncertain parameter is set to a specific value within its interval. For instance, in the midpoint scenario heuristic each parameter is set to the medium value of its interval. For the ALWABP-2, the task times of each task t and worker w are set to $p_{tw} = \frac{p^+_{tw} + p^-_{tw}}{2}$ and the corresponding ALWABP-2 is solved.

Lemma 4. *The midpoint scenario does not provide a 2-approximation for the MMR-ALWABP.*

Proof. Consider the following instance with 3 tasks, $T = \{t_1, t_2, t_3\}$, and 2 workers $W = \{w_1, w_2\}$. The operation time interval of task t_1 and worker w_1 is $p_{t_1,w_1} = [1, 3]$. The remaining operation times are deterministic and equal to 1, that is, the remaining operation time intervals are $p_{t_1,w_2} = p_{t_2,w_1} = p_{t_2,w_2} = p_{t_3,w_1} = p_{t_3,w_2} = [1, 1]$.

It is straightforward to show that the minimum cycle time for any possible realization of task times is no smaller than 2 (one worker is to perform two tasks and the minimum task time of each task is 1).

Among the optimal solutions to the midpoint scenario, one corresponds to worker w_1 performing task t_1 and worker w_2 performing tasks t_2 and t_3 (under the midpoint scenario, this solution has cycle time 2). The worst-case scenario of the above solution corresponds to a realization of operation times equal to $p_{t_1,w_1} = 3$. In this case, the cycle time is 3, while the optimal solution has cycle time 2 (worker w_1 performs tasks t_2 and t_3, worker w_2 performs task t_1), hence the regret of the solution is 1.

Now consider a feasible solution in which worker w_1 performs tasks t_2 and t_3 and worker w_2 performs task t_1. This solution has regret 0 for each possible realization of task times.

As there exists a solution with regret 0, and the midpoint scenario is not able to provide such a solution, it is not a 2-approximation for the MMR-ALWABP. □

Note that the midpoint scenario heuristic still provides a valid solution but Lemma 4 highlights that no approximation guarantee can be made regarding its quality. Also note that a scenario-based heuristic requires the resolution of an ALWABP-2 instance, an NP-hard problem by itself. Consequently, the relevance of scenario based heuristics for the MMR-ALWABP is limited by the instance sizes in which it is possible to solve the ALWABP-2 efficiently.

4. Solution methods for the MMR-ALWABP

Section 3 illustrates the differences between the problem under consideration and previously considered IDMR problems in the literature. These differences highlight the need to develop specific methods to efficiently tackle the MMR-ALWABP. This section introduces the algorithmic components used by the exact method and by the heuristic method. Section 4.1 provides a succinct description of the method used to solve the ALWABP-2, since regret evaluation requires to optimally solve multiple ALWABP-2 instances. Section 4.2 describes the exact mixed integer programming based solution method, while Section 4.3 introduces the modification of the exact method which leads to the proposed heuristic approach.

4.1. ALWABP-2 exact solution method

As Lemma 4 states, the regret evaluation of any given MMR-ALWABP solution requires the optimal solution of $|W|$ ALWABP-2 instances. Due to the computational complexity of the ALWABP-2, a special purpose code is used to obtain these optimal solutions. Among the available methods, the branch, bound and remember algorithm proposed in Vilà and Pereira (2014) is used. For completion purposes, a short description including two enhancements is given below, but the interested reader is referred to Vilà and Pereira (2014) for a detailed description.

The method is based on the resolution of type-1 ALWABP instances with decreasing values of cycle time until no feasible solution for the given set of workers is found, and thus the minimum feasible cycle time has been found. Therefore, the method is iterative in nature and solves multiple type-1 line balancing problems (which are considered as feasibility problems) to solve a type-2 problem. Note that iterative methods have been widely used for other type-2 line balancing problems, see Blum and Miralles (2011) and Scholl and Becker (2006).

Starting from an upper bound on the feasible cycle time and for each tentative (one time unit smaller) cycle time, the algorithm performs a station-oriented enumeration procedure, following the branch, bound and remember scheme proposed in Sewell and Jacobson (2012). The search space is structured as a directed acyclic graph in which the nodes correspond to partial solutions and the arcs correspond to branching decisions. To reduce the number of considered partial solutions, variable prefixing rules are applied whenever a task can only be assigned to a specific worker.

When a node of the search tree is selected for exploration, the branching scheme applies the preprocessing step, and then generates all of its descending nodes. Each branch corresponds to the assignment of a worker and a set of tasks to the first (lowest numbered) empty station while fulfilling precedence and cycle time constraints.

The exploration graph is constructed using a hybrid between a depth-first and a best-first search, see Sewell and Jacobson (2012), in which the next node to explore is selected from different depth levels of the tree in a cyclic fashion. Among the unexplored nodes of a given depth, the most-promising node (according to a heuristic measure that evaluates the likelihood to generate a feasible solution with the desired number of stations) is selected.

Explored and unexplored nodes of the graph are stored in memory using their equivalent state in a dynamic programming formulation. For any given partial solution s, its state representation corresponds to a set of assigned tasks $T(s)$ and workers $W(s)$. Note that the state representation enables the algorithm to avoid exploring equivalent (in terms of their state representation) partial solutions as only one equivalent partial solution is stored and explored. The complete state space is stored, resulting in a memory-intensive procedure.

After a partial solution is generated, some lower bounds are used to verify whether the unassigned tasks can be feasibly assigned to the unassigned workers. If the lower bound reports a minimum number of stations larger than the number of unassigned workers, the partial solution can be pruned from the exploration graph. The following four families of bounds (each based on a different relaxation) are used:

- Bin packing problem (BPP) lower bounds (Blum and Miralles, 2011; Miralles et al., 2008). The bound is derived by discarding precedence constraints among tasks and considering an item for each unassigned task with weight equal to its smallest processing time among unassigned workers. After the derivation, any lower bound for the BPP is a valid bound for the ALWABP.
- One-machine scheduling based SALBP relaxation (Vilà and Pereira, 2014). In this lower bound, unassigned tasks are interpreted as jobs with delivery times and processing times. The bound takes into account packing constraints, precedence relations and the assignment of workers to stations.
- Unrelated parallel machine scheduling problem relaxation (Borba and Ritt, 2014; Vilà and Pereira, 2014). If precedence constraints are relaxed, any lower bound for the unrelated parallel machine scheduling problem provides a lower bound on the cycle time.
- Assignment problem (AP) bound (Vilà and Pereira, 2014). If the subset of tasks that cannot share a station among them is considered (for example, each task t with processing time larger than half the cycle time, $p_{wt} > C/2$, for each unassigned worker w), their optimal assignment to the workers corresponds to optimally solving AP, which may be used to strengthen BPP bounds.

In addition to these lower bounds, dominance rules are used to identify potentially dominated partial solutions. The following five rules are used in order to explore unnecessary partial solutions and their corresponding states:

- Memory rule (Vilà and Pereira, 2014). If the state of the partial solution is identical to a state already in the exploration graph, the partial solution does not need to be further explored.
- Maximum load rule (Scholl and Becker, 2006). The rule enables the algorithm to consider only branching decisions in which no additional tasks can be assigned while satisfying the cycle time constraint.
- Sewell–Jacobson rule (Sewell and Jacobson, 2012). The rule avoids symmetries caused by station assignments that could be postponed.
- Adapted version of the Jackson dominance rule (Scholl and Becker, 2006). This version of the rule tests whether some (dominated) task can be substituted for another (dominating) task while satisfying cycle time constraints.
- Operator dominance rule (Vilà and Pereira, 2014). The rule tests whether some (dominated) worker can be replaced by another (dominating) worker while satisfying cycle time constraints.

Two additional dominance rules were introduced to the original implementation described in Vilà and Pereira (2014). These rules are modified versions of the rules proposed in Pereira (2016a) and make use of the exploration graph.

The first rule, the generalized maximum load rule (Pereira, 2016a), considers two partial solutions s, s' and a task t such that $T(s) = T(s') \cup \{t\}$ and $W(s) = W(s')$. If these conditions hold, the state of s dominates the state of s' and thus we could prune any partial solution associated to s'. The rule is applied as follows: for each unassigned task t such that all of its predecessors already belong to the partial solution, we test whether the dominating state has been previously generated and stored and, if the answer is affirmative, the current partial solution is discarded.

The second rule is an extension of the Jackson dominance rule. The rule prunes suboptimal branching decisions in which a task t' could be substituted by a dominating task t without affecting the feasibility of the station assignment, see Vilà and Pereira (2014) for a specific definition on the conditions required for a task to dominate another. Consider two states s and s' with the same set of used workers, $W(s) = W(s')$, in which state s' includes task t' but not task t, $t' \in T(s')$, $t \notin T(s')$, and state s substitutes task t' by task t, $T(s) = T(s') \setminus \{t'\} \cup \{t\}$. Then, state s dominates state S'. Consequently, if a state s has been generated, any partial solution associated to state s' can be safely pruned.

4.2. An exact solution method for the MMR-ALWABP

Model (14)–(16) enables the development of a cutting planes framework to solve the MMR-ALWABP. Consider a, possibly empty, subset $P' \subseteq P^{\xi}$ of extreme scenarios, and solve model (18)–(20). Whenever a feasible solution $\mathbf{x}$ is found, the set of scenarios P' is updated in order to include its worst-case scenario. Upon termination, the algorithm provides an optimal solution.

$$\text{minimize } \theta, \tag{18}$$

subject to:

$$\theta \geq c(\mathbf{x}, f) - c(\mathbf{x}^*_f, f) \qquad \forall f \in P', \tag{19}$$

$$\mathbf{x} \in \mathcal{X}. \tag{20}$$

The previous scheme has been proposed in multiple works for other IDMR problems, see Feizollahi and Averbakh (2014), Furini et al. (2015), Mausser and Laguna (1999), Pereira and Averbakh (2011, 2013), among others. In such approaches, whenever a new incumbent solution $\tilde{\mathbf{x}}$ is found, Corollary 2 is used to ascertain its maximum regret scenario and its corresponding constraints are added to (19). Note that for any given scenario f, constraint set (19) is implemented as (21), and thus each scenario adds a constraint for each worker instead of a single constraint.

$$\theta \geq \sum_{t \in T} p^f_{wt} x_{wt} - c(\mathbf{x}^*_f, f) \qquad \forall f \in P',\ \forall w \in W, \tag{21}$$

As pointed out in Furini et al. (2015), this approach has been (wrongly) referred to as a Benders decomposition approach possibly due to some similarities with the mentioned decomposition, even when it does not fit within the theoretical framework of the Benders decomposition Benders (1962). Also note that constraint (19) can be introduced in the formulation as soon as any feasible solution is found. The addition of these cuts as soon as new feasible solutions are found leads to an algorithm within the branch-and-cut framework.

Note that the applicability of the procedure depends on the ability to successfully obtain the worst-case scenario of a solution. Consequently, an alternative method that does not rely on the direct evaluation of regret is proposed.

4.3. Heuristic evaluation of the maximum regret

The evaluation of the maximum regret of each feasible solution constitutes the main computational burden of the proposed exact method. Consequently, in some cases it may be useful to avoid an exact evaluation of regret even at the expense of losing any optimality guarantee from the method.

The literature covers some approaches to heuristically evaluate the maximum regret of any given solution. The discussed methods are based on substituting the evaluation of $c(\mathbf{x}^*_f, f)$ within constraint set (19) by $c'(\mathbf{x}^*_f, f)$, which corresponds to an approximation of the actual value of the optimal solution.

A first approach that approximates regret is based on underestimating it. This approach is discussed in Feizollahi and Averbakh (2014) in the context of the design of a heuristic solution procedure for the minmax regret quadratic assignment problem. Whenever the regret of a solution is to be evaluated, the worst-case scenario is determined, and the objective of the optimal solution under the scenario is approximated by heuristically solving the scenario using a tabu search metaheuristic. As the heuristic solution is no smaller than the optimal value, this method underestimates the regret of any given solution.

A second approximation involves overestimating the regret. This approach is considered in Furini et al. (2015) in the context of the design of heuristic solution procedures for the minmax regret knapsack problem. The idea builds upon a compact formulation for some IDMR combinatorial optimization problems with 0 duality gap that comply to Lemma 1, see Kasperski (2008).

For these problems, the evaluation of a solution under its worst-case scenario only depends on the upper limit of the uncertainty. Moreover, finding the worst-case alternative can be substituted for its dual, leading to a compact linear formulation, see examples in Furini et al. (2015), Kasperski (2008) and Pereira and Averbakh (2011).

When strong duality does not hold, as in the knapsack problem, it may still be possible to replace the evaluation of the worst-case scenario by the dual of its linear relaxation, and the model provides an overestimating bound on the regret of any given solution (the objective value of the worst-case alternative is underestimated as the duality gap may be larger than 0).

In this work we put forward an alternative, but methodologically equivalent, method to dual substitution, in which formulation (18)–(20) substitutes exact regret evaluation, equation set (19), with a linear relaxation of regret evaluation, equation set (21).

The method derives from the approach described in Section 4.2. Whenever the regret of a feasible solution, $\mathbf{x} \in \mathcal{X}$, is evaluated, the linear relaxation of the two-index formulation (1)–(9) of each of the $|W|$ possible worst-case scenarios, see Corollary 2, is solved, and its value is used to approximate $c(\mathbf{x}^*_f, f)$. Then, among the $|W|$ scenarios, the scenario f that maximizes the regret is used and its set of corresponding constraints, Eq. (21), is added to the model.

As in Furini et al. (2015), the method substitutes optimal regret evaluation for an approximation, but unlike in Furini et al. (2015) the function is not compact and needs to be approximated using (21) constraints. Nonetheless, the proposed formulation avoids non-linearities introduced by the need to solve multiple problems (rather than one) to ascertain the worst-case alternative and its corresponding worst-case scenario.

Let us denote the resulting problem by $R-MMR-ALWABP$, whose formulation corresponds to model (14)–(16) substituting $c(x^*_f, f)$ in (15) by the linear relaxation of (1)–(9) with task times according to scenario f.

Lemma 5. *The R-MMR-ALWABP is NP-hard.*

Proof. The proof follows Furini et al. (2015). Given an instance of the ALWABP-2, define an instance of the MMR-ALWABP with $p^-_{wt} = p^+_{wt} = p_{wt}$ $(t \in T,\ w \in W)$. Since there is only one scenario, we substitute (19) in (18), and the problem decomposes into two parts.

The formulation and objective of the first part, calculating $c(\mathbf{x}, f)$, corresponds to the optimal resolution of the ALWAPB-2, while approximating $c(\mathbf{x}^*_f, f)$ corresponds to a linear relaxation of the ALWABP-2. As the optimal solution requires to solve the ALWABP-2, its complexity results extend to the considered formulation (the relaxed model). Likewise, the ALWABP-2 extends the unrelated parallel machine scheduling problem and the bin packing problem, the relaxed problem is NP-hard (Garey and Johnson, 1979). □

Even if the complexity results are discouraging, from a practical perspective, the above approximation is computationally easier, given the fact that the optimization of the worst-case alternative does not require the resolution of multiple NP-hard problems. Moreover, the method only requires the exact resolution of the ALWABP-2 when the regret of the solution is evaluated. Consequently, if only a feasible solution is sought, retrieving its exact regret is irrelevant and no ALWABP-2 instance needs to be optimally solved.

5. Computational experiment

5.1. Implementation details and instance description

In order to test the quality of the exact and heuristic procedures proposed in this paper, both methods, as well as the midpoint scenario heuristic (the optimal solution for an ALWABP-2 instance with $p_{tw} = \frac{p^-_{tw}+p^+_{tw}}{2}$, $t \in T$, $w \in W$, is the solution suggested by the heuristic) were coded in C++ and compiled using GNU GCC 4.4.7. IBM ILOG CPLEX library, version 12.6.3, was used to solve linear programming and mixed integer programming (MIP) problems. The cutting planes based algorithm was implemented using the Concert Technology provided by the linear programming solver. The optimal solution of ALWABP-2 instances was obtained by the code described in Vilà and Pereira (2014) with the addition of the two dominance rules introduced in Section 4.1. Some tests were performed to evaluate the use of the linear programming solver instead of the branch, bound and remember method described in Vilà and Pereira (2014), but it was deemed impractical as the ability to solve even small instances was heavily hindered, see results reported in Tables 1 and 2 and their corresponding discussion below.

The tests reported in this section were carried out using a 32-core 2.0 GHz Intel Xeon with 128GB RAM, running the Linux operating system. The 32 cores were not used by the code and were only used to perform the resolution of 32 different instances simultaneously. Therefore, the code can be considered to have been executed in a single processor machine. Moreover to ensure that results are comparable to current commodity computers, the total amount of RAM allotted to each process was limited to 4GB.

Due to the lack of previous instances for the MMR-ALWABP, four instance sets were generated. Each set corresponds to a combination of methods to generate the uncertainty intervals and to generate a basic ALWABP-2 instance in which uncertainty is then introduced. Each element and method is described separately.

5.1.1. ALWABP-2 instances derived from the classical ALWABP-2 set

This set corresponds to the small size instances for the ALWABP-2 derived from the Roszieg and Heskia SALBP instances (Scholl and Klein, 1997). Large size instances derived from the Tonge and Wee-Mag SALBP instances are not considered as they are deemed challenging even to evaluate regret (none of the available codes is able to solve all of these instances within an hour time limit per instance).

The original Roszieg and Heskia instances have 25 and 28 tasks respectively, and differ mainly in the order strength, OS, the number of precedence relations with respect to the maximum number of precedence relations. The Roszieg instance has a high OS, 72.16% of precedence relations, and the Heskia instance has a low OS, 22.49%. Based on these instances, ALWABP-2 instances were generated according to three characteristics with two factors (low/high): (1) relation between number of tasks and number of workers; (2) variability of tasks times among workers; and (3) percentage of task-worker incompatibilities. Each of these three factors are respectively referred to as t./w., t.v. and t.w.i in the tables that summarize the results of the proposed solution methods.

Table 1
Average, column av.t., and maximum, column max.t., running times (in seconds) for the IP solver for the classical ALWABP instances. Instances are grouped according to the method to generate uncertainty intervals, column u.i.g.., the precedence graph (be it *Roszieg* or *Heskia*), and the three characteristics that control the original ALWABP-2 instance: the task to worker variability characteristic, column t./w., the task time variability characteristic, column t.v., and the number of incompatible task-worker characteristic, column t.w.i.

t./w.	t.v.	t.w.i.	u.i.g.	*Roszieg*		*Heskia*	
				av.t.	max.t.	av.t.	max.t.
Low	Low	Low	M.	15.73	34.62	20	54.69
Low	Low	Low	K.Z	16.43	35.78	19.04	54.74
Low	Low	High	M.	14.13	25.38	12.18	24.91
Low	Low	High	K.Z	14	26.86	13	26.83
Low	High	Low	M.	13.85	24.94	12.04	24.84
Low	High	Low	K.Z	14.53	25.32	12.85	26.72
Low	High	High	M.	14.28	27.38	11.6	25.4
Low	High	High	K.Z	14.29	26.43	13.26	26.39
High	Low	Low	M.	11.43	72.18	3.68	26.08
High	Low	Low	K.Z	17.16	54.28	4.41	21.6
High	Low	High	M.	13.13	45.38	4.28	28.75
High	Low	High	K.Z	11.4	33.88	5.93	32.71
High	High	Low	M.	16.3	51.93	4.73	35.22
High	High	Low	K.Z	16.87	41.36	4.22	27.52
High	High	High	M.	15.55	46.3	4.06	20.76
High	High	High	K.Z.	14.66	39.96	4.45	25

Table 2
Average, column av.t., and maximum, column max.t., running times (in seconds) for the IP solver for the SALBP derived instances. Instances are grouped according to the method to generate uncertainty intervals, column u.i.g.., the precedence graph (be it *Buxey, Gunther, Hahn, Kilbrid, Lutz1* or *Sawyer30*), and the three factors that control the original ALWABP-2 instance, the task to worker variability factor, column t./w., the task time variability factor, column t.v., and the number of incompatible task-worker factor, column t.w.i.

t./w.	t.wi.	u.i.g.	*Buxey*		*Gunther*		*Hahn*	
			Gap	av.t.	Gap	av.t.	Gap	av.t.
Low	Low	M.	0	3.08	0	4.9	0	185
Low	Low	K.Z.	0	4.49	0	10.84	0	458.75
Low	High	M.	0	3.45	0	4.69	0	293.74
Low	High	K.Z.	0	3.91	0	7.2	0	366.96
High	Low	M.	0	24.15	0	116.13	5.21	22285.36
High	Low	K.Z.	0	29.9	0	165.24	1.62	11961.46
High	High	M.	0	24.19	0	115.49	6.61	20062.64
High	High	K.Z.	0	29.95	0	176.94	0.56	12291.35
t./w.	t.wi.	u.i.g.	*Kilbrid*		*Lutz1*		*Sawyer30*	
			Gap	av.t.	Gap	av.t.	Gap	av.t.
Low	Low	M.	0	26.06	0	1.8	0	3.02
Low	Low	K.Z.	0	57.66	0	2.25	0	4.29
Low	High	M.	0	36.33	0	1.44	0	3.39
Low	High	K.Z.	0	48.67	0	1.91	0	4.19
High	Low	M.	0	5052.48	0	294.61	0	17.63
High	Low	K.Z.	0.67	12230.48	0	563.73	0	27.87
High	High	M.	0	4018.9	0	198.45	0	22.76
High	High	K.Z.	0.04	8929.81	0	340.21	0	30.24

Table 3
Results of the exact method for the classical ALWABP-2 instances. Instances are grouped according to the method to generate uncertainty intervals, column u.i.g.., the precedence graph (column Graph), and the three characteristics that define the original ALWABP-2 instance: the task to worker variability, column t./w., the task time variability, column t.v., and the number of incompatible task-worker pairs, column t.w.i. Each group contains 100 instances. The number of optimal solutions, column # opt, and the average gap, column Gap, for two different time limits, 600 s and 3600 s, are reported. Metrics on average and maximum computational times (in seconds) to reach and verify optimality, as well as the average number of cuts, i.e. scenarios, required to evaluate the maximum regret, column av. cuts., are also reported.

Characteristics					600 s		3600 s		No time limit		
Graph	t./w.	t.v.	t.w.i.	u.i.g.	# opt	Gap	# opt	Gap	av. t	max.t	av. cuts
Roszieg	Low	Low	Low	M.	100	0	100	0	2.9	10	36.6
Roszieg	Low	Low	Low	K.Z.	100	0	100	0	3.3	6.5	66.7
Roszieg	Low	Low	High	M.	100	0	100	0	0.9	4	21.2
Roszieg	Low	Low	High	K.Z.	100	0	100	0	1.2	6.1	33
Roszieg	Low	High	Low	M.	100	0	100	0	3.3	13.5	46.9
Roszieg	Low	High	Low	K.Z.	100	0	100	0	4.6	16.3	83.9
Roszieg	Low	High	High	M.	100	0	100	0	2.6	10.8	44.78
Roszieg	Low	High	High	K.Z.	100	0	100	0	3.3	12.6	74.6
Roszieg	High	Low	Low	M.	99	0	100	0	147.4	598.9	92.9
Roszieg	High	Low	Low	K.Z.	89	3.1	100	0	340.3	987.6	304.6
Roszieg	High	Low	High	M.	100	0	100	0	42.8	172.9	72.8
Roszieg	High	Low	High	K.Z.	100	0	100	0	77.9	265.3	172.7
Roszieg	High	High	Low	M.	100	0	100	0	120.4	347.9	116.5
Roszieg	High	High	Low	K.Z.	91	1.7	100	0	330.2	765.9	251.9
Roszieg	High	High	High	M.	100	0	100	0	123.9	402.4	118.9
Roszieg	High	High	High	K.Z.	85	6.13	100	0	362.4	1157.9	260.8
Heskia	Low	Low	Low	M.	100	0	100	0	12.8	62.3	88.3
Heskia	Low	Low	Low	K.Z.	100	0	100	0	31.1	215	165.2
Heskia	Low	Low	High	M.	100	0	100	0	3.8	21.4	66.4
Heskia	Low	Low	High	K.Z.	100	0	100	0	9.7	84.7	118.2
Heskia	Low	High	Low	M.	100	0	100	0	8.8	46.4	88.7
Heskia	Low	High	Low	K.Z.	100	0	100	0	19.4	68.7	142.1
Heskia	Low	High	High	M.	100	0	100	0	10.8	89.1	89.1
Heskia	Low	High	High	K.Z.	100	0	100	0	27.1	146.6	156.3
Heskia	High	Low	Low	M.	99	0.43	100	0	85.4	1487.9	216.5
Heskia	High	Low	Low	K.Z.	82	6.56	98	0.35	441.4	5016.5	508.3
Heskia	High	Low	High	M.	100	0	100	0	34.6	423.9	192.4
Heskia	High	Low	High	K.Z.	91	2.86	100	0	225.3	3400.5	379.5
Heskia	High	High	Low	M.	98	1.29	100	0	104.9	2697.5	219.2
Heskia	High	High	Low	K.Z.	93	3.44	99	0.53	276.1	8443.3	348.9
Heskia	High	High	High	M.	99	0.75	100	0	79.8	2238.8	239.1
Heskia	High	High	High	K.Z.	96	1.83	100	0	144	1706.1	371.6

Each characteristic influences how the ALWABP instances were derived from their corresponding SALBP instances. The task to worker variability characteristic controls the number of workers in the ALWABP instances. If the factor is high, the number of tasks is four times higher than the number of workers, and if it is low, the number of tasks is seven times higher than the number of workers.

The variability of task times among workers characteristic controls the task time of each worker with the exception of the first one, in which the task times, p_t, of the SALBP instance are used, hence $p_{t1} = p_t$ for each task $t \in T$. For the remaining workers, the task times are randomly drawn from a discrete uniform distribution $\mathcal{U}\{1, 3p_t\}$ if the factor is high and from a discrete uniform distribution $\mathcal{U}\{1, p_t\}$ if the factor is low.

The task-worker incompatibility characteristic controls the probability that a worker cannot perform a task. For each worker, $w = 2, \ldots, |W|$, and task, $t \in T$, the probability that the worker cannot perform the task equals 10% if the factor is low and 20% if the factor is high.

For each combination of instance and configuration of the three characteristics, the ALWABP-2 instance set contains 10 randomly generated instances. This amounts to sixteen groups of 10 instances for each of the two original SALBP instances.

5.1.2. ALWABP-2 instances derived from other SALBP instances

In addition to the Roszieg and Heskia instances, six other small- to medium-sized instances from the SALBP reference set, see description in Scholl and Klein (1997), were used to construct additional ALWABP-2 instances. These instances correspond to *Buxey* (29 tasks, OS 50.7%), *Gunther* (35 tasks, OS 59.5%), *Hahn* (53 tasks, OS 83.8%), *Kilbrid* (45 tasks, OS 44.6%), *Lutz1* (32 tasks, OS 83.5%) and *Sawyer30* (30 tasks, OS 44.8%). The instances were chosen because they have been used in computational experiments of other robust based formulations for assembly line balancing problems, (Hazir and Dolgui, 2013; 2015). Moreover, instances of this size cannot be considered trivial, since, on the one hand, they already pose difficulties for their exact solution to deterministic versions of the ALWABP-2 and, on the other hand, they provide evidence of the strengths and limitations of the proposed approaches. For each of the six SALBP instances, four instances were generated according to characteristic (1) and (3) with the same two factors (low and high) described above. Characteristic (2) is not used because task time variability is already achieved with the introduction of uncertainty intervals (see below). Consequently, for each task $t \in T$ and worker $w \in W$, the task time is set to be equal to the original task times, $p_{wt} = p_t$.

For each combination of instance and factor of the characteristics, an instance is generated. Consequently, a total of 24 base instances were constructed.

5.1.3. Uncertainty intervals

Two alternative methods found in the literature to generate intervals are considered. Each method defines a different strategy to generate uncertainty intervals maintaining the rest of the characteristics of the instance.

(1) The first method was proposed in Kasperski and Zielinski (2004), and it is denoted by *Kasperski–Zielinski* instances, or K.Z. for short. The method considers each parameter under uncertainty p_{tw} $(t \in T, w \in W)$ separately, and randomly generates

Table 4
Results of the heuristic method for the classical ALWABP-2 instances. Instances are grouped according to the method to generate uncertainty intervals, column u.i.g.., the precedence graph, column Graph, and the three characteristics that define the original ALWABP-2 instance: the task to worker variability, column t./w., the task time variability, column t.v., and the number of incompatible task-worker pairs, column t.w.i. Each group contains 100 instances. The number of optimal solutions, column # opt, and the average gap, column Gap, for the midpoint scenario, column *Midpoint*, and the heuristic-based regret evaluation method, column *Heuristic regret*, are reported.

Characteristics					*Midpoint*		*Heuristic regret*	
Graph	t./w.	t.v.	t.wi.	u.m.	# opt	Gap	# opt	Gap
Roszieg	Low	Low	Low	M.	45	18.42	81	5.86
Roszieg	Low	Low	Low	K.Z.	42	6.84	65	3.66
Roszieg	Low	Low	High	M.	66	11.66	68	11.25
Roszieg	Low	Low	High	K.Z.	50	7.89	60	7.71
Roszieg	Low	High	Low	M.	31	17.32	77	5.43
Roszieg	Low	High	Low	K.Z.	38	6.87	63	2.76
Roszieg	Low	High	High	M.	38	16.34	71	5.74
Roszieg	Low	High	High	K.Z.	46	6.15	71	2.79
Roszieg	High	Low	Low	M.	50	26.82	99	0.5
Roszieg	High	Low	Low	K.Z.	49	8.27	73	6.59
Roszieg	High	Low	High	M.	48	25.15	94	2.33
Roszieg	High	Low	High	K.Z.	43	9.35	73	3.81
Roszieg	High	High	Low	M.	44	20	92	2.42
Roszieg	High	High	Low	K.Z.	33	10.18	71	4.88
Roszieg	High	High	High	M.	36	24.52	91	3.5
Roszieg	High	High	High	K.Z.	30	10.91	72	8.9
Heskia	Low	Low	Low	M.	2	19.68	40	5.45
Heskia	Low	Low	Low	K.Z.	1	17.48	37	3.3
Heskia	Low	Low	High	M.	9	16.52	37	6.17
Heskia	Low	Low	High	K.Z.	5	15	24	5.05
Heskia	Low	High	Low	M.	7	16.14	42	5.21
Heskia	Low	High	Low	K.Z.	10	12.19	22	4.53
Heskia	Low	High	High	M.	7	18.33	39	5.18
Heskia	Low	High	High	K.Z.	8	13.78	19	4.21
Heskia	High	Low	Low	M.	12	23.45	72	4.75
Heskia	High	Low	Low	K.Z.	10	20.83	46	9.37
Heskia	High	Low	High	M.	13	23.26	74	4.83
Heskia	High	Low	High	K.Z.	6	22.7	50	6.65
Heskia	High	High	Low	M.	16	19.69	61	6.36
Heskia	High	High	Low	K.Z.	12	23.46	45	11.35
Heskia	High	High	High	M.	11	21.81	67	4.61
Heskia	High	High	High	K.Z.	4	20.33	46	7.16

the minimum value of its interval p^-_{tw} from the integer-valued uniform distribution on the interval $[1,p_{tw}]$, and the maximum value p^+_{tw} from the integer-valued uniform distribution on the interval $[p^-_{tw}, p^-_{tw} + p_{tw}]$.

(2) The second method was proposed in Montemanni et al. (2007), and it is denoted by *Montemanni* instances, or M. for short. It considers each parameter under uncertainty p_{tw} $(t \in T, w \in W)$ separately, and randomly generates the maximum value of its interval p^+_{tw} from the integer-valued uniform distribution on the interval $[1,p_{tw}]$, and the minimum value p^-_{tw} from the integer-valued uniform distribution on the interval $[1,p^+_{tw}]$.

Please note that these distributions are slightly different from the original descriptions given in Kasperski and Zielinski (2004) and Montemanni et al. (2007), as the authors allowed a parameter to take value 0. As a 0 processing time would imply that the task does not require time to be performed, the value was avoided during the generation of the instances. The method used to generate the uncertainty intervals is referred to as u.i.g. in the tables that summarize the results.

For each combination of base instance and uncertainty interval, ten random instances were generated. Consequently, a total of 3200 instances derived from the classical ALWABP-2 set and 480 instances derived from the SALBP set were generated.

5.1.4. Computational complexity of the instances

A preliminary test was conducted to evaluate the applicability of an MIP based method to solve the deterministic (basic) version of the MMR-ALWABP. For each of the 3680 generated instances, ten realizations of processing times were generated and the deterministic MIP formulation, Eqs. (1)–(9), was solved using the default CPLEX configuration with a 10-h time limit. In order to obtain the processing times of each pair of task-worker, task times were generated from a discrete uniform distribution $\mathcal{U}\{p^-_{tw}, p^+_{tw}\}$. Therefore, we can regard them as possible scenarios of the corresponding MMR-ALWABP instance.

Table 1 provides average and maximum execution times in seconds for each instance from the ALWABP-2 set grouped according to their different characteristics. The results show that all of the instances are solvable within reduced running times, less than a minute.

Table 2 reports the results for the instances derived from the SALBP. In this case, the IP solver cannot verify optimality for all of the tested instances. Consequently, Table 2 provides the average running time among all instances from a group, as well as the average gap and the optimality gap reported by the IP solver. The average gap is calculated as $100 \cdot (UB - LB)/UB$, in which *UB* and *LB* are the upper and lower bounds provided by the IP solver, and the average running time considers the totality of allotted time, that is 10 h, or the time required to verify the optimality of the incumbent solution. The results from Table 2 show that the requirements of slightly larger instances (instances with 45 and 53 tasks, like *Kilbrid* and *Hahn*, respectively) of the ALWABP are challenging for the IP formulation even with a 10-h time limit.

Note that the MMR-ALWABP, the proposed heuristic relaxation and the original ALWABP-2 all share the same solution space and only differ in their objective. If we consider the additional hardness introduced by the objective function, which relies on successive approximations and the resolution of auxiliary problems, we are inclined to deem the set of instances challenging, at least for any solution method based on an MIP solver. Note that this result accords with previous results found in the literature for robust assembly line balancing problems whose solution methods are based on an MIP approach (see Hazir and Dolgui, 2013; 2015, among others).

5.2. Results for classical ALWABP-2 instances

Table 3 provides the results of the exact method to the instances derived from the ALWABP-2 set. Instances are grouped according to five characteristics, namely: (1) the method to generate uncertainty intervals, column u.i.g., (2) the precedence graph, column Graph, and the three characteristics of the original ALWABP-2 instance, which are (3) the relation between number of tasks and workers, column t./w., (4) the variability of task times, column t.v., and (5) the percentage of task-worker incompatibilities, column t.w.i. Each group is composed of 100 instances and results with two different running time limits (600 s and 3600 s) are reported. The number of optimal solutions, columns # opt, and the optimality gap, column Gap, for the exact algorithm under two run time limits, columns 600 s and 3600 s, are reported. The optimality gap is measured as $100 \cdot (UB - LB)/UB$, in which *UB* and *LB* correspond to the upper and the lower bounds reported by the MIP solver. Two additional columns with the results under no time limit are also provided, see columns no time limit. These columns report the average and maximum running times, in seconds, required by the exact method to reach and verify the optimality of the solution.

As Table 3 highlights, the exact algorithm is able to solve most of the instances within very reduced running times. While some instances require larger running times, these running times can still be considered to be reasonably low when compared to the

Table 5
Results of the exact method for the set of instances derived from the SALBP. Instances are grouped according to the original SALBP instance, column Graph, and the instance characteristic, namely: (1) task to worker variability, column t./w.; (2) number of incompatible task-worker pairs, column t.w.i.; and (3) method used to generate uncertainty intervals, column u.i.g. Each group contains ten instances. The number of optimal solutions found, column #opt, and the average optimality gap, column Gap, are reported for different run time limits, 600, 3600 and 36,000 s. Additionally, the average number of cuts, i.e. scenarios, generated are also reported.

Characteristics				600 s		3600 s		36,000 s		
Graph	t./w.	t.wi.	u.i.g.	# opt	Gap	# opt	Gap	# opt	Gap	av. cuts
Buxey	Low	Low	M.	10	0	10	0	10	0	103.2
Buxey	Low	Low	K.Z.	10	0	10	0	10	0	238.8
Buxey	Low	High	M.	10	0	10	0	10	0	58.4
Buxey	Low	High	K.Z.	10	0	10	0	10	0	91.6
Buxey	High	Low	M.	1	49.4	8	1.11	10	0	377.3
Buxey	High	Low	K.Z.	0	70.54	0	42.2	10	0	1060.5
Buxey	High	High	M.	4	33.45	9	2.5	10	0	364.7
Buxey	High	High	K.Z.	0	64.71	7	11.73	10	0	753.9
Sawyer30	Low	Low	M.	10	0	10	0	10	0	105.6
Sawyer30	Low	Low	K.Z.	10	0	10	0	10	0	326.4
Sawyer30	Low	High	M.	10	0	10	0	10	0	84.4
Sawyer30	Low	High	K.Z.	10	0	10	0	10	0	136.4
Sawyer30	High	Low	M.	2	38.31	10	0	10	0	373.8
Sawyer30	High	Low	K.Z.	0	65.38	0	44.86	10	0	1148
Sawyer30	High	High1	M.	3	31.36	10	0	10	0	309.4
Sawyer30	High	High	K.Z.	0	66.55	4	14.87	10	0	967.4
Lutz1	Low	Low	M.	10	0	10	0	10	0	82.4
Lutz1	Low	Low	K.Z.	10	0	10	0	10	0	70
Lutz1	Low	High	M.	10	0	10	0	10	0	49.6
Lutz1	Low	High	K.Z.	10	0	10	0	10	0	46.4
Lutz1	High	Low	M.	0	79.48	0	63.74	10	0	541.6
Lutz1	High	Low	K.Z.	1	83.82	1	71.55	8	5.6	688
Lutz1	High	High	M.	0	75.17	5	37.18	10	0	369.6
Lutz1	High	High	K.Z.	2	77.91	4	42.86	10	0	455.2
Gunther	Low	Low	M.	10	0	10	0	10	0	172.5
Gunther	Low	Low	K.Z.	7	6.11	10	0	10	0	419.5
Gunther	Low	High	M.	10	0	10	0	10	0	123
Gunther	Low	High	K.Z.	10	0	10	0	10	0	223
Gunther	High	Low	M.	0	86.93	1	49.83	10	0	507.2
Gunther	High	Low	K.Z.	0	85.23	0	61.56	8	6.7	1216
Gunther	High	High	M.	1	70.54	5	18.18	10	0	371.2
Gunther	High	High	K.Z.	0	85.39	1	53.42	10	0	763.2
Kilbrid	Low	Low	M.	0	88.5	1	48.34	10	0	604.2
Kilbrid	Low	Low	K.Z.	0	91.8	0	72.83	4	20.26	2653.2
Kilbrid	Low	High	M.	0	94.15	2	45.77	10	0	427.2
Kilbrid	Low	High	K.Z.	0	97.82	0	72.89	8	3.49	1710
Kilbrid	High	Low	M.	0	98.89	0	87.71	2	53.55	968
Kilbrid	High	Low	K.Z.	0	100	0	99.13	1	72.86	958.1
Kilbrid	High	High	M.	0	93.42	1	81.03	1	57.62	823.9
Kilbrid	High	High	K.Z.	0	100	0	98.8	0	81.25	763.4
Hahn	Low	Low	M.	0	97.82	1	59	10	0	306.6
Hahn	Low	Low	K.Z.	0	96.26	6	18.73	10	0	228.9
Hahn	Low	High	M.	1	86.98	8	16.66	10	0	174.3
Hahn	Low	High	K.Z.	1	88.18	9	10	10	0	148.4
Hahn	High	Low	M.	0	100	0	94.97	3	28.63	1250.6
Hahn	High	Low	K.Z.	0	100	0	97.31	9	1.62	1323.4
Hahn	High	High	M.	0	93.49	0	86.19	4	24.17	1101.1
Hahn	High	High	K.Z.	0	99.59	2	66.67	7	17.26	913.9

running times allotted to robust approaches reported in the literature for line balancing problems, see Hazir and Dolgui (2013, 2015), among others. Moreover, the average number of scenarios required to approximate P^{ξ}, P' in Eq. (19) remains within reasonable values, column av. cuts.

An analysis of the characteristics of the instance and their impact on the ability of the exact method to reach optimal solutions show that *Heskia* instances can be considered to be more difficult than *Roszieg* instances. Since the *Roszieg* and *Heskia* instances have approximately the same number of tasks (25 and 28 instances, respectively), the differences among the results for these instances should be attributed to the differences between their precedence graphs: the *Heskia* graph has fewer precedence relations between tasks than the *Roszieg* graph. This result follows common intuition, as the number of candidate solutions (number of possible assignments of tasks to stations fulfilling precedence constraints) is larger for low precedence-constrained graphs than for high precedence-constrained graphs. A similar interpretation can be given to the task-worker and incompatibility pairs characteristics, columns t./w. and t.w.i respectively. The above rationale extends to the number of cuts required to evaluate the maximum regret. Instances with a *high* task-worker characteristic have a larger number of workers (hence, stations) and a larger number of alternative solutions (derived from the combinatorial assignment of workers to stations). Similarly, instances with a *low* incompatibility characteristic have a larger number of feasible assignments, and thus a larger number of alternative assignments.

Another feature that influences the time required by the exact method to verify optimality corresponds to the uncertainty interval generation method. The results show that instances with larger uncertainty intervals, those generated according to *Kasperski-Zielinski*, are more difficult to solve than instances with smaller uncertainty

Table 6
Results of the heuristic methods for the set of instances derived from the SALBP. Instances are grouped according to the original SALBP instance, column Graph, and the instance characteristic, namely: (1) task to worker variability, column t./w.; (2) number of incompatible task-worker pairs, column t.w.i.; and (3) method used to generate uncertainty intervals, column u.i.g. Each group contains ten instances. For each heuristic method, be it Midpoint, columns *Midpoint*, or Heuristic regret evaluation, columns *Heuristic Regret*, three values are reported: (1) the number of instances in which the algorithm is able to find the best solution found by any of the algorithms proposed in this work, column # best; (2) the number of instances in which the heuristic algorithm provides the same as or a better solution than the solution provided by the alternative heuristic, column # c.alt; and (3) the optimality gap when compared to the lower bound provided by the exact algorithm, column Gap.

Characteristics				Midpoint			Heuristic regret		
Graph	t./w.	t.wi.	u.i.g.	# best	# c.alt	Gap	# best	# c.alt	Gap
Buxey	Low	Low	M.	1	3	12.21	7	9	3.55
Buxey	Low	Low	K.Z.	0	3	11.25	3	10	3.87
Buxey	Low	High	M.	3	7	10.57	4	9	9.33
Buxey	Low	High	K.Z.	1	3	12.03	3	8	4.29
Buxey	High	Low	M.	1	2	19.36	8	9	2.34
Buxey	High	Low	K.Z.	1	2	11.82	8	9	1.28
Buxey	High	High	M.	2	2	17.66	6	9	4.65
Buxey	High	High	K.Z.	0	0	13.99	6	10	2.26
Sawyer30	Low	Low	M.	2	3	11.36	4	9	5.62
Sawyer30	Low	Low	K.Z.	1	1	12.07	4	9	2.07
Sawyer30	Low	High	M.	1	3	22.95	3	9	11.02
Sawyer30	Low	High	K.Z.	0	4	11.61	3	9	5.17
Sawyer30	High	Low	M.	2	3	17.52	8	10	2.54
Sawyer30	High	Low	K.Z.	1	1	13.47	8	10	0.95
Sawyer30	High	High	M.	1	2	18.51	8	10	2.36
Sawyer30	High	High	K.Z.	0	1	12.47	5	10	3.25
Lutz1	Low	Low	M.	0	2	13.85	5	8	5.56
Lutz1	Low	Low	K.Z.	5	6	5.17	5	8	1.51
Lutz1	Low	High	M.	2	3	20.92	2	9	16.07
Lutz1	Low	High	K.Z.	0	4	15.39	2	9	7.16
Lutz1	High	Low	M.	0	0	23.13	8	10	0.18
Lutz1	High	Low	K.Z.	2	2	19.03	7	8	5.79
Lutz1	High	High	M.	2	2	13.8	8	10	0.8
Lutz1	High	High	K.Z.	3	3	9.42	7	9	0.06
Gunther	Low	Low	M.	0	0	20.14	7	10	2.1
Gunther	Low	Low	K.Z.	0	0	12.6	6	10	1.03
Gunther	Low	High	M.	0	2	18.28	5	10	5.43
Gunther	Low	High	K.Z.	2	2	11.39	5	9	3.5
Gunther	High	Low	M.	1	2	18.94	6	9	4.52
Gunther	High	Low	K.Z.	0	0	23.04	6	10	7.86
Gunther	High	High	M.	1	2	19.43	8	10	1.43
Gunther	High	High	K.Z.	0	0	16.68	9	10	0.34
Kilbrid	Low	Low	M.	1	1	24.15	8	10	1.12
Kilbrid	Low	Low	K.Z.	0	0	30.62	8	10	18.43
Kilbrid	Low	High	M.	0	0	20.78	5	10	3.15
Kilbrid	Low	High	K.Z.	0	0	12.86	4	10	4.71
Kilbrid	High	Low	M.	4	5	66.77	6	6	56.92
Kilbrid	High	Low	K.Z.	6	6	74	4	5	74.66
Kilbrid	High	High	M.	4	5	65.63	4	7	62.07
Kilbrid	High	High	K.Z.	8	8	81.25	4	4	82.01
Hahn	Low	Low	M.	0	0	17.9	4	10	2.62
Hahn	Low	Low	K.Z.	2	2	8.85	6	9	3.01
Hahn	Low	High	M.	1	1	15.68	5	10	5.31
Hahn	Low	High	K.Z.	1	1	11.41	8	10	0.66
Hahn	High	Low	M.	2	8	40.5	1	2	52.46
Hahn	High	Low	K.Z.	3	5	13.87	4	7	18.85
Hahn	High	High	M.	3	8	41.07	1	2	57.77
Hahn	High	High	K.Z.	5	10	27.74	3	3	49.28

intervals, such as those proposed in *Montemanni*. Note that the effect of this variability terms of the task times is relative and not absolute, since instances with *high* task time variability, column t.v., have larger task time values but do not seem to be more difficult than instances with *low* task time variability.

Table 4 reports results for the heuristic based on approximating regret evaluation, columns *Heuristic regret*, as well as the midpoint heuristic, columns *Midpoint*. While the instances of this set can be optimally solved, our goal is to assess the quality of these heuristics when compared to the optimal solution.

Table 4 groups tasks like Table 3. The table shows that the regret evaluation heuristic provides better solutions regardless of the characteristics of the instances. Moreover, the average gap offered by the midpoint scenario heuristic is larger than the average gap offered for other min-max regret problems studied in the literature, like in Pereira and Averbakh (2011, 2013). These results may be attributed to differences among minmax regret problems (in previous cases, the midpoint scenario heuristic provided a 2-approximation).

5.3. Results for new ALWABP-2 instances

Tables 5 and 6 provide results for the proposed algorithms when applied to the new set of instances directly derived from the SALBP.

The results of the exact method, see Table 5, show that the cutting planes framework is still capable of solving most of the instances. Moreover, when the results of Tables 2 and 5 are compared, we can observe that the groups of instances that can be considered to be difficult coincide, that is the largest instances

with large task-worker ratio. This results reinforce the intuition that the bottleneck of the proposed method lies in the ability to solve the MIP formulation. Notice that, while the rest of instances can be optimally solved, the solution comes at the cost of additional computing time requirements.

A similar analysis can be made based on the results of the heuristic methods, see Table 6. First, the heuristic regret evaluation method clearly outperforms the midpoint-scenario heuristic for most of the instances of the set, with the exception of the *Kilbrid* and *Hahn* instances with high task-worker variability. This set corresponds to the set of instances in which the MIP formulation fails to provide optimal solutions.

Note that although we do not report running times for the heuristic regret calculation method, its running time is comparable to the running time reported in Table 5 for the exact method minus the time required for the exact solution of the ALWABP-2 instances, which is negligible when compared to the time required by the MIP solver. Consequently, the heuristic regret evaluation method, while improving the quality of the midpoint heuristic solution, can only be considered as a viable solution approach when the exact resolution of ALWABP-2 instances is the bottleneck of the application. Otherwise, the exact solution method is the preferred option since the time required for the exact evaluation of the regret is irrelevant when compared to the time required to search the solution space of the MMR-ALWABP.

6. Conclusions

This paper considers an assembly line balancing problem with heterogeneous workforce known as ALWABP-2 and introduces uncertainty intervals within the problem data. The objective is then to obtain a robust solution, a good solution for any parameter realization. Among the alternative robust approaches, this work considers the minmax regret objective in which we aim to minimize the worst regret (the maximum difference among all of the scenarios between the quality of the proposed solution for the scenario and the quality of the optimal solution for the scenario). The contributions of this paper are threefold:

- First, the paper contributes to the assembly line balancing literature in two different ways.
  On the one hand, interval data minmax regret (IDMR) is introduced within the line balancing literature and a mathematical model for a generalized assembly line balancing problem formulation is put forward. IDMR offers a simple method to describe uncertainty (using intervals) and reflects a robust criterion in which we try to hedge against risk by remaining as close as possible to the optimal solution in any possible outcome. To our knowledge, this work is the first to consider such an objective within the line balancing literature, even when minmax regret approaches were suggested in Dolgui and Kovalev (2012) as valid formulations to introduce robustness within assembly line balancing. Moreover, from a practical standpoint, IDMR not only provides a simple approach to define uncertainty but also offers a starting solution which can be later improved through rebalancing the assembly line as more information becomes available (Gamberini et al., 2009).
  On the other hand, an exact procedure as well as a heuristic one are introduced. The exact method allows to optimally solve instances with over 50 tasks and 7 workers, instances with sizes comparable to those of previously reported robust methods for line balancing. The heuristic method avoids the optimal evaluation of discrete solutions of the problem which may become the computational bottleneck of the method on large instances.
- Second, new results for IDMR formulations are given. Due to the structure of the objective function, an alternative method to evaluate regret is developed. The method extends previous results applicable to problems with uncertainty on the objective coefficients to problems with uncertainty on the coefficient constraints used to evaluate the cost function (like the problem in hand). These results may be applicable to other problems such as the unrelated parallel machine problem (Martello et al., 1997, among others). Moreover, the inability of the midpoint-scenario heuristic to obtain a 2-approximation algorithm for the problem under study is also shown.
- Third, a general heuristic for IDMR problems is introduced. The method generalizes the ILP based heuristic introduced in Furini et al. (2015) and it is applicable to alternative IDMR problems. Moreover, the method avoids the exact evaluation of regret, which is the most computational intensive task in many of the previously proposed methods for IDMR problems.

Note that the above results apply to an absolute maximum regret objective. While this objective is commonly used in other IDMR works, alternative approaches exist, like absolute robustness (Aissi et al., 2009) or relative regret (Averbakh, 2005; Kouvelis and Yu, 1997) objectives.

For the ALWABP-2, absolute robustness (also known as minmax) reduces to the deterministic problem (the worst scenario for any solution corresponds to an extreme scenario in which all tasks times are equal to their respective maximum values) and thus it does not require any special consideration, but the relative regret objective poses additional complexities that need further attention.

Relative regret aims at minimizing the ratio between the regret and the optimal value for the worst-case scenario. This problem has attracted less attention than its absolute regret counterpart from the scientific literature, see some exceptions in Averbakh (2005), Kouvelis and Yu (1997) and Mausser and Laguna (1999). This may be due to the complex structure of its objective function, which negatively impacts the ability of the available or proposed methods to offer either exact or heuristic solution methods. For the ALWABP this issue could further hinder the ability of any proposed solution procedure in which regret evaluation requires the optimal resolution of several difficult, NP-hard, problems. Moreover, theoretical contributions may be needed as the literature is mostly focused on formulations in which Lemma 1 holds, see Averbakh (2005).

As a final reflection, we would like to draw attention to three different open topics, each of them related to assembly line balancing or IDMR problems:

- Most of the literature covering exact solution methods on assembly line balancing has been focused on branch-and-bound based enumeration methods. While these methods outmatch MIP-based approaches, the literature covers multiple objectives and formulations in which there exists a body of state-of-the-art based on MIP formulations, such as in robust optimization. For these problems, enhanced MIP methods would lead to improvements on the methodology used in the present and other works, and thus constitute relevant contributions to the state of the art in assembly line balancing solution procedures.
- Most of the previous work on IDMR based formulations has been mainly limited to problems in which classical results, Lemma 1, are applicable. These problems would require alternative regret evaluation methods, and therefore some of the ideas proposed in this work might be used in the design of specific regret evaluation methods for these problems.
- While this work tackles a basic line balancing formulation with heterogeneity issues among different workstations, the literature covers many other alternative models. The applicability of the proposed method, after modifications, to those problems,

like the robotic assembly line balancing problem (Gao et al., 2009; Levitin et al., 2006), or to other assembly line configurations, like multi-manned stations (Giglio et al., 2017), U-shaped assembly lines (Oksuz et al., 2017) or parallel assembly lines (Araújo et al., 2015) are open areas of research.

Acknowledgments

This research has been funded by the research grant number 1150306 “Heterogeneous assembly line balancing problems with process selection features” from the “Fondo Nacional de Desarrollo Científico y Tecnológico” (FONDECYT) of the Ministry of Education of Chile. This research was also partially supported by the supercomputing infrastructure of the NLHPC (ECM-02) in which some preliminary computational tests were conducted.

References

Aissi, H., Bazgan, C., Vanderpooten, D., 2009. Min-max and min-max regret versions of combinatorial optimization problems: a survey. Eur. J. Oper. Res. 197, 427–438.

Araújo, F., Costa, A., Miralles, C., 2015. Balancing parallel assembly lines with disabled workers. Eur. J. Ind. Eng. 9 (3), 344–365. doi:10.1504/EJIE.2015.069343.

Assunção, L., Noronha, T.F., Santos, A.C., Andrade, R., 2017. A linear programming based heuristic framework for min-max regret combinatorial optimization problems with interval costs. Comput. Oper. Res. 81, 51–66. doi:10.1016/j.cor.2016.12.010.

Averbakh, I., 2005. Computing and minimizing the relative regret in combinatorial optimization with interval data. Discrete Optim. 2 (4), 273–287. doi:10.1016/j.disopt.2005.07.001.

Averbakh, I., Lebedev, V., 2004. Interval data minmax regret network optimization problems. Discrete Appl. Math. 138, 289–301.

Battaïa, O., Dolgui, A., 2013. A taxonomy of line balancing problems and their solution approaches. Int. J. Prod. Econ. 142, 259–277.

Becker, C., Scholl, A., 2006. A survey on problems and methods in generalized assembly line balancing. Eur. J. Oper. Res. 168, 694–715. doi:10.1016/j.ejor.2004.07.023.

Ben-Tal, A., Ghaoui, L.E., Nemirovski, A., 2009. Robust Optimization. Princeton University Press, Princeton, New Jersey.

Benders, J., 1962. Partitioning procedures for solving mixed integer variables programming problems. Numerische Mathematik 4, 238–252.

Bertsimas, D., Sim, M., 2003. Robust discrete optimization and network flows. Math. Programm. 98, 49–71.

Blum, C., Miralles, C., 2011. On solving the assembly line worker assignment and balancing problem via beam search. Comput. Oper. Res. 38, 328–339.

Borba, L., Ritt, M., 2014. Exact and heuristic methods for the assembly line worker assignment and balancing problem. Comput. Oper. Res. 45, 87–96.

Cakir, B., Altiparmak, F., Dengiz, B., 2011. Multi-objective optimization of a stochastic assembly line balancing: a hybrid simulated annealing algorithm. Comput. Ind. Eng. 60 (3), 376–384. doi:10.1016/j.cie.2010.08.013.

Conde, E., 2017. A minimum expected regret model for the shortest path problem with solution-dependent probability distributions. Comput. Oper. Res. 77, 11–19. doi:10.1016/j.cor.2016.07.007.

Dolgui, A., Kovalev, S., 2012. Scenario based robust line balancing: Computational complexity. Discrete Appl. Math. 160, 1955–1963.

Feizollahi, M.J., Averbakh, I., 2014. The robust (Minmax Regret) quadratic assignment problem with interval flows. INFORMS J. Comput. 26, 321–335.

Furini, F., Iori, M., Martello, S., Yagiura, M., 2015. Heuristic and exact algorithms for the interval min-max regret knapsack problem. INFORMS J. Comput. 27, 392–405.

Gamberini, R., Gebennini, E., Grassi, A., Regattieri, A., 2009. A multiple single-pass heuristic algorithm solving the stochastic assembly line rebalancing problem. Int. J. Prod. Res. 47 (8), 2141–2164. doi:10.1080/00207540802176046.

Gao, J., Sun, I., Wang, I., Gen, M., 2009. An efficient approach for type ii robotic assembly line balancing problems. Comput. Ind. Eng. 56, 1065–1080.

Garey, M., Johnson, D., 1979. Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H. Freeman and Company.

Giglio, D., Paolucci, M., Roshani, A., Tonelli, F., 2017. Multi-manned assembly line balancing problem with skilled workers: a new mathematical formulation. IFAC-PapersOnLine 50 (1), 1211–1216. doi:10.1016/j.ifacol.2017.08.344.

Gurevsky, E., Battaïa, O., Dolgui, A., 2012. Balancing of simple assembly lines under variations of task processing times. Ann. Oper. Res. 201, 265–286.

Hazir, O., Dolgui, A., 2013. Assembly line balancing under uncertainty: robust optimization models and exact solution method. Comput. Ind. Eng. 65, 261–267.

Hazir, O., Dolgui, A., 2015. A decomposition based solution algorithm for u-type assembly line balancing with interval data. Comput. Oper. Res. 59, 126–131.

Kasperski, A., 2008. Discrete Optimization with Interval Data. Springer-Verlag, Berlin.

Kasperski, A., Zielinski, P., 2004. Minimizing Maximal Regret in Linear Assignment Problems with Interval Data (Raport serii PREPRINTY Nr. 007). Technical Report. Instytut Matematyki PWr., Wroclaw.

Kasperski, A., Zieliński, P., 2006. An approximation algorithm for interval data minmax regret combinatorial optimization problems. Inf. Process. Lett. 97, 177–180.

Kouvelis, P., Yu, G., 1997. Robust Discrete Optimization and Its Applications. Kluwer Academic.

Levitin, G., Rubinovitz, J., Shnits, B., 2006. A genetic algorithm for robotic assembly line balancing. Eur. J. Oper. Res. 168, 811–825.

Martello, S., Soumis, F., Toth, P., 1997. Exact and approximation algorithms for makespan minimization on unrelated parallel machines. Discrete Appl. Math. 75, 169–188.

Mausser, H., Laguna, M., 1999. Minimising the maximum relative regret for linear programmes with interval objective function coefficients. J. Oper. Res. Soc. 50, 1063–1070.

Miralles, C., García-Sabater, J., Andrés, C., Cardos, M., 2007. Advantages of assembly lines in sheltered work centres for the dsabled: a case study. Int. J. Prod. Econ. 110, 187–197.

Miralles, C., García-Sabater, J.P., Andrés, C., Cardos, M., 2008. Branch and bound procedures for solving the assembly line worker assignment and balancing problem: application to sheltered work centres for disabled. Discrete Appl. Math. 156, 352–367.

Montemanni, R., Marta, J., Mastrolilli, M., Gambardella, L., 2007. The robust traveling salesman problem with interval data. Transp. Sci. 41, 366–381.

Moreira, M., Cordeau, J.-F., Costa, A., Laporte, G., 2015. Robust assembly line balancing with heterogenous workers. Comput. Ind. Eng. 88, 254–263.

Moreira, M., Miralles, C., Costa, A., 2015. Models and heuristics for the assembly line worker integration and balancing problem. Comput. Oper. Res. 54, 64–73.

Moreira, M., Ritt, M., Costa, A., Chaves, A., 2012. Simple heuristics for the assembly line worker assignment and balancing problem. J. Heuristics 18, 505–524.

Mutlu, O., Polat, O., Supciller, A., 2013. An iterative genetic algorithm for the assembly line worker assignment and balancing problem of type-ii. Comput. Oper. Res. 40, 418–426.

Oksuz, M., Buyukozkan, K., Satoglu, S., 2017. U-shaped assembly line worker assignment and balancing problem: a mathematical model and two meta-heuristics. Comput. Ind. Eng. 112, 246–263. doi:10.1016/j.cie.2017.08.030.

Pereira, J., 2016. Procedures for the bin packing problem with precedence constraints. Eur. J. Oper. Res. 250, 794–806. doi:10.1016/j.ejor.2015.10.048.

Pereira, J., 2016. The robust (minmax regret) single machine scheduling with interval processing times and total weighted completion time objective. Comput. Oper. Res. 66, 141–152.

Pereira, J., Averbakh, I., 2011. Exact and heuristic algorithms for the interval data robust assignment problem. Comput. Oper. Res. 38, 1153–1163.

Pereira, J., Averbakh, I., 2013. The robust set covering problem with interval data. Ann. Oper. Res. 207, 217–235.

Pereira, J., Álvarez-Miranda, E., 2017. An exact approach for the robust assembly line balancing problem. Omega doi:10.1016/j.omega.2017.08.020. To appear

Polat, O., Kalayci, C., Mutlu, O., Gupta, S., 2016. A two-phase variable neighbourhood search algorithm for assembly line workerassignment and balancing problem type-ii: an industrial case study. Int. J. Prod. Res. 54, 722–741. doi:10.1080/00207543.2015.1055344.

Rubinovitz, J., Bukchin, J., 1993. Ralb - a heuristic algorithm for design and balancing of robotic assembly lines. Ann. CIRP 42, 497–500.

Sarin, S., Erel, E., Dar-El, E., 1999. A methodology for solving single-model, stochastic assembly line balancing problem. Omega 27 (5), 525–535. doi:10.1016/S0305-0483(99)00016-X.

Scholl, A., Becker, C., 2006. State-of-the-art exact and heuristic solution procedures for simple assembly line balancing. Eur. J. Oper. Res. 168, 666–693.

Scholl, A., Klein, R., 1997. SALOME: A bidirectional branch-and-bound procedure for assembly line balancing. INFORMS J. Comput. 9, 319–334.

Sewell, E.C., Jacobson, S.H., 2012. A Branch, Bound, and Remember algorithm for the simple assembly line balancing problem. INFORMS J. Comput. 24, 433–442. doi:10.1287/ijoc.1110.0462.

Sotskov, Y., Dolgui, A., Portmann, M., 2006. Stability analysis of an optimal balance for an assembly line with fixed cycle time. Eur. J. Oper. Res. 168 (3), 783–797.

Vilà, M., Pereira, J., 2014. A branch-and-bound algorithm for assembly line worker assignment and balancing problems. Comput. Oper. Res. 44, 105–114.