Discrete Applied Mathematics 208 (2016) 123–136

Contents lists available at ScienceDirect

Discrete Applied Mathematics

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

Maximizing the robustness for simple assembly lines with fixed cycle time and limited number of workstations

André Rossi [a,*], Evgeny Gurevsky [b], Olga Battaïa [c], Alexandre Dolgui [d]

[a] LERIA, Université d'Angers, France
[b] IRCCyN, Université de Nantes, France
[c] Institut Supérieur de l'Aéronautique et de l'Espace, Toulouse, France
[d] IRCCyN, École Nationale Supérieure des Mines de Nantes, France

ARTICLE INFO

Article history:
Received 14 January 2015
Received in revised form 13 February 2016
Accepted 14 March 2016
Available online 11 April 2016

Keywords:
Assembly line design/balancing
Stability radius
Sensitivity
Robustness
Task time variability
MILP

ABSTRACT

This paper deals with an optimization problem that arises when a new paced simple assembly line has to be designed subject to a limited number of available workstations, cycle time constraint, and precedence relations between necessary assembly tasks. The studied problem, referred to as SALPB-S, consists in assigning the set of tasks to workstations so as to find the most robust line configuration (or solution) under task time variability. The robustness of solution is measured via its stability radius, i.e., as the maximal amplitude of deviations for task time nominal values that do not violate the solution feasibility. In this work, the concept of stability radius is considered for two well-known norms: ℓ_1 and ℓ_∞. For each norm, the problem is proven to be strongly $\mathcal{NP}$-hard and a mixed-integer linear program (MILP) is proposed for addressing it. To accelerate the seeking of optimal solutions, an upper bound on the stability radius is devised and integrated into the corresponding MILP. Computational results are reported on a collection of instances derived from classic benchmark data used in the literature for the Simple Assembly Line Balancing Problem.

1. Introduction

A simple assembly line is a typical flow-oriented manufacturing system (see, for example, [22,6]), which is used to fabricate a large quantity of a single type of product. It can be viewed as a set of linearly ordered workstations linked by a conveyor belt moving the product units. During manufacturing, the units pass through the workstations in the order of their location. Thus, they are sequentially injected at the beginning of the line, are transferred from one workstation to another, and are outputted at the end of the line. The workstations operate simultaneously. At each of them, its own set of tasks is repetitively carried out on the successive units.

In addition to the above, functioning simple assembly lines have also the following quite natural characteristics (see [3]):

- only one unit can be processed simultaneously at the workstation and only one workstation at a time can handle the unit;
- the tasks of any workstation are performed sequentially one by one without splitting;

* Corresponding author.
E-mail addresses: andre.rossi@univ-angers.fr (A. Rossi), evgeny.gurevsky@univ-nantes.fr (E. Gurevsky), olga.battaia@isae.fr (O. Battaïa), alexandre.dolgui@mines-nantes.fr (A. Dolgui).

http://dx.doi.org/10.1016/j.dam.2016.03.005
0166-218X/

- there is neither buffer stock nor parallel workstation and, as a consequence, the transfer of all the units situated on the line is implemented in a synchronized manner, i.e., all units are moved from their current workstations to the next ones simultaneously.

The design of such lines is an important problem, since it generally involves significant investments. This stage includes several important issues, one of which is named as *balancing problem*. In general, it consists in a partition of the set of necessary assembly tasks among workstations in an optimal way with respect to a given production goal. Mostly, supplementary restrictions can also be taken into consideration for this problem. For instance, some tasks are usually not executed in an arbitrary manner, but are subject to *precedence constraints*. The representation of these constraints is often done by a directed acyclic graph, where the set of nodes corresponds to the set of tasks and the arcs introduce a partial order over them. Thus, an arc (i, j) means that the task j cannot start before the task i is completed. The synchronized manner of the units transportation enforces that the total working time (or load) of any workstation is not greater than a certain given value determining a production rate of the line. Such a value is referred to as *cycle time*[1] and the corresponding constraint to as *cycle time constraint*. Finally, limitations of the available space for assembling may be naturally translated into restraints on the maximal number of workstations to be installed.

With regard to the objectives used, simple assembly line balancing problems (SALBP) are commonly classified into the following types (see, e.g., [22,2]): minimize the number of used workstations for a fixed cycle time (SALBP-1); minimize the cycle time with a given number of workstations (SALBP-2); and if neither the number of workstations nor the line cycle time is fixed, maximize the line efficiency (SALBP-E). The latter problem seeks a line configuration that minimizes the following expression: the number of used workstations multiplied by the working time on the most loaded one. For these problems, known to be $\mathcal{NP}$-hard (see [22], Chapter 2.2.1.5), a great number of exact and heuristic methods have been developed. Their comprehensive surveys can be found in [20,23,2].

Despite of all the attention given to SALBP, its classic formulation remains quite general and does not always reflect particular real-world situations in manufacturing. Frequently, more specific assumptions have to be taken into account. Thus, for instance, one of the important subjects to be considered is the task time variability. Indeed, as mentioned in [2], task times are often not exactly known at the preliminary design stage of the line and only their nominal (or estimated) values are used. This is caused by the following practical factors:

- for manual assembly lines, the performance of operators, implementing tasks, depends on their work rate, skill level, fatigue and motivation;
- product specifications as well as workstation characteristics may be changed during the line life cycle. It can be reasoned by a customer demand or updating the market of materials;
- various delays and micro-stoppages when tasks are executed.

Any of these events may occur in any moment of the line exploitation and can cause a costly line interruption if the cycle time is exceeded. As a consequence, to construct a robust line configuration for a long term usage, the task time variability should be anticipated at the line balancing stage. In what follows, we present an overview of the existing approaches dedicated to these aspects.

The choice of an appropriate approach for handling the processing time of tasks strongly depends on the available information dealing with its uncertainty. Thus, among the ways used in the literature, we can distinguish the following ones: *stochastic*, *fuzzy* and *robust approaches*.

For the *stochastic approach*, task processing times are represented as independent random variables with known probability distributions. As a consequence, for grouping the tasks into workstations, a particular technique supervising the cycle time constraint has to be used. Among the works dealing with this topic, those applying the so-called *chance-constrained method* are usually cited. This method consists in assigning the tasks so as to ensure that the probability of respecting the cycle time is greater, for each workstation, than a given value named as *confidence level*. For instance, such method was applied in [29,1] for a U-type assembly line balancing and in [18] for a two-sided assembly line design. In these articles, the authors use an integer linear programming (ILP) formulation of the corresponding problem that integrates the probabilistic cycle time constraint. Based on the information expressing the task times, they introduce new supplementary variables and use various linearization techniques in order to obtain again an equivalent deterministic ILP formulation for the probabilistic problem studied.

Concerning the *fuzzy approach*, the potential task processing time values are represented as a fuzzy set whose membership function describes their possibility distribution. Similarly to the stochastic case, for assigning such tasks to workstations, controlling the cycle time constraint is needed. To do this, a suitable fuzzy arithmetic and an appropriate method dedicated to comparing these fuzzy sets have to be introduced. An application of tasks with fuzzy times was presented in [28,9] for SALBP-1, in [13] for a mixed-model line balancing and in [30] for a bi-objective variant of SALBP-2.

However, it should be noted that the use of these two approaches in practice could be a difficult challenge. This is due to the fact that the available knowledge on the input data is not always sufficient to derive adequate probability or possibility

[1] In this paper, we suppose that the cycle time can be greater than or equal to the working time of the most loaded workstation.

distributions for all task processing times, especially if the design of the assembly line is planned for the first time. *Robust approaches* (see [15]) are often more relevant in such situations, since they assume that only a discrete set of scenarios and/or intervals of potential task time realizations are known without any distribution or even only a set of tasks whose processing time may vary is given as input.

The use of the *discrete* or *interval* representation of scenarios usually modifies the goal of a problem considered. Indeed, an optimal solution found for one scenario can lose its optimality and even its feasibility for another one. To get around this situation, a criterion named as *absolute robustness* or *min–max* can be applied. Widely used in robust optimization, it aims to seek a solution remaining feasible for all scenarios and having the best performance for the worst of them. For instance, this criterion was studied in [12] for SALBP-1 with interval processing times. To find the solution mentioned above, the authors develop a branch-and-bound algorithm. A similar approach was applied in [5] for SALBP-2, but with a discrete set of scenarios. In the latter work, the computational complexity of seeking the robust solution was presented for different types of precedence constraints.

The case where only a *set of uncertain tasks* (with variable processing times) can be identified without any additional information is less informative, but probably the most frequent in practice. Because of the lack of information, the methods used for the approaches referred earlier are not applicable. To evaluate a solution in such a situation, [25,24], studying SALBP-1 and SALBP-2, have suggested a specific indicator, called as *stability radius*. Given a solution, it is calculated as the maximal amplitude of the deviations of the uncertain task times from their nominal values for which the solution feasibility (or optimality) remains respected. The authors show that this indicator can serve as an appropriate robustness measure. Indeed, the greater its value for a solution studied, the greater the robustness of the line configuration engaged. Moreover, it was proven that computing the stability radius in the sense of feasibility is a polynomial problem for any admissible solution of SALBP-1. These positive outcomes have inspired several other studies. Thus, in [10], the results obtained for SALBP-1 and SALBP-2 were generalized for SALBP-E. A robust version for a more complex assembly line balancing problem subject to task time uncertainty has been also proposed in [11]. For that problem, the stability radius in the sense of feasibility was considered as the second objective to be maximized. To find a trade-off between its value and the cost of the line, an ϵ-constraint based heuristic approach has been developed for seeking a Pareto set approximation for such bi-objective optimization problem. It is important to mention that the concept of stability radius has been also studied for various scheduling problems in [27], different combinatorial optimization problems in [26,16] and multi-objective integer linear programming problems in [7].

In this paper, we continue to study the stability radius within the framework of assembly line design subject to task times uncertainty. For the case considered, we address an assembly line design with a fixed cycle time, a limited number of workstations and precedence constraints. This new optimization problem is referred to as SALBP-S, where S stands for stability radius maximization. Namely, compared with the previous works, instead of calculating the stability radius for a given line configuration, we seek a feasible one with the greatest stability radius value. To evaluate the stability radius, two norms are used in this paper: ℓ_1 and ℓ_∞. For each norm, the corresponding problem is proven to be strongly $\mathcal{NP}$-hard and a mixed-integer linear program (MILP) is developed and tested on a set of numerical instances.

The rest of the paper is organized as follows. Basic definitions, properties and illustrative examples are presented in Section 2. In Section 3, the $\mathcal{NP}$-hardness as well as upper bounds on the stability radius of the problems considered is established. Section 4 describes two MILP formulations for both norms studied. Computational results constitute Section 5. Final remarks and conclusions are given in Section 6.

2. Basic definitions and properties

In order to simplify the further statement, we introduce below some notations related to SALBP-S:

- $V = \{1, 2, \ldots, n\}$ is the set of necessary assembly tasks;
- $W = \{1, 2, \ldots, m\}$ is the set of available workstations;
- t_j is a nominal processing time of the task $j \in V$;
- T is the cycle time;
- $G = (V, A)$ is a directed acyclic graph representing the precedence constraints, where A is the set of arcs.

Thus, an allocation of the set of tasks to the set of workstations is called a feasible solution if each task is assigned to exactly one workstation (1)–(2) such that the precedence and cycle time constraints are respectively satisfied (3)–(4):

$$V = \bigcup_{k \in W} V_k^s, \tag{1}$$

$$V_p^s \cap V_q^s = \emptyset, \quad \forall p, q \in W, \text{ where } p \neq q, \tag{2}$$

$$(i, j) \in A \Rightarrow k \leq l, \quad \text{where } i \in V_k^s, j \in V_l^s, \tag{3}$$

$$\sum_{j \in V_k^s} t_j \leq T, \quad \forall k \in W. \tag{4}$$

Here V_k^s is a set of tasks assigned to the workstation k of the solution s.

It is supposed here that there exist two sets of *uncertain tasks*: a non-empty set $\widetilde{V}^1$ ($\widetilde{V}^1 \subseteq V$) of *a priori uncertain tasks* whose processing time may deviate from its nominal value with regard to time without any additional information and a set $\widetilde{V}^2$ ($\widetilde{V}^2 \subseteq V$) of *a posteriori uncertain tasks* whose uncertainty is caused by a set $\widehat{W}$ ($\widehat{W} \subseteq W$) of *uncertain workstations*. These workstations are such that any task allocated to them becomes uncertain (even if it belongs to $V \setminus \widetilde{V}^1$). Hereinafter, the set of all uncertain tasks is denoted as $\widetilde{V} = \widetilde{V}^1 \cup \widetilde{V}^2$ and any workstation from $W \setminus \widehat{W}$ is called *certain*. The presence of certain and/or uncertain workstations can be explained by the existence of assembly lines having simultaneously two types of workstations: workstations with automatic tasks executing by robots or machines and workplaces where tasks are operated in a manual manner, respectively.

To evaluate the robustness of a feasible solution, we use the concept of *stability radius* whose formal definition requires some supplementary notations:

- $t = (t_1, t_2, \ldots, t_n) \in \mathbb{R}^n_{\geq}$ is a vector expressing the nominal task times;
- $\Xi = \{\xi \in \mathbb{R}^n_{\geq} \mid \xi_j = 0, j \in V \setminus \widetilde{V}\}$ is the set of vectors where each of which presents possible processing time deviations (or variations) for the uncertain tasks;
- $F(t)$ is the set of feasible solutions with respect to a given vector $t \in \mathbb{R}^n_{\geq}$.

Here, $\mathbb{R}_{\geq}$ is the set of non-negative real numbers.

Remark 1. Since any decrease of task processing time cannot compromise the solution feasibility, it is sufficient to consider only non-negative task time deviations in this work, i.e., for any $j \in \widetilde{V}$ we have $\xi_j \in \mathbb{R}_{\geq}$.

Thus, the stability radius of a feasible solution $s \in F(t)$ can be defined as follows (see [24]):

$$\rho(s, t) = \max\{\epsilon \geq 0 | \forall \xi \in B(\epsilon)\ (s \in F(t + \xi))\}, \tag{5}$$

where $B(\epsilon) = \{\xi \in \Xi | \|\xi\| \leq \epsilon\}$.

In other words, $\rho(s, t)$ is determined as the value of the radius of the greatest closed ball $B(\cdot)$, called *stability ball*, representing the deviations of the uncertain task nominal processing times, for which s remains feasible. Any element ξ of $B(\cdot)$ is evaluated based on a given norm $\|\cdot\|$ defining the distance between vectors t and $t + \xi$ (or the amplitude of deviations from t).

In this paper, two norms ℓ_1 ($\|\cdot\|_1$) and ℓ_∞ ($\|\cdot\|_\infty$) are studied in detail, where by definition $\|\xi\|_1 = \sum_{j \in \widetilde{V}} \xi_j$ and $\|\xi\|_\infty = \max_{j \in \widetilde{V}} \xi_j$. As a consequence, the notations $\rho_1(\cdot, \cdot), B_1(\cdot)$ and $\rho_\infty(\cdot, \cdot), B_\infty(\cdot)$ will be used for ℓ_1 and ℓ_∞, respectively.

The following useful property and lemma are direct corollaries from the definition of stability radius.

Property 1. *For any solution $s \in F(t)$ and $\xi \in \Xi$ such that $\xi_j = \rho_\infty(s, t), j \in \widetilde{V}$ we have $s \in F(t + \xi)$.*

Lemma 1. *The inequality $\rho_\infty(s, t) \leq \rho_1(s, t)$ holds for any $s \in F(t)$.*

Proof. Let $s \in F(t)$ for some $t \in \mathbb{R}^n_{\geq}$. Then, taking into account the definition of $\rho_\infty(s, t)$, we obtain the following:

$$\forall \xi \in B_\infty(\rho_\infty(s, t)) \quad (s \in F(t + \xi)).$$

Hence, based on the evident inclusion $B_1(\epsilon) \subseteq B_\infty(\epsilon)$ that is valid for any $\epsilon \geq 0$, we have

$$\forall \xi \in B_1(\rho_\infty(s, t)) \quad (s \in F(t + \xi)).$$

Consequently, due to the definition of $\rho_1(s, t)$, the necessary inequality is proven. □

Below, we provide an illustrative example of the interpretation of the stability radius in the ℓ_∞- and ℓ_1-norms. The following problem instance is considered: $n = 6, m = 2, \widetilde{V} = \{1, 2\}, t = (1, 1, 1, 1, 1, 1), T = 4$ and $\widehat{W} = \emptyset$. There is no precedence constraint.

Two feasible solutions s' and s'' are shown in Figs. 1 and 2, respectively.

It is easy to see that *any* increase within the limit of one time unit of the processing time of *any* uncertain task does not compromise the feasibility of s' and s'' (see Fig. 3), i.e., for any $\xi \in B_\infty(1)$ we have $s' \in F(t + \xi)$ and $s'' \in F(t + \xi)$.

At the same time, even for a small excess ($\delta > 0$) of this limit, the solution feasibility can be violated. Thus, for instance, s' and s'' do not belong to $F(t+\xi^*)$, where $\xi^* := (1+\delta, 1, 0, 0, 0, 0) \in B_\infty(1+\delta)$. Hence, we obtain $\rho_\infty(s', t) = \rho_\infty(s'', t) = 1$.

Simultaneously, it can be observed that *any total* increase, i.e. the sum of the processing time deviations, within the limit of one time unit (resp. two time units), of *all* uncertain tasks does not disturb the feasibility of s' (resp. s''), i.e. (see Figs. 4 and 5), for any $\xi \in B_1(1)$ we have $s' \in F(t + \xi)$ (resp. for any $\xi \in B_1(2)$ we have $s'' \in F(t + \xi)$).

However, any small exceeding of these limits can affect the solution feasibility. For example, $s' \notin F(t + \xi')$ and $s'' \notin F(t + \xi'')$, where $\xi' := (1 + \delta, 0, 0, 0, 0, 0) \in B_1(1 + \delta)$ and $\xi'' := (1 + \delta, 1, 0, 0, 0, 0) \in B_1(2 + \delta)$. In other words, we obtain $\rho_1(s', t) = 1$ and $\rho_1(s'', t) = 2$.

This example demonstrates that the study of the stability radius in the ℓ_1- and ℓ_∞-norms at the same time may provide an interesting information concerning the robustness of solutions in order to differentiate them.

The next two theorems confirm the presented above example and show how to calculate the exact value of the stability radius for a given feasible solution for two introduced norms. They prove that the stability radius in the ℓ_1-norm is equal to the minimum idle time among the workstations containing uncertain tasks. While for the ℓ_∞-norm, it needs to seek the workstation that provides the minimum value of the idle time divided by the number of its uncertain tasks.

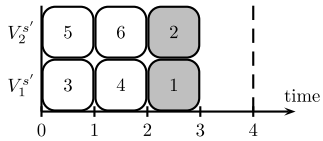

Fig. 1. Solution s'.

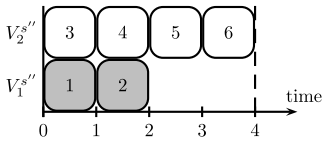

Fig. 2. Solution s''.

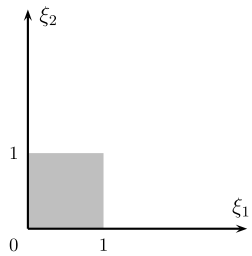

Fig. 3. Projection of $B_\infty(1)$ on $\mathbb{R}^2$ for s' and s''.

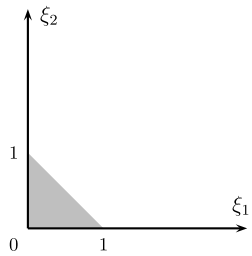

Fig. 4. Projection of $B_1(1)$ on $\mathbb{R}^2$ for s'.

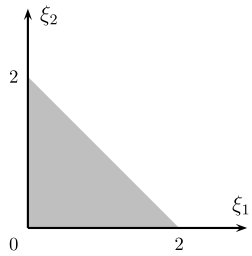

Fig. 5. Projection of $B_1(2)$ on $\mathbb{R}^2$ for s''.

Theorem 1. *The stability radius $\rho_1(s,t)$ for $s \in F(t)$ is calculated as follows*

$$\rho_1(s,t) = \min_{k \in \widetilde{W}^s} \left(T - \sum_{j \in V_k^s} t_j \right), \tag{6}$$

where $\widetilde{W}^s = \{k \in W \mid \widetilde{V}_k^s \neq \emptyset\}$, $\widetilde{V}_k^s = V_k^s \cap \widetilde{V}$.

Proof. To simplify the further statement, the following notation is introduced: ρ_1 and φ_1 are the left-hand and the right-hand sides of (6), respectively. To prove (6), we consequently show that the inequalities $\rho_1 \geq \varphi_1$ and $\rho_1 \leq \varphi_1$ hold.

First, let us prove that $\rho_1 \geq \varphi_1$. To do this, it is sufficient to check the following

$$\forall \xi \in B_1(\varphi_1) \quad (s \in F(t+\xi)). \tag{7}$$

If $\varphi_1 = 0$, the inequality $\rho_1 \geq \varphi_1$ is evident. Let $\varphi_1 > 0$ and $\xi \in B_1(\varphi_1)$. Then, by the definition of φ_1, for any $k \in \widetilde{W}^s$ we have $\varphi_1 \leq T - \sum_{j \in V_k^s} t_j$.

Hence, due to the definition of $B_1(\varphi_1)$, we obtain

$$\begin{aligned} T - \sum_{j \in V_k^s}(t_j + \xi_j) &= T - \left(\sum_{j \in V_k^s} t_j + \sum_{j \in \widetilde{V}_k^s} \xi_j \right) \\ &\geq T - \left(\sum_{j \in V_k^s} t_j + \varphi_1 \right) \geq 0, \quad k \in \widetilde{W}^s. \end{aligned}$$

Therefore, taking into account the following evident inequalities

$$T - \sum_{j \in V_k^s}(t_j + \xi_j) \geq 0, \quad k \in W \setminus \widetilde{W}^s,$$

we conclude that $s \in F(t+\xi)$, i.e., formula (7) holds.

Now let us show that $\rho_1 \leq \varphi_1$. The latter inequality is equivalent to the following formula

$$\forall \delta > \varphi_1 \quad \exists \xi^* \in B_1(\delta) \quad (s \notin F(t+\xi^*)). \tag{8}$$

To prove (8), the definition of φ_1 is used. By the definition of φ_1, there exists $k^* \in \widetilde{W}^s$ such that $\varphi_1 = T - \sum_{j \in V_{k^*}^s} t_j$. Then, setting $j^* \in \widetilde{V}_{k^*}^s$ and $\xi^* \in B_1(\delta)$, where $\xi_j^* = \delta$, if $j = j^*$ and $\xi_j^* = 0$ otherwise, we obtain

$$T - \sum_{j \in V_{k^*}^s}(t_j + \xi_j^*) = T - \left(\sum_{j \in V_{k^*}^s} t_j + \delta \right) = \varphi_1 - \delta < 0.$$

In other words, $s \notin F(t+\xi^*)$ and therefore (8) holds. □

Theorem 2 is a result obtained in the work of [24] and can be proven in the same way as Theorem 1.

Theorem 2. *The stability radius $\rho_\infty(s,t)$ for $s \in F(t)$ is calculated as follows*

$$\rho_\infty(s,t) = \min_{k \in \widetilde{W}^s} \frac{T - \sum_{j \in V_k^s} t_j}{|\widetilde{V}_k^s|}. \tag{9}$$

It is easy to see that formulas (6) and (9) can be calculated in polynomial time for any given feasible solution. For more details, we address the reader to the paper of [24], where such a linear complexity algorithm is presented for the ℓ_∞-norm with $\widehat{W} = \emptyset$ and which can be easily adapted for the studied case for both norms.

3. Stability radius maximization

In this section, we study the complexity of SALBP-S. Hereinafter, this problem is denoted as P_1 (resp. P_∞) for the ℓ_1-norm (resp. ℓ_∞-norm). At first, we establish their $\mathcal{NP}$-hardness and show that P_1 and P_∞ are not equivalent. At the end, respective upper bounds are provided for both problems.

3.1. Complexity

Here, we show that P_1 and P_∞ are strongly $\mathcal{NP}$-hard. For both problems, no precedence constraints are assumed to be given and $\widehat{W} = \emptyset$.

Lemma 2. *P_∞ and P_1 are strongly $\mathcal{NP}$-hard even if $V = \widetilde{V}$.*

Proof. To prove this theorem, we will use a reduction from the $\mathcal{NP}$-complete in the strong sense problem Bin-Packing (see [8]) to the decision version of P_∞ and P_1, denoted as $P_{1,\infty}$-Decision.

Bin-Packing: Given a set $B = \{1, 2, \ldots, q\}$ of q bins of size 1 and a set $I = \{1, 2, \ldots, k\}$ of k items with sizes $v_1, v_2, \ldots, v_k$ such that $v_i \in [0, 1]$, $i \in I$, does there exists a partition of I into q disjoint subsets (or bins) I_j, $j \in B$ such that $\sum_{l \in I_j} v_l \leq 1$ for any $j \in B$?

$P_{1,\infty}$-Decision: Given a set V of tasks, their processing times $t \in \mathbb{R}_{\geq}^{|V|}$, a number m of workstations, a cycle time T and a value ρ, does there exist a feasible allocation s of the tasks to the workstations such that $\rho_{1,\infty}(s, t) \geq \rho$?

Given an instance of Bin-Packing, we construct the following generic instance of $P_{1,\infty}$-Decision: $m = q$, $n = k$, $T = 1$, $\rho = 0$ and $t_j = v_j$ for each $j \in V$. Since, by the definition of stability radius, $\rho_{1,\infty}(s, t) \geq 0$, it is easy to see that for the considered instances, a solution of $P_{1,\infty}$-Decision exists if and only if there exists a solution of Bin-Packing. □

Here, we show that P_1 and P_∞ are not equivalent problems. Consider the following instance: $n = 5$, $m = 2$, $t = (1, 1, 1, 1, 4)$, $T = 7$, $\widehat{W} = \{1, 2\}$ and $\widetilde{V} = \emptyset$. Following Theorem 1, solution s' shown in Fig. 6 is optimal for P_1 for which $\rho_1(s', t) = 3$, while $\rho_\infty(s', t) = \min\left\{\frac{7-4}{1}, \frac{7-4}{4}\right\} = \frac{3}{4}$ due to Theorem 2. At the same time, solution s'' shown in Fig. 7 is optimal for P_∞ so as $\rho_\infty(s'', t) = \min\left\{\frac{7-5}{2}, \frac{7-3}{3}\right\} = 1$, whereas $\rho_1(s'', t) = 2$ due to Theorem 1.

3.2. Upper bounds

In this sub-section, we propose tight upper bounds for P_1 and P_∞. In order to simplify the further statement, we denote the optimal values of these problems as ρ_1 and ρ_∞, respectively.

3.2.1. An upper bound on ρ_1

First, it is easy to deduce that

$$\rho_1 \leq UB_1^1 := T - \max_{j \in \widetilde{V}^1} t_j.$$

Second, as the stability radius only depends on the load of the workstations that process uncertain tasks, theoretically its value is maximal when the tasks of $V \setminus \widetilde{V}^1$ are allocated to workstations from $W \setminus \widehat{W}$ without idle time and the rest is distributed between the remaining workstations as uniformly as possible. In the best case, the maximal number of the workstations that process certain tasks only and whose load is equal to T is $\min\left\{m - |\widehat{W}|, \chi\right\}$, where $\chi = \left\lfloor \frac{\sum_{j \in V \setminus \widetilde{V}^1} t_j}{T} \right\rfloor$.

At this point, two cases are possible:

1. $m - |\widehat{W}| \leq \chi$. Then only uncertain workstations are available for processing all the tasks that have not been allocated to workstations of $W \setminus \widehat{W}$. In that case,
$$\rho_1 \leq UB_1^2 := T - \frac{\sum_{j \in V} t_j - T \cdot (m - |\widehat{W}|)}{|\widehat{W}|}.$$
2. $m - |\widehat{W}| > \chi$. Then at least one certain workstation is available for processing a total load of $\sum_{j \in V \setminus \widetilde{V}^1} t_j - T \cdot \chi$ units of time remaining from $V \setminus \widetilde{V}^1$, plus some $\sum_{j \in \widetilde{V}^1} t_j$ units of time originating from a priori uncertain tasks. The present case is then split into two sub-cases:
   2.1. Either all $m - \chi$ remaining workstations share the remaining load, which leads to a minimum uniform load per workstation that is equal to
   $$\frac{\sum_{j \in V} t_j - T \cdot \chi}{m - \chi}.$$
   2.2. Or a certain workstation processes all the remaining tasks from $V \setminus \widetilde{V}^1$, letting $m - \chi - 1$ workstations sharing the load of all a priori uncertain tasks.

   Consequently,
   $$\rho_1 \leq UB_1^3 := T - \min\left\{\frac{\sum_{j \in V} t_j - T \cdot \chi}{m - \chi}, \max\left\{\frac{\sum_{j \in \widetilde{V}^1} t_j}{m - \chi - 1}, \Theta\left(m - \chi - 1, \widetilde{V}^1\right)\right\}\right\},$$
   where
   $$\Theta(p, U) = \max\left\{\sum_{i=0}^{k} t_{\tau_{k \cdot p + 1 - i}} | k = 1, \ldots, \left\lfloor \frac{|U| - 1}{p} \right\rfloor\right\},$$
   introduced in [14], determines a lower bound on the total working time of the most loaded workstation for the case of p workstations and the set U of tasks. Here, $\tau = (\tau_1, \tau_2, \ldots, \tau_{|U|})$ is a permutation of U with respect to the non-increasing order of their processing times.

So finally,

$$\rho_1 \leq UB_1 := \begin{cases} \min\{UB_1^1, UB_1^2\}, & \text{if } m - |\widehat{W}| \leq \chi, \\ \min\{UB_1^1, UB_1^3\} & \text{otherwise.} \end{cases} \tag{10}$$

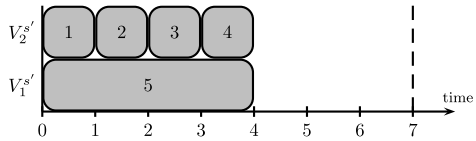

Fig. 6. s' is optimal for P_1.

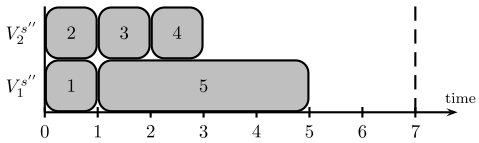

Fig. 7. s'' is optimal for P_∞.

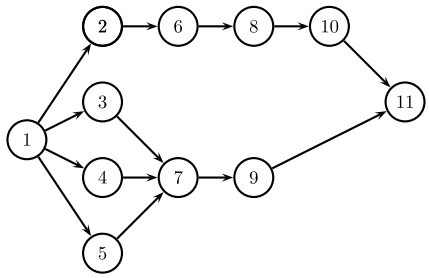

Fig. 8. Precedence constraints of the instance JACKSON.

3.2.2. An upper bound on ρ_∞

It can be observed that, whatever a feasible solution, the number of workstations which process at least $\gamma = \left\lceil \frac{|\widetilde{V}^1|}{m} \right\rceil$ a priori uncertain tasks belongs to $\mathcal{X}_\gamma = \left\{1, \ldots, \left\lfloor \frac{|\widetilde{V}^1|}{\gamma} \right\rfloor\right\}$. Hereinafter, the set of such workstations is denoted as W_γ.

Let us estimate the maximal value of the stability radius, denoted below as ρ_∞^k, among the feasible solutions for which $|W_\gamma| = k$. By definition, each workstation of $W \setminus W_\gamma$ processes at most $\gamma - 1$ a priori uncertain tasks. Thus, the total number of a priori uncertain tasks allocated to W_γ is at least

$$\zeta_k = \max\left\{\gamma k, |\widetilde{V}^1| - (\gamma - 1)(m - k)\right\}.$$

As a consequence, the load originating from the tasks of $\widetilde{V}^1$ is not less than $\sum_{j=1}^{\gamma} t_{\pi_j}$ for each workstation of W_γ and there exists at least one of them for which this load is at least $\sum_{j=1}^{\zeta_k} t_{\pi_j}/k$, where $\pi = (\pi_1, \pi_2, \ldots, \pi_{|\widetilde{V}^1|})$ is a permutation of $\widetilde{V}^1$ with respect to the non-decreasing order of their processing times. However, if the remaining total load $\sum_{j \in V \setminus \Pi_{\zeta_k}} t_j$ exceeds the remaining total capacity $T \cdot (m - k)$, then the surplus, formed in such a manner, has to be redistributed to the workstations of W_γ. Consequently, there exists at least one workstation in W_γ^s that has to process a supplementary load originating from the tasks of $V \setminus \Pi_{\zeta_k}$ that amounts to at least Δ_k units of time, with

$$\Delta_k = \frac{\left[\sum_{j \in V \setminus \Pi_{\zeta_k}} t_j - T \cdot (m - k)\right]^+}{k},$$

where $\Pi_{\zeta_k} = \{\pi_1, \pi_2, \ldots, \pi_{\zeta_k}\}$ and $[x]^+ = \max\{0, x\}, x \in \mathbb{R}$.

So, we have

$$\rho_\infty^k \le UB_\infty^k := \min\left\{\frac{T - \sum_{j=1}^{\zeta_k} \frac{t_{\pi_j}}{k}}{\gamma}, \frac{T - \left(\sum_{j=1}^{\gamma} t_{\pi_j} + \Delta_k\right)}{\gamma}\right\}.$$

Finally, taking into account Lemma 1, we obtain

$$\rho_\infty \le UB_\infty := \min\left\{UB_1, \max_{k \in \mathcal{X}_\gamma} UB_\infty^k\right\}.$$

4. MILP formulations for P_1 and P_∞

Here, we present two MILP formulations: one for P_1 and another for P_∞.

4.1. MILP formulation for p_1

P_1 is formulated as a mixed integer linear program on the following decision variables: ρ_1 is the stability radius value to maximize, $x_{j,k}$ is a binary variable that is set to one if and only if the task j is allocated to the workstation k, and a_k is a nonnegative variable that is positive if the workstation k has at least one uncertain task, or if the workstation k is uncertain. The central idea of the MILP formulation for P_1 consists in considering ρ_1 as the minimum idle time of the workstations that process uncertain tasks (see Theorem 1).

$$\text{Maximize} \quad \rho_1$$

$$\sum_{k\in W} x_{j,k} = 1 \qquad \forall j \in V \tag{11}$$

$$\sum_{j\in V} t_j x_{j,k} \le T \qquad \forall k \in W \tag{12}$$

$$\sum_{k\in W} k x_{i,k} \le \sum_{k\in W} k x_{j,k} \qquad \forall (i,j) \in A \tag{13}$$

$$\sum_{q=k}^{m} x_{i,q} \le \sum_{q=k}^{m} x_{j,q} \qquad \forall (i,j) \in A,\ \forall k \in W \setminus \{1\} \tag{14}$$

$$x_{j,k} \le a_k \qquad \forall j \in \widetilde{V}^1,\ \forall k \in W \setminus \widehat{W} \tag{15}$$

$$a_k = 1 \qquad \forall k \in \widehat{W} \tag{16}$$

$$\rho_1 \le T(2 - a_k) - \sum_{j\in V} t_j x_{j,k} \qquad \forall k \in W \tag{17}$$

$$x_{j,k} = 0 \qquad \forall j \in V,\ \forall k \notin Q(j) \tag{18}$$

$$\rho_1 \ge 0$$

$$a_k \ge 0 \qquad \forall k \in W$$

$$x_{j,k} \in \{0, 1\} \qquad \forall j \in V,\ \forall k \in W.$$

Constraints (11) ensure that each task is allocated to exactly one workstation. As for constraints (12), they state that the load of any workstation does not exceed the cycle time. The precedence constraints are expressed by inequalities (13) that are reinforced by those of (14). The latter inequalities are shown to be the most efficient ones for the simple assembly line balancing problem (see [21]). Constraints (15)–(17) describe the result obtained in Theorem 1. Indeed, it is easy to see that (15)–(17) imply that $a_k \in \{0, 1\}$. As a consequence, if k is a certain workstation without uncertain tasks, then $a_k = 0$ and (12) together with (17) yields $\rho_1 \le T$, which is always valid. Otherwise, when $a_k = 1$, (17) is a corollary of (6). Constraints (18) induce that the task j can only be allocated to a restricted set of workstations denoted by the interval $Q(j)$, where (see [19])

$$Q(j) = \left[\left\lceil \frac{t_j + \sum_{i\in\mathcal{P}(j)} t_i}{T} \right\rceil, m + 1 - \left\lceil \frac{t_j + \sum_{i\in\mathcal{S}(j)} t_i}{T} \right\rceil \right].$$

Here, $\mathcal{P}(j)$ (resp. $\mathcal{S}(j)$) is a set of all predecessors (resp. all successors) of j in the graph G representing the precedence constraints. Finally, (10) is a helpful valid inequality for addressing P_1 with a solver.

4.2. MILP formulation for P_∞

P_∞ is formulated as a mixed integer linear program on the following decision variables: ρ_∞ is the stability radius value to maximize, $x_{j,k}$ is a binary variable that is set to one if and only if the task j is allocated to the workstation k, and $\xi_{j,k}$ is a processing time deviation of the task j on the workstation k. The main principle of the MILP formulation for P_∞ is focused on the fact that the processing time of all uncertain tasks can be increased by ρ_∞ without losing the feasibility for the optimal solution (see Property 1). However, it is recalled that, in practice, uncertain tasks can have different processing

time deviations.

$$\begin{aligned}
\text{Maximize} \quad & \rho_\infty \\
\sum_{k\in W} x_{j,k} = 1 \quad & \forall j \in V \qquad (19)\\
\xi_{j,k} \le UB_\infty x_{j,k} \quad & \forall j \in V,\ \forall k \in W \qquad (20)\\
\rho_\infty = \sum_{k\in W} \xi_{j,k} \quad & \forall j \in V \qquad (21)\\
\sum_{j\in V} t_j x_{j,k} + \sum_{j\in V} \xi_{j,k} \le T \quad & \forall k \in \widehat{W} \qquad (22)\\
\sum_{j\in V} t_j x_{j,k} + \sum_{j\in \widetilde{V}^1} \xi_{j,k} \le T \quad & \forall k \in W \setminus \widehat{W} \qquad (23)\\
\sum_{k\in W} k x_{i,k} \le \sum_{k\in W} k x_{j,k} \quad & \forall (i,j) \in A \qquad (24)\\
\sum_{q=k}^{m} x_{i,q} \le \sum_{q=k}^{m} x_{j,q} \quad & \forall (i,j) \in A,\ \forall k \in W \setminus \{1\} \qquad (25)\\
x_{j,k} = 0 \quad & \forall j \in V,\ \forall k \notin Q(j) \qquad (26)\\
\rho_\infty \ge 0 \quad & \\
\xi_{j,k} \ge 0 \quad & \forall j \in V,\ \forall k \in W\\
x_{j,k} \in \{0,1\} \quad & \forall j \in V,\ \forall k \in W.
\end{aligned}$$

Constraints (19) and (26) are same as (11) and (18), they provide that each task j is allocated to exactly one workstation from $Q(j)$. Based on Property 1, constraints (20) and (21) state that the processing time deviation ξ_j of any task j is set to ρ_∞, which in turn is not greater than UB_∞. This is also due to the fact that constraints (19) ensure that only one value $\xi_{j,k}$, $k \in W$ is non-zero for any fixed $j \in V$. As to constraints (22) and (23), they induce that the total load of each workstation does not exceed T, whatever possible processing task time deviations within the stability ball. Moreover, they also indicate that $\xi_{j,k}$ has no impact if k is a certain workstation and $j \in V \setminus \widetilde{V}^1$. Inequalities (24) and (25) are respectively identical to those of (13) and (14) and express the precedence constraints.

5. Computational results

A set of 25 instances has been used to test the upper bounds and mixed integer linear programming models of P_1 and P_∞. These instances can be found at http://alb.mansci.de/. Each of them has been enriched with the number of workstations $m = \left\lceil 1.2 \frac{\sum_{j\in V} t_j}{T} \right\rceil$ and the cycle time defined by $T = 1.5 \max_{j\in V} t_j$. If the cycle time value T and the number of workstations m were set to their optimal value for SALBP-2 and SALBP-1 respectively, the stability radius would be zero for most instances. Since such settings are inappropriate when processing times are bound to deviate, larger values for T and m have been set to let the problem have solutions with a nonzero stability radius. Furthermore, we suppose that $|\widetilde{V}^1| \in \{0, \lceil \frac{n}{4} \rceil, \lceil \frac{n}{2} \rceil, n\}$ and $|\widehat{W}| \in \{0, \lceil \frac{m}{4} \rceil, \lceil \frac{m}{2} \rceil, m\}$, where $\widetilde{V}^1$ (resp. $\widehat{W}$) is built by taking the first $|\widetilde{V}^1|$ (resp. $|\widehat{W}|$) elements of a random permutation of $\{1, \ldots, n\}$ (resp. $\{1, \ldots, m\}$) associated with each instance. All the permutations used are given in Appendix B. Only seven combinations of $\widetilde{V}^1$ and $\widehat{W}$ are considered, since the other ones do not bring a useful additional information. The detailed results are given in Appendix A. The tests were carried out on a computer disposing Intel Xeon 2.66 GHz and 8 GB RAM. Xpress-Mosel was used as a solver for addressing the mixed integer linear programming models proposed in this paper.

Tables A.1–A.7 report the stability radius found and the computational time required to solve to optimality all 25 instances that have been sorted by increasing number of binary decision variables (i.e., $n \times m$). Each table corresponds to a given proportion of uncertain tasks and uncertain workstations. All these tables are built as follows. The first column is the instance's name, followed by the number of tasks, the number of workstations, and the cycle time. Columns 5–8 are related to P_1, and the last four columns report the results for P_∞. More precisely, columns 5 and 9 give the stability radius value of the best solution found for P_1 and P_∞, respectively. The values, presented in bold type, correspond to the case where there is no uncertain task and all uncertain workstations are empty for the solution provided by MILP, which implies the infinite stability radius. Columns 6 and 10 provide the best upper bound on the stability radius returned by the solver for P_1 and P_∞, respectively. Columns 7 and 11 inform the upper bounds on the stability radius for P_1 and P_∞ introduced in Sections 3.2.1 and 3.2.2, respectively. Finally, columns 8 and 12 report the computational time in seconds for solving the instance to optimality within the limits of 900 s (15 min). Thus, if no optimal solution was found after 900 s, 'dnf' (did not finish) is displayed in the corresponding column. The last row of any table displays the number of instances solved to optimality, as well as the average computational time calculated over them for both P_1 and P_∞.

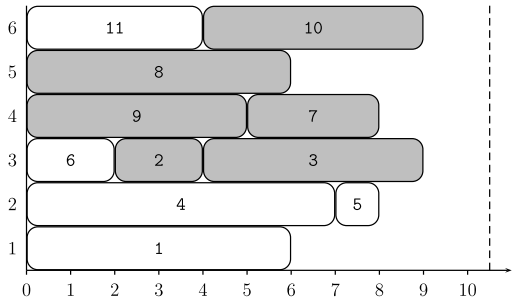

Fig. 9. $\rho_1 = 1.5$ for JACKSON with $\widetilde{V}^1 = \{2, 3, 7, 8, 9, 10\}$ and $\widehat{W} = \emptyset$.

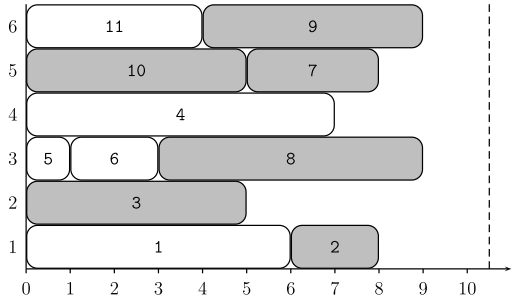

Fig. 10. $\rho_\infty = 1.25$ for JACKSON with $\widetilde{V}^1 = \{2, 3, 7, 8, 9, 10\}$ and $\widehat{W} = \emptyset$.

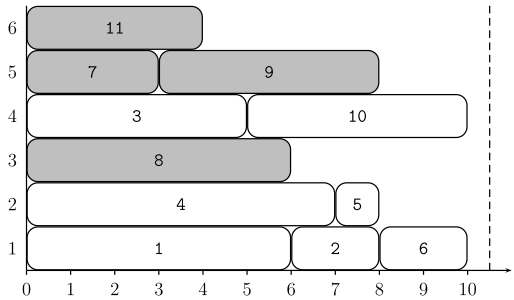

Fig. 11. $\rho_1 = 2.5$ for JACKSON with $\widetilde{V}^1 = \emptyset$ and $\widehat{W} = \{3, 5, 6\}$.

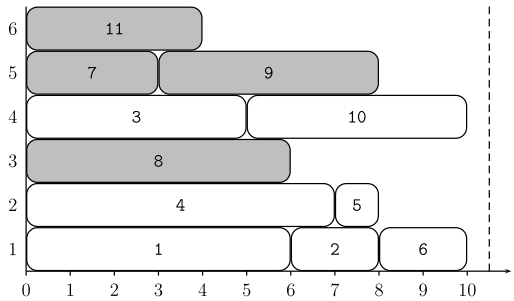

Fig. 12. $\rho_\infty = 1.25$ for JACKSON with $\widetilde{V}^1 = \emptyset$ and $\widehat{W} = \{3, 5, 6\}$.

For the sake of illustrating the results presented in these tables, the solutions returned for the instance JACKSON are shown in Figs. 9–14. That instance is defined by the precedence graph in Fig. 8.

For Table A.4, the uncertain tasks account for half of the total amount of tasks and there is no uncertain workstation. Fig. 9 shows an optimal solution for P_1, for which ρ_1 is set by workstations 3 and 6, as they both have the greatest load (9 units of time) among those disposing uncertain tasks. Since $T = 10.5$, then $\rho_1 = 1.5$ (see Theorem 1). Fig. 10 shows an optimal solution for P_∞, for which ρ_∞ is set by workstation 5 and equals 1.25 (see Theorem 2).

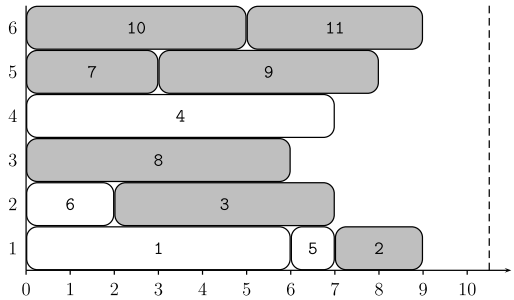

Fig. 13. $\rho_1 = 1.5$ for JACKSON with $\widetilde{V}^1 = \{2, 3, 7, 8, 9, 10\}$ and $\widehat{W} = \{3, 5, 6\}$.

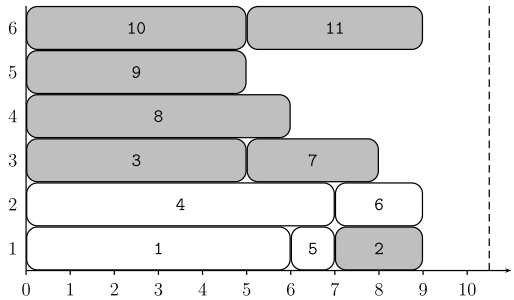

Fig. 14. $\rho_\infty = 0.75$ for JACKSON with $\widetilde{V}^1 = \{2, 3, 7, 8, 9, 10\}$ and $\widehat{W} = \{3, 5, 6\}$.

For Table A.5, $|\widetilde{V}^1| = 0$ and $|\widehat{W}| = \lceil \frac{m}{2} \rceil$. In Fig. 11, even though $\widetilde{V}^1$ is empty, the gray tasks are those allocated to uncertain workstations. The value for ρ_1 is set by workstation 5, as its load is maximum. Indeed, since $\widehat{W} = \{3, 5, 6\}$, the stability radius in the ℓ_1-norm is determined by the maximum load over these three workstations. The optimal solution for P_∞ shown in Fig. 12 happens to be exactly the same as for P_1 and workstation 5 is again responsible for the numerical value of ρ_∞.

In Table A.6, $|\widetilde{V}^1| = \lceil \frac{n}{2} \rceil$ and $|\widehat{W}| = \lceil \frac{m}{2} \rceil$. As can be seen in Fig. 13, the number of uncertain tasks is actually larger than $|\widetilde{V}^1|$ because any task allocated to an uncertain workstation becomes uncertain: that is the case of task 11. The numerical value for ρ_1 is set by workstations 1 and 6. The optimal solution of P_∞ shown in Fig. 14 is different from that in Fig. 13 but is equivalent to it. Indeed, both solutions are optimal for P_1 and P_∞ at the same time.

Table 1, which summarizes Tables A.1–A.7 shows that P_∞ is more difficult to solve than P_1. This is not surprising, since P_∞ has a greater number of variables and constraints than P_1. Furthermore, the difficulty of finding an optimal solution with the maximum stability radius increases with the amount of uncertainty, i.e., with $|\widetilde{V}^1|$ and $|\widehat{W}|$. This can be explained by the fact that more tasks and workstations are involved in the objective function value when $|\widetilde{V}^1|$ and $|\widehat{W}|$ increase. Indeed, a certain workstation that has no uncertain task has no impact on the stability radius.

The role of the upper bounds for addressing P_1 and P_∞ are quite different: UB_1 is more often equal to the maximum stability radius, especially for low values of $|\widetilde{V}^1|$ and $|\widehat{W}|$. However, even for large scale instances that are not solved to optimality for P_1, the solver may be unable to improve UB_1. Thus, for example, if the instance BARTHOL2 with $|\widetilde{V}^1| = \lceil \frac{n}{4} \rceil$ and $|\widehat{W}| = 0$ is addressed without UB_1, then the best upper bound found by the solver is 97.31 instead of 41.5 when UB_1 is used (see Table A.1). By contrast, the upper bound returned by the solver for P_∞ is often much better than UB_∞. Consequently, UB_∞ does not provide a significant advantage, but it can help anyway. For example, if the instance ARC83 is addressed without UB_∞ in the case where $|\widetilde{V}^1| = 0$ and $|\widehat{W}| = \lceil \frac{m}{4} \rceil$, then the best upper bound found by the solver is 4039.48 instead of 3355.72 when UB_∞ is used (see Table A.2).

For the sake of evaluating the impact of UB_1 and UB_∞ for difficult instances, a more systematic study is performed on the instance SCHOLL for which no optimal solution is found neither for P_1 nor for P_∞. Table 2 shows the best upper bound returned by Xpress-Mosel in 15 min with and without UB_1 and UB_∞ (for P_1 and P_∞, respectively). An asterisk denotes the cases where the solver is unable to improve the bound (UB_1 or UB_∞). As can be seen in Table 2, using UB_1 and UB_∞ does not always help, and can even be slightly detrimental, since the best upper bound is sometimes better without applying UB_1 and UB_∞. The main reason for such a counter-intuitive behavior might be *performance variability* (see [4]) affecting the solver. The complexity of the code of the numerous ingredients that are part of modern solvers may sometimes lead to unexpected negative interactions, as stated by Andrea Lodi in [17].

This observation is confirmed by the computational experiment performed with the instance GUNTHER, for which all optimal solutions have been found. It can be seen in Table 3 that enforcing the upper bound on the stability radius sometimes

Table 1
Number of optimal solutions found (out of 25 instances), and average CPU time (in seconds) required by Xpress-Mosel for finding an optimal solution.

	$\lvert\widetilde{V}^1\rvert = \lceil \frac{n}{4} \rceil$ $\lvert\widehat{W}\rvert = 0$	$\lvert\widetilde{V}^1\rvert = 0$ $\lvert\widehat{W}\rvert = \lceil \frac{m}{4} \rceil$	$\lvert\widetilde{V}^1\rvert = \lceil \frac{n}{4} \rceil$ $\lvert\widehat{W}\rvert = \lceil \frac{m}{4} \rceil$	$\lvert\widetilde{V}^1\rvert = \lceil \frac{m}{2} \rceil$ $\lvert\widehat{W}\rvert = 0$	$\lvert\widetilde{V}^1\rvert = 0$ $\lvert\widehat{W}\rvert = \lceil \frac{m}{2} \rceil$	$\lvert\widetilde{V}^1\rvert = \lceil \frac{n}{2} \rceil$ $\lvert\widehat{W}\rvert = \lceil \frac{m}{2} \rceil$	$\lvert\widetilde{V}^1\rvert = n$ $\lvert\widehat{W}\rvert = m$
# opt. sol to P_1	17	17	15	13	16	14	13
# opt. sol to P_∞	14	15	14	13	13	13	12
Avg. CPU time for P_1	0.65	6.36	0.90	8.49	0.33	4.53	5.02
Avg. CPU time for P_∞	29.54	25.94	64.47	52.48	22.83	21.10	41.26

Table 2
Impact of UB_1 and UB_∞ on the best upper bounds returned by Xpress-Mosel in 15 min for the instance SCHOLL.

	Bound used	$\lvert\widetilde{V}^1\rvert = \lceil \frac{n}{4} \rceil$ $\lvert\widehat{W}\rvert = 0$	$\lvert\widetilde{V}^1\rvert = 0$ $\lvert\widehat{W}\rvert = \lceil \frac{m}{4} \rceil$	$\lvert\widetilde{V}^1\rvert = \lceil \frac{n}{4} \rceil$ $\lvert\widehat{W}\rvert = \lceil \frac{m}{4} \rceil$	$\lvert\widetilde{V}^1\rvert = \lceil \frac{m}{2} \rceil$ $\lvert\widehat{W}\rvert = 0$	$\lvert\widetilde{V}^1\rvert = 0$ $\lvert\widehat{W}\rvert = \lceil \frac{m}{2} \rceil$	$\lvert\widetilde{V}^1\rvert = \lceil \frac{n}{2} \rceil$ $\lvert\widehat{W}\rvert = \lceil \frac{m}{2} \rceil$	$\lvert\widetilde{V}^1\rvert = n$ $\lvert\widehat{W}\rvert = m$
Best UB for ρ_1	None	1261.00	1416.00	1060.00	1635.45	742.00	742.10	380.00
	UB_1	984.13*	1416.73*	984.13*	623.36*	742.10*	623.36*	380.10 *
Best UB for ρ_∞	None	207.79	1903.03	207.79	104.59	1164.27	103.89	52.47
	UB_∞	207.79	1379.43	207.79	104.59	501.78	104.59	52.47

Table 3
Impact of UB_1 and UB_∞ on the CPU time (in seconds) required by Xpress-Mosel for finding an optimal solution for the instance GUNTHER.

	Bound used	$\lvert\widetilde{V}^1\rvert = \lceil \frac{n}{4} \rceil$ $\lvert\widehat{W}\rvert = 0$	$\lvert\widetilde{V}^1\rvert = 0$ $\lvert\widehat{W}\rvert = \lceil \frac{m}{4} \rceil$	$\lvert\widetilde{V}^1\rvert = \lceil \frac{n}{4} \rceil$ $\lvert\widehat{W}\rvert = \lceil \frac{m}{4} \rceil$	$\lvert\widetilde{V}^1\rvert = \lceil \frac{m}{2} \rceil$ $\lvert\widehat{W}\rvert = 0$	$\lvert\widetilde{V}^1\rvert = 0$ $\lvert\widehat{W}\rvert = \lceil \frac{m}{2} \rceil$	$\lvert\widetilde{V}^1\rvert = \lceil \frac{n}{2} \rceil$ $\lvert\widehat{W}\rvert = \lceil \frac{m}{2} \rceil$	$\lvert\widetilde{V}^1\rvert = n$ $\lvert\widehat{W}\rvert = m$
CPU time for P_1	None	1.09	0.30	6.27	55.27	0.16	27.14	1.51
	UB_1	0.94	0.08	5.38	25.41	0.44	7.18	22.78
CPU time for P_∞	None	188.60	1.01	10.51	5.93	45.16	73.40	208.10
	UB_∞	113.51	1.50	4.87	28.49	162.10	54.57	358.05

leads to a dramatic increase of the CPU time. More precisely, when $|\widetilde{V}^1| = n$ and $|\widehat{W}| = m$, we have $UB_1 = 11.7$ (see Table A.7), and using $\rho_1 \leq UB_1$ leads to increase the CPU time from 1.51 s (without upper bound) to 22.78 s. Replacing the last inequality by $\rho_1 \leq 12$ (which is obviously weaker) or by $\rho_1 \leq 11$ (which is stronger) leads to a CPU time of 1.19 s and 1.81 s, respectively. The fact that the solver finds the optimal solution at the root node (i.e., after calling a heuristic procedure) and spends the remaining time trying to prove the optimality status of this solution is also an indication that the solver used is subject to *performance variability* in that case.

6. Conclusion and perspectives

This paper deals with SALBP-S, which is a problem of robust balancing for simple paced assembly lines without buffer stock nor parallel workstations. It consists in finding a line configuration with the greatest stability radius subject to restricted number of workstations, fixed cycle time, precedence constraints, and task time variability. The stability radius is evaluated in both ℓ_1- and ℓ_∞-norms. For each norm, the corresponding problem, denoted respectively as P_1 and P_∞, was proven to be strongly $\mathcal{NP}$-hard. A MILP formulation as well as tight upper bounds was proposed for each problem. Numerical results show that the used commercial solver can find an optimal solution in less than 15 min for half of the instances.

The proposed MILP models are a first attempt to address the problems studied. The second natural step of our future research is a detailed analysis concerning the influence of different parameters on the value of the stability radius as well as the development of an efficient appropriate branch-and-bound procedure. Studying the stability radius as a robustness measure either for other configurations of assembly lines or within the framework of dynamic balancing is interesting as well. Another attractive task is investigating a new industrial optimization problem that can be called as "reverse robust balancing". This problem appears when we seek a simple assembly line configuration whose stability radius has to be greater than a given value enforced by a decision maker. In this situation, it is not always possible to find such a solution and the unique possibility to get around this difficulty is to use parallel workstations with duplicated tasks that require supplementary financial expenses. As a consequence, the aim of the reverse robust balancing problem is to find a line configuration with the desired value of stability radius, while minimizing the number of parallel workstations.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at http://dx.doi.org/10.1016/j.dam.2016.03.005.

References

[1] K. Ağpak, H. Gökçen, A chance-constrained approach to stochastic line balancing problem, European J. Oper. Res. 180 (3) (2007) 1098–1115.
[2] O. Battaïa, A. Dolgui, A taxonomy of line balancing problems and their solution approaches, Int. J. Prod. Econ. 142 (2) (2013) 259–277.
[3] I. Baybars, A survey of exact algorithms for the simple assembly line balancing problem, Manage. Sci. 32 (8) (1986) 909–932.
[4] E. Danna, 2008. Performance variability in mixed integer programming. Workshop on Mixed Integer Programming (MIP 2008), August 4–7, 2008, New York City, USA.
[5] A. Dolgui, S. Kovalev, Scenario based robust line balancing: Computational complexity, Discrete Appl. Math. 160 (13–14) (2012) 1955–1963.
[6] A. Dolgui, J.-M. Proth, Supply Chain Engineering: Useful Methods and Techniques, Springer-Verlag, London, 2010.
[7] V. Emelichev, E. Girlich, Yu. Nikulin, D. Podkopaev, Stability and regularization of vector problems of integer linear programming, Optimization 51 (4) (2002) 645–676.
[8] M. Garey, D. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W.H. Freeman and Company, San Francisco, 1979.
[9] M. Gen, Y. Tsujimura, Y. Li, Fuzzy assembly line balancing using genetic algorithms, Comput. Ind. Engrg. 31 (3–4) (1996) 631–634.
[10] E. Gurevsky, O. Battaïa, A. Dolgui, Balancing of simple assembly lines under variations of task processing times, Ann. Oper. Res. 201 (1) (2012) 265–286.
[11] E. Gurevsky, O. Battaïa, A. Dolgui, Stability measure for a generalized assembly line balancing problem, Discrete Appl. Math. 161 (3) (2013) 377–394.
[12] E. Gurevsky, O. Hazır, O. Battaïa, A. Dolgui, Robust balancing of straight assembly lines with interval task times, J. Oper. Res. Soc. 64 (11) (2013) 1607–1613.
[13] N. Hop, A heuristic solution for fuzzy mixed-model line balancing problem, European J. Oper. Res. 168 (3) (2006) 798–810.
[14] R. Klein, A. Scholl, Maximizing the production rate in simple assembly line balancing – a branch and bound procedure, European J. Oper. Res. 91 (2) (1996) 367–385.
[15] P. Kouvelis, G. Yu, Robust Discrete Optimization and Its Applications, Kluwer Academic Publishers, 1996.
[16] M. Libura, E. van der Poort, G. Sierksma, J. van der Veen, Stability aspects of the traveling salesman problem based on k-best solutions, Discrete Appl. Math. 87 (1–3) (1998) 159–185.
[17] A. Lodi, Mixed integer programming computation, in: M. Jünger, et al. (Eds.), 50 Years of Integer Programming 1958–2008, Springer-Verlag, Berlin Heidelberg, 2010, pp. 619–646. Ch. 16.
[18] U. Özcan, Balancing stochastic two-sided assembly lines: A chance-constrained, piecewise-linear, mixed integer program and a simulated annealing algorithm, European J. Oper. Res. 205 (1) (2010) 81–97.
[19] J. Patterson, J. Albracht, Assembly-line balancing: zero–one programming with Fibonacci search, Oper. Res. 23 (1) (1975) 166–172.
[20] B. Rekiek, A. Dolgui, A. Delchambre, A. Bratcu, State of art of optimization methods for assembly line design, Annu. Rev. Control 26 (2) (2002) 163–174.
[21] M. Ritt, A. Costa, 2011. A comparison of formulations for the simple assembly line balancing problem. http://arxiv.org/abs/1111.0934.
[22] A. Scholl, Balancing and Sequencing of Assembly Lines, second ed., Physica-Verlag Heidelberg, 1999.
[23] A. Scholl, C. Becker, State-of-the-art exact and heuristic solution procedures for simple assembly line balancing, European J. Oper. Res. 168 (3) (2006) 666–693.
[24] Y. Sotskov, A. Dolgui, M.-C. Portmann, Stability analysis of an optimal balance for an assembly line with fixed cycle time, European J. Oper. Res. 168 (3) (2006) 783–797.
[25] Y. Sotskov, A. Dolgui, N. Sotskova, F. Werner, Stability of optimal line balance with given station set, in: A. Dolgui, J. Soldek, O. Zaikin (Eds.), Supply Chain Optimisation: Product/Process Design, Facility Location and Flow Control, in: Applied Optimization, Vol. 94, Springer, US, 2005, pp. 135–149. Ch. 10.
[26] Yu. Sotskov, V. Leontev, E. Gordeev, Some concepts of stability analysis in combinatorial optimization, Discrete Appl. Math. 58 (2) (1995) 169–190.
[27] Yu. Sotskov, N. Sotskova, T.-C. Lai, F. Werner, Scheduling under Uncertainty: Theory and Algorithms, Belorusskaya nauka, Minsk, 2010.
[28] Y. Tsujimura, M. Gen, E. Kubota, Solving fuzzy assembly-line balancing problem with genetic algorithms, Comput. Ind. Engrg. 29 (1–4) (1995) 543–547.
[29] T. Urban, W.-C. Chiang, An optimal piecewise-linear program for the U-line balancing problem with stochastic task times, European J. Oper. Res. 168 (3) (2006) 771–782.
[30] P. Zacharia, A. Nearchou, Multi-objective fuzzy assembly line balancing using genetic algorithms, J. Intell. Manuf. 23 (3) (2012) 615–627.

Appendix A. Detailed computational results

				P_1				P_∞			
Instance	n	m	T	best ρ_1	best UB	UB_1	CPU	best ρ_∞	best UB	UB_∞	CPU
MERTENS	7	4	9	0.00	0.00	3.50	0.02	0.00	0.00	3.50	0.03
BOWMAN8	8	4	25.5	4.50	4.50	13.50	0.02	4.50	4.50	13.50	0.01
MANSOOR	11	4	67.5	41.50	41.50	42.50	0.06	21.50	21.50	42.50	0.16
JAESCHKE	9	5	9	1.00	1.00	2.67	0.03	1.00	1.00	2.67	0.03
JACKSON	11	6	10.5	1.50	1.50	4.50	0.08	1.50	1.50	4.50	0.17
MITCHELL	21	7	19.5	6.50	6.50	9.17	0.16	4.25	4.25	9.17	0.62
ROSZIEG	25	8	19.5	4.50	4.50	6.50	0.27	3.25	3.25	6.50	0.87
HESKIA	28	8	162	55.00	55.00	55.00	0.23	38.00	38.00	55.00	210.85
LUTZ1	32	9	2100	574.00	574.00	700.00	1.03	494.00	494.00	700.00	6.46
BUXEY	29	11	37.5	12.50	12.50	12.50	0.20	8.25	8.25	12.50	12.43
SAWYER30	30	11	37.5	14.50	14.50	18.50	3.43	9.50	9.50	18.50	36.63
GUNTHER	35	10	60	18.00	18.00	20.00	0.94	12.00	12.00	20.00	113.51
HAHN	53	7	2662.5	537.50	537.50	1455.50	1.98	268.75	268.75	1224.75	1.47
KILBRID	45	9	82.5	27.50	27.50	27.50	0.39	15.50	15.88	27.50	dnf
TONGE70	70	18	234	77.00	78.00	78.00	dnf	33.50	39.00	78.00	dnf
WARNECKE	58	24	79.5	27.50	27.50	27.50	255.67	14.83	23.27	27.50	dnf
ARC83	83	17	5536.5	1688.50	2702.67	2702.67	dnf	751.50	841.59	2256.75	dnf
LUTZ3	89	18	111	24.00	33.75	37.00	dnf	13.50	15.11	37.00	dnf
BARTHOLD	148	12	574.5	191.50	191.50	191.50	165.10	33.63	34.05	138.13	dnf
MUKHERJE	94	20	256.5	67.50	85.50	85.50	dnf	33.75	33.75	85.50	66.94
ARC111	111	22	8533.5	3474.50	3474.50	3474.50	257.54	1184.75	1318.10	3474.50	dnf
LUTZ2	89	38	15	2.00	5.00	5.00	dnf	1.00	4.05	5.00	dnf
WEE-MAG	75	60	40.5	9.50	14.50	14.50	dnf	9.50	14.50	14.50	dnf
BARTHOL2	148	41	124.5	37.50	41.50	41.50	dnf	18.25	23.53	41.50	dnf
SCHOLL	297	41	2079	505.00	984.13	984.13	dnf	137.00	207.79	866.07	dnf
			Nb. opt		**17/25**		**0.65**		**14/25**		**29.54**

Table A.1: Comparison of P_1 and P_∞ for $|\widetilde{V}^1| = \lceil \frac{n}{4} \rceil$ and $|\widehat{W}| = 0$

				P_1				P_∞			
Instance	n	m	T	best ρ_1	best UB	UB_1	CPU	best ρ_∞	best UB	UB_∞	CPU
MERTENS	7	4	9	4.00	4.00	7.00	0.02	4.00	4.00	7.00	0.02
BOWMAN8	8	4	25.5	7.50	7.50	25.50	0.03	3.75	3.75	25.50	0.02
MANSOOR	11	4	67.5	**67.50**	67.50	67.50	0.01	**67.50**	67.50	67.50	0.01
JAESCHKE	9	5	9	1.00	1.00	4.00	0.01	0.50	0.50	4.00	0.01
JACKSON	11	6	10.5	5.50	5.50	8.50	0.02	5.50	5.50	8.50	0.16
MITCHELL	21	7	19.5	12.50	12.50	15.75	0.09	11.50	11.50	15.75	0.58
ROSZIEG	25	8	19.5	7.50	7.50	15.50	0.19	6.50	6.50	15.50	0.50
HESKIA	28	8	162	136.00	136.00	136.00	0.94	131.00	131.00	136.00	1.08
LUTZ1	32	9	2100	1186.00	1186.00	1586.67	2.03	700.00	700.00	1586.67	8.44
BUXEY	29	11	37.5	17.50	17.50	29.50	1.26	17.50	17.50	29.50	3.65
SAWYER30	30	11	37.5	25.50	25.50	29.50	1.30	23.50	23.50	29.50	6.57
GUNTHER	35	10	60	20.00	20.00	39.00	0.08	20.00	20.00	39.00	1.50
HAHN	53	7	2662.5	1691.50	1691.50	2305.75	0.09	1691.50	1691.50	2305.75	0.36
KILBRID	45	9	82.5	60.50	60.50	63.50	0.61	24.75	24.75	63.50	4.48
TONGE70	70	18	234	132.00	140.10	140.40	dnf	91.00	135.37	140.40	dnf
WARNECKE	58	24	79.5	45.50	55.95	60.00	dnf	43.50	56.10	60.00	dnf
ARC83	83	17	5536.5	3436.50	3436.50	3682.70	139.96	1571.75	3355.72	3682.70	dnf
LUTZ3	89	18	111	51.00	51.00	70.80	88.70	43.00	43.00	70.80	361.83
BARTHOLD	148	12	574.5	417.50	418.50	420.00	dnf	198.25	407.55	420.00	dnf
MUKHERJE	94	20	256.5	171.50	171.50	184.40	675.21	142.50	172.62	184.40	dnf
ARC111	111	22	8533.5	5570.50	6151.30	6223.00	dnf	5104.50	6028.95	6223.00	dnf
LUTZ2	89	38	15	6.00	8.00	10.00	dnf	3.50	9.06	10.00	dnf
WEE-MAG	75	60	40.5	17.50	18.50	40.50	dnf	17.50	30.19	40.50	dnf
BARTHOL2	148	41	124.5	67.50	77.56	79.14	dnf	0.00	77.80	79.14	dnf
SCHOLL	297	41	2079	1267.00	1416.73	1416.73	dnf	0.00	1379.43	1416.73	dnf
			Nb. opt		**17/25**		**6.36**		**15/25**		**25.94**

Table A.2: Comparison of P_1 and P_∞ for $|\widetilde{V}^1| = 0$ and $|\widehat{W}| = \lceil \frac{m}{4} \rceil$

				P_1				P_∞			
Instance	n	m	T	best ρ_1	best UB	UB_1	CPU	best ρ_∞	best UB	UB_∞	CPU
MERTENS	7	4	9	0.00	0.00	3.50	0.06	0.00	0.00	3.50	0.08
BOWMAN8	8	4	25.5	4.50	4.50	13.50	0.01	1.50	1.50	13.50	0.03
MANSOOR	11	4	67.5	41.50	41.50	42.50	0.02	14.50	14.50	42.50	0.16
JAESCHKE	9	5	9	1.00	1.00	2.67	0.04	0.50	0.50	2.67	0.03
JACKSON	11	6	10.5	1.50	1.50	4.50	0.06	1.50	1.50	4.50	0.23
MITCHELL	21	7	19.5	6.50	6.50	9.17	0.23	3.50	3.50	9.17	0.61
ROSZIEG	25	8	19.5	3.50	3.50	6.50	0.22	1.17	1.17	6.50	0.52
HESKIA	28	8	162	55.00	55.00	55.00	0.17	34.67	34.67	55.00	620.20
LUTZ1	32	9	2100	494.00	494.00	700.00	2.71	295.00	295.00	700.00	9.08
BUXEY	29	11	37.5	12.50	12.50	12.50	0.28	5.83	5.83	12.50	11.23
SAWYER30	30	11	37.5	14.50	14.50	18.50	1.34	7.25	7.25	18.50	189.14
GUNTHER	35	10	60	17.00	17.00	20.00	5.38	7.50	7.50	20.00	4.87
HAHN	53	7	2662.5	537.50	537.50	1455.50	1.17	250.30	250.30	1224.75	1.84
KILBRID	45	9	82.5	27.50	27.50	27.50	0.39	9.50	10.11	27.50	dnf
TONGE70	70	18	234	71.00	78.00	78.00	dnf	25.00	39.00	78.00	dnf
WARNECKE	58	24	79.5	20.50	27.50	27.50	dnf	10.50	20.74	27.50	dnf
ARC83	83	17	5536.5	1512.50	2674.81	2702.67	dnf	488.25	695.83	2256.75	dnf
LUTZ3	89	18	111	21.00	25.50	37.00	dnf	10.00	10.00	37.00	386.35
BARTHOLD	148	12	574.5	191.50	191.50	191.50	8.38	32.25	33.26	138.13	dnf
MUKHERJE	94	20	256.5	67.50	85.50	85.50	dnf	24.25	31.18	85.50	dnf
ARC111	111	22	8533.5	3187.50	3474.50	3474.50	dnf	880.75	1230.17	3474.50	dnf
LUTZ2	89	38	15	2.00	5.00	5.00	dnf	1.00	3.40	5.00	dnf
WEE-MAG	75	60	40.5	9.50	14.50	14.50	dnf	9.50	14.50	14.50	dnf
BARTHOL2	148	41	124.5	33.50	41.50	41.50	dnf	11.63	23.53	41.50	dnf
SCHOLL	297	41	2079	485.00	984.13	984.13	dnf	0.00	207.79	866.07	dnf
			Nb. opt		**15/25**		**0.90**		**14/25**		**64.47**

Table A.3: Comparison of P_1 and P_∞ for $|\widetilde{V}^1| = \lceil \frac{n}{4} \rceil$ and $|\widehat{W}| = \lceil \frac{m}{4} \rceil$

				P_1				P_∞			
Instance	n	m	T	best ρ_1	best UB	UB_1	CPU	best ρ_∞	best UB	UB_∞	CPU
MERTENS	7	4	9	0.00	0.00	2.33	0.07	0.00	0.00	2.33	0.04
BOWMAN8	8	4	25.5	4.50	4.50	9.00	0.03	2.25	2.25	9.00	0.03
MANSOOR	11	4	67.5	23.50	23.50	28.33	0.06	12.75	12.75	26.25	0.11
JAESCHKE	9	5	9	1.00	1.00	2.00	0.03	0.50	0.50	2.00	0.05
JACKSON	11	6	10.5	1.50	1.50	4.00	0.19	1.25	1.25	4.00	0.30
MITCHELL	21	7	19.5	3.50	3.50	6.30	0.16	1.75	1.75	5.75	0.50
ROSZIEG	25	8	19.5	3.50	3.50	5.17	0.39	1.75	1.75	5.17	0.80
HESKIA	28	8	162	45.00	45.00	45.33	1.65	19.00	19.00	45.33	397.07
LUTZ1	32	9	2100	494.00	494.00	700.00	2.37	228.00	228.00	700.00	5.35
BUXEY	29	11	37.5	9.50	9.50	12.50	2.61	5.25	5.25	12.50	228.73
SAWYER30	30	11	37.5	9.50	9.50	12.50	13.57	5.25	5.25	12.42	19.33
GUNTHER	35	10	60	12.00	12.00	16.71	25.41	6.00	6.00	16.71	28.49
HAHN	53	7	2662.5	326.50	326.50	1152.88	1.26	133.17	133.17	573.88	1.48
KILBRID	45	9	82.5	26.50	27.00	27.50	dnf	8.17	8.28	20.67	dnf
TONGE70	70	18	234	47.00	58.50	58.50	dnf	17.33	20.06	58.50	dnf
WARNECKE	58	24	79.5	15.50	25.71	25.71	dnf	8.25	12.41	25.71	dnf
ARC83	83	17	5536.5	1147.50	1673.95	1673.95	dnf	357.25	418.65	1256.67	dnf
LUTZ3	89	18	111	19.00	32.00	32.00	dnf	7.00	7.33	28.00	dnf
BARTHOLD	148	12	574.5	154.50	164.21	164.21	dnf	16.75	17.03	74.50	dnf
MUKHERJE	94	20	256.5	63.50	76.83	76.83	dnf	15.88	17.85	71.28	dnf
ARC111	111	22	8533.5	2301.50	2667.00	2667.00	dnf	600.17	666.75	2630.83	dnf
LUTZ2	89	38	15	0.00	4.35	4.35	dnf	0.50	1.83	4.35	dnf
WEE-MAG	75	60	40.5	7.50	13.50	13.50	dnf	6.50	13.50	13.50	dnf
BARTHOL2	148	41	124.5	22.50	33.62	33.62	dnf	9.25	11.76	33.62	dnf
SCHOLL	297	41	2079	338.00	623.36	623.36	dnf	56.33	104.59	390.67	dnf
			Nb. opt		**13/25**		**8.49**		**13/25**		**52.48**

Table A.4: Comparison of P_1 and P_∞ for $|\widetilde{V}^1| = \lceil \frac{n}{2} \rceil$ and $|\widehat{W}| = 0$

				P_1				P_∞			
Instance	n	m	T	best ρ_1	best UB	UB_1	CPU	best ρ_∞	best UB	UB_∞	CPU
MERTENS	7	4	9	3.00	3.00	3.50	0.02	3.00	3.00	3.50	0.01
BOWMAN8	8	4	25.5	4.50	4.50	13.50	0.02	1.50	1.50	13.50	0.02
MANSOOR	11	4	67.5	33.50	33.50	42.50	0.06	23.75	23.75	42.50	0.02
JAESCHKE	9	5	9	0.00	0.00	2.67	0.01	0.00	0.00	2.67	0.02
JACKSON	11	6	10.5	2.50	2.50	5.67	0.02	1.25	1.25	5.67	0.16
MITCHELL	21	7	19.5	6.50	6.50	7.88	0.19	2.17	2.17	7.88	0.47
ROSZIEG	25	8	19.5	3.50	3.50	7.75	0.17	1.75	1.75	7.75	0.14
HESKIA	28	8	162	68.00	68.00	68.00	0.44	54.00	54.00	68.00	1.58
LUTZ1	32	9	2100	822.00	822.00	952.00	0.69	312.00	312.00	952.00	4.88
BUXEY	29	11	37.5	12.50	12.50	14.75	0.53	6.25	6.25	14.75	36.69
SAWYER30	30	11	37.5	13.50	13.50	14.75	1.34	6.25	6.25	14.75	77.72
GUNTHER	35	10	60	18.00	18.00	23.40	0.44	9.00	9.00	23.40	162.10
HAHN	53	7	2662.5	1109.50	1109.50	1152.88	0.42	134.39	134.39	1152.88	13.01
KILBRID	45	9	82.5	36.50	37.08	38.10	dnf	11.13	12.50	38.10	dnf
TONGE70	70	18	234	74.00	78.00	78.00	dnf	20.50	54.49	78.00	dnf
WARNECKE	58	24	79.5	20.50	29.30	30.00	dnf	9.50	24.68	30.00	dnf
ARC83	83	17	5536.5	1845.50	1845.50	2045.94	51.08	477.75	1093.73	2045.94	dnf
LUTZ3	89	18	111	37.00	37.00	39.33	30.26	10.33	14.01	39.33	dnf
BARTHOLD	148	12	574.5	208.50	209.50	210.00	dnf	51.38	117.74	210.00	dnf
MUKHERJE	94	20	256.5	84.50	86.36	92.20	dnf	25.38	45.38	92.20	dnf
ARC111	111	22	8533.5	3216.50	3320.90	3394.36	dnf	1015.17	2517.57	3394.36	dnf
LUTZ2	89	38	15	3.00	4.00	5.00	dnf	0.50	4.06	5.00	dnf
WEE-MAG	75	60	40.5	17.50	17.50	31.03	150.73	16.50	20.13	31.03	dnf
BARTHOL2	148	41	124.5	35.50	40.67	41.45	dnf	0.00	34.45	41.45	dnf
SCHOLL	297	41	2079	673.00	742.10	742.10	dnf	0.00	501.78	742.10	dnf
			Nb. opt		**16/25**		**0.33**		**13/25**		**22.83**

Table A.5: Comparison of P_1 and P_∞ for $|\widetilde{V}^1| = 0$ and $|\widehat{W}| = \lceil \frac{m}{2} \rceil$

				P_1				P_∞			
Instance	n	m	T	best ρ_1	best UB	UB_1	CPU	best ρ_∞	best UB	UB_∞	CPU
MERTENS	7	4	9	0.00	0.00	2.33	0.08	0.00	0.00	2.33	0.05
BOWMAN8	8	4	25.5	4.50	4.50	9.00	0.03	1.50	1.50	9.00	0.04
MANSOOR	11	4	67.5	23.50	23.50	28.35	0.06	7.75	7.75	26.25	0.16
JAESCHKE	9	5	9	0.00	0.00	2.00	0.03	0.00	0.00	2.00	0.03
JACKSON	11	6	10.5	1.50	1.50	4.00	0.08	0.75	0.75	4.00	0.23
MITCHELL	21	7	19.5	3.50	3.50	6.30	0.27	1.50	1.50	5.75	0.83
ROSZIEG	25	8	19.5	3.50	3.50	5.17	0.28	1.17	1.17	5.17	0.94
HESKIA	28	8	162	44.00	44.00	45.33	1.55	17.33	17.33	44.00	128.56
LUTZ1	32	9	2100	462.00	462.00	700.00	2.81	218.00	218.00	700.00	1.59
BUXEY	29	11	37.5	7.50	7.50	12.50	5.07	3.75	3.75	12.50	16.05
SAWYER30	30	11	37.5	9.50	9.50	12.50	6.93	4.17	4.17	12.42	57.86
GUNTHER	35	10	60	12.00	12.00	16.71	7.18	4.25	4.25	16.94	54.57
HAHN	53	7	2662.5	326.50	326.50	1152.88	0.45	100.59	100.59	573.88	13.67
KILBRID	45	9	82.5	26.50	26.50	27.50	54.23	6.50	6.80	20.67	dnf
TONGE70	70	18	234	44.00	58.43	58.50	dnf	12.67	19.31	58.50	dnf
WARNECKE	58	24	79.5	13.50	25.71	25.71	dnf	5.75	11.82	25.71	dnf
ARC83	83	17	5536.5	1007.50	1673.95	1673.95	dnf	273.25	347.50	1256.67	dnf
LUTZ3	89	18	111	18.00	21.00	32.00	dnf	5.00	5.97	28.00	dnf
BARTHOLD	148	12	574.5	154.50	164.21	164.21	dnf	15.36	17.03	74.50	dnf
MUKHERJE	94	20	256.5	57.50	67.50	76.83	dnf	12.63	16.71	71.28	dnf
ARC111	111	22	8533.5	2300.50	2667.00	2667.00	dnf	476.58	647.24	2630.83	dnf
LUTZ2	89	38	15	0.00	4.00	4.35	dnf	0.50	1.65	4.35	dnf
WEE-MAG	75	60	40.5	7.50	13.50	13.50	dnf	6.50	13.42	13.50	dnf
BARTHOL2	148	41	124.5	21.50	33.62	33.62	dnf	4.75	11.76	33.62	dnf
SCHOLL	297	41	2079	362.00	623.36	623.36	dnf	0.00	104.59	390.67	dnf
			Nb. opt		**14/25**		**4.53**		**13/25**		**21.10**

Table A.6: Comparison of P_1 and P_∞ for $|\widetilde{V}^1| = \lceil \frac{n}{2} \rceil$ and $|\widehat{W}| = \lceil \frac{m}{2} \rceil$

				P_1				P_∞			
Instance	n	m	T	best ρ_1	best UB	UB_1	CPU	best ρ_∞	best UB	UB_∞	CPU
MERTENS	7	4	9	0.00	0.00	1.75	0.05	0.00	0.00	0.67	0.08
BOWMAN8	8	4	25.5	3.50	3.50	6.75	0.01	1.50	1.50	3.38	0.02
MANSOOR	11	4	67.5	19.50	19.50	21.25	0.08	6.88	6.88	13.17	0.13
JAESCHKE	9	5	9	0.00	0.00	1.60	0.02	0.00	0.00	0.63	0.01
JACKSON	11	6	10.5	1.50	1.50	2.83	0.03	0.75	0.75	1.35	0.17
MITCHELL	21	7	19.5	3.50	3.50	4.50	0.17	1.13	1.13	2.17	0.80
ROSZIEG	25	8	19.5	3.50	3.50	3.88	0.14	0.88	0.88	3.13	1.01
HESKIA	28	8	162	33.00	33.00	34.00	2.68	9.67	9.71	33.50	dnf
LUTZ1	32	9	2100	462.00	462.00	528.89	0.98	124.50	124.50	223.25	7.78
BUXEY	29	11	37.5	5.50	5.50	8.05	16.77	2.50	2.50	4.57	15.10
SAWYER30	30	11	37.5	6.50	6.50	8.05	18.86	2.50	2.50	4.43	109.19
GUNTHER	35	10	60	10.00	10.00	11.70	22.78	3.00	3.00	11.50	358.05
HAHN	53	7	2662.5	326.50	326.50	658.79	0.42	52.05	52.05	255.13	3.04
KILBRID	45	9	82.5	20.50	20.99	21.17	dnf	3.93	4.23	8.83	dnf
TONGE70	70	18	234	37.00	39.00	39.00	dnf	8.67	10.03	29.71	dnf
WARNECKE	58	24	79.5	11.50	14.67	15.00	dnf	3.83	6.21	12.17	dnf
ARC83	83	17	5536.5	955.50	1048.00	1083.15	dnf	178.75	215.71	451.94	dnf
LUTZ3	89	18	111	18.00	18.50	19.67	dnf	3.40	3.92	10.57	dnf
BARTHOLD	148	12	574.5	104.50	104.71	105.00	dnf	8.32	8.51	35.88	dnf
MUKHERJE	94	20	256.5	35.50	36.71	46.10	dnf	8.50	9.13	31.74	dnf
ARC111	111	22	8533.5	1633.50	1694.69	1697.18	dnf	306.50	336.38	1403.58	dnf
LUTZ2	89	38	15	1.00	2.00	2.56	dnf	0.33	0.99	2.56	dnf
WEE-MAG	75	60	40.5	5.50	13.50	13.50	dnf	1.75	5.75	5.98	dnf
BARTHOL2	148	41	124.5	18.50	21.03	21.23	dnf	4.10	5.88	15.81	dnf
SCHOLL	297	41	2079	348.00	380.10	380.10	dnf	0.00	52.47	196.21	dnf
			Nb. opt		**13/25**		**5.02**		**12/25**		**41.26**

Table A.7: Comparison of P_1 and P_∞ for $|\widetilde{V}^1| = n$ and $|\widehat{W}| = m$

Appendix B. Detailed instances

Data added to instance `ARC83`

Random task permutation

39 12 18 41 8 23 56 69 30 59 16 26 7 2 57 14 47 27 11 37 81 36 1 68 73 71 61 3 72 64 19 34 45 65 6 43 35 29 74 62 75 66 33 20 21 58 9 24 50 82 22 78 13 25 83 5 54 4 38 60 49 52 40 17 55 79 53 10 32 48 46 28 76 44 67 77 51 63 31 15 80 42 70

Random workstation permutation

13 5 4 3 16 9 1 6 17 12 7 14 8 15 10 2 11

Data added to instance `ARC111`

Random task permutation

39 20 95 64 85 33 75 49 52 13 107 1 61 98 81 94 18 70 26 63 56 38 35 44 16 45 89 77 19 73 23 43 74 32 67 51 96 58 76 5 109 28 4 7 71 102 101 12 34 30 9 10 17 99 100 3 25 104 111 57 72 84 65 108 80 82 91 14 27 90 92 41 11 60 50 93 54 22 103 68 88 8 59 47 97 2 29 79 48 86 83 66 24 6 46 62 69 78 87 31 15 55 36 53 40 110 21 106 37 42 105

Random workstation permutation

4 8 1 20 7 21 19 2 15 13 17 11 16 3 10 5 12 18 22 9 6 14

Data added to instance `BARTHOL2`

Random task permutation

39 77 70 121 74 75 136 145 96 33 60 16 58 12 40 107 115 98 8 120 131 18 129 37 126 111 106 76 6 36 108 7 38 92 48 141 122 97 34 143 66 44 41 54 123 127 104 4 114 57 128 21 137 24 67 42 2 130 135 68 47 73 19 146 109 35 113 134 147 83 100 85 87 95 28 55 22 81 63 139 26 9 13 88 103 64 31 56 59 25 3 51 14 72 29 71 1 89 65 125 82 53 52 45 148 50 78 43 11 142 46 5 15 117 93 23 20 138 91 119 132 90 110 61 84 112 140 32 17 133 49 79 124 116 102 27 99 101 118 30 10 62 94 80 105 86 69 144

Random workstation permutation

34 19 25 32 29 4 37 12 1 13 21 30 11 35 23 28 5 8 17 10 22 40 36 7 14 31 24 2 16 41 27 6 20 9 18 39 38 33 15 3 26

Data added to instance `BARTHOLD`

Random task permutation

39 77 70 121 74 75 136 145 96 33 60 16 58 12 40 107 115 98 8 120 131 18 129 37 126 111 106 76 6 36 108 7 38 92 48 141 122 97 34 143 66 44 41 54 123 127

104 4 114 57 128 21 137 24 67 42 2 130 135 68 47 73 19 146 109 35 113 134 147 83 100 85 87 95 28 55 22 81 63 139 26 9 13 88 103 64 31 56 59 25 3 51 14 72 29 71 1 89 65 125 82 53 52 45 148 50 78 43 11 142 46 5 15 117 93 23 20 138 91 119 132 90 110 61 84 112 140 32 17 133 49 79 124 116 102 27 99 101 118 30 10 62 94 80 105 86 69 144
Random workstation permutation
12 7 4 1 8 6 9 3 2 5 11 10

Data added to instance `BOWMAN8`
Random task permutation
7 6 5 3 8 2 4 1
Random workstation permutation
4 3 2 1

Data added to instance `BUXEY`
Random task permutation
10 21 18 23 6 16 22 14 19 17 2 4 7 20 5 9 3 15 24 1 27 13 26 28 12 29 8 11 25
Random workstation permutation
11 8 2 5 9 7 3 6 4 10 1

Data added to instance `GUNTHER`
Random task permutation
23 7 18 19 22 28 4 33 15 14 13 9 17 11 31 6 24 3 10 1 35 2 29 21 32 25 20 8 27 5 30 16 12 34 26
Random workstation permutation
10 8 9 4 1 7 3 5 6 2

Data added to instance `HAHN`
Random task permutation
5 41 50 53 20 22 21 6 29 9 35 11 13 8 49 10 38 34 32 25 46 36 23 30 19 18 7 26 24 37 51 16 39 43 28 12 17 52 14 15 2 1 44 45 4 47 40 3 33 48 42 31 27
Random workstation permutation
1 2 4 5 7 3 6

Data added to instance `HESKIA`
Random task permutation
7 25 15 19 24 4 1 17 3 28 16 8 21 10 13 9 26 18 14 11 2 12 27 5 22 6 20 23
Random workstation permutation

4 7 1 3 6 8 5 2

Data added to instance `JACKSON`
Random task permutation
8 9 10 2 7 3 1 5 6 11 4
Random workstation permutation
5 3 6 4 1 2

Data added to instance `JAESCHKE`
Random task permutation
6 4 9 8 5 1 2 7 3
Random workstation permutation
3 2 1 5 4

Data added to instance `KILBRID`
Random task permutation
16 8 22 45 31 32 23 19 18 37 21 44 5 14 15 4 30 38 12 3 6 43 1 33 25 13 28 42 7 39 2 10 24 35 36 27 29 34 40 20 41 17 11 9 26
Random workstation permutation
9 8 1 2 5 7 3 4 6

Data added to instance `LUTZ1`
Random task permutation
15 18 8 30 2 4 7 11 14 31 13 5 25 21 19 20 17 29 24 10 32 3 23 12 27 26 9 22 28 1 6 16
Random workstation permutation
7 5 4 2 6 9 1 3 8

Data added to instance `LUTZ2`
Random task permutation
3 85 71 18 30 46 62 2 11 53 58 14 78 12 22 52 82 20 68 27 28 37 73 87 89 50 13 25 61 17 70 86 24 47 54 29 5 66 1 31 56 69 79 4 21 75 74 64 32 41 38 9 59 63 10 48 49 7 19 39 65 15 51 60 43 55 76 33 57 8 23 84 35 26 44 6 83 40 45 80 16 88 36 42 67 72 77 81 34
Random workstation permutation
23 8 38 6 20 19 29 7 15 24 13 39 2 34 22 4 17 35 31 33 26 36 25 16 1 3 14 32 27 5 28 30 10 21 9 18 37 12 11

Data added to instance `LUTZ3`
Random task permutation
27 37 22 75 29 32 56 5 78 41 47 82 7 59 4 15 80 25 28 57 34 16 55 61 54 35 42 71 44 73 50 31 63 76 51 72 40 20 81 83 33 21 85 36 3 43 48 79 19 46 77 70 64 12 6 18 67 23 45 2 30 69 9 11 10 17 53 52 86 89 8 38 74 65 24 39 14 26 49 13 87 62 88 84 58 66 68 60 1
Random workstation permutation
8 16 18 6 5 17 9 10 11 7 1 14 15 3 13 2 4 12

Data added to instance `MANSOOR`
Random task permutation
7 9 4 2 11 1 10 6 3 8 5
Random workstation permutation
3 4 1 2

Data added to instance `MERTENS`
Random task permutation
3 7 6 4 5 2 1
Random workstation permutation
3 4 2 1

Data added to instance `MITCHELL`
Random task permutation
14 10 16 5 8 7 17 2 3 15 19 18 12 21 4 11 13 9 1 20 6
Random workstation permutation
6 3 7 4 5 1 2

Data added to instance `MUKHERJE`
Random task permutation
79 33 54 7 88 18 40 30 25 65 3 5 17 87 73 45 52 8 53 68 83 64 80 32 57 92 42 59 72 48 61 50 24 82 9 35 63 75 23 85 49 67 70 81 58 77 21 89 36 22 4 90 47 12 51 2 56 10 78 14 69 19 27 76 6 66 43 60 39 28 46 38 71 93 44 94 1 37 26 84 13 16 74 31 41 62 86 34 15 55 11 20 29 91
Random workstation permutation
12 4 17 10 3 1 6 15 13 7 18 5 11 20 14 9 19 2 16 8

Data added to instance `ROSZIEG`
Random task permutation

14 23 19 15 11 8 17 6 24 5 1 4 7 25 16 2 13 21 18 12 20 22 9 10 3
Random workstation permutation
7 1 3 6 2 5 8 4

Data added to instance `SAWYER30`
Random task permutation
30 3 2 9 22 12 11 17 29 19 27 26 16 28 23 5 21 25 18 7 10 24 8 13 6 4 20 15 1 14
Random workstation permutation
2 4 11 3 8 9 10 6 5 7 1

Data added to instance `SCHOLL`
Random task permutation
121 53 239 163 222 272 207 175 15 271 285 250 74 168 228 214 263 196 256 30 211 147 144 227 201 106 46 141 52 184 177 146 219 240 205 249 125 283 171 216 68 223 234 10 29 145 274 48 209 19 108 194 111 182 183 160 77 269 200 112 85 41 237 2 25 257 124 235 164 149 270 24 136 9 169 203 291 210 220 76 57 23 99 88 42 225 64 120 162 16 122 155 62 34 289 116 148 60 245 102 248 32 114 246 231 142 199 193 192 258 118 190 189 1 130 84 265 153 204 259 101 104 218 134 113 82 47 20 81 282 224 58 33 296 91 229 131 281 186 27 179 253 65 35 247 51 254 4 129 242 73 208 161 295 137 212 241 105 266 45 151 275 103 8 31 264 152 233 119 260 244 92 202 133 37 284 59 66 273 174 195 159 49 70 187 252 166 165 191 226 79 36 67 185 294 83 7 55 251 293 297 39 78 96 97 69 197 279 157 206 90 178 213 110 287 56 180 93 11 107 44 126 286 109 80 17 139 150 292 167 172 230 94 143 238 5 40 232 135 18 138 13 75 236 127 117 156 217 3 95 288 12 278 262 215 22 132 86 173 38 261 176 71 140 170 181 54 123 21 61 6 277 87 115 154 290 26 280 89 243 98 128 72 188 28 255 63 198 158 268 221 43 100 14 276 50 267
Random workstation permutation
27 14 28 4 35 34 11 30 9 20 13 29 1 36 32 22 17 41 12 10 21 2 15 33 40 18 24 5 8 19 37 6 38 31 39 26 16 23 3 25 7

Data added to instance `TONGE70`
Random task permutation
49 16 13 57 11 9 63 24 59 14 21 69 36 55 37 7 15 39 33 46 52 67 5 3 19 45 28 25 4 48 22 31 56 61 10 23 12 66 34 54 50 38 62 65 30 40 43 47 68 51 58 42 44 70 20 26 27 17 41 8 64 32 1 60 18 29 2 53 35 6
Random workstation permutation

10 4 2 16 8 5 18 1 6 17 7 14 3 15 13 9 12 11

Data added to instance `WARNECKE`
Random task permutation
35 29 42 51 15 21 38 33 45 41 40 34 14 47 57 1 49 3 22 10 27 16 26 28 19 37 8 36 43 54 12 9 4 24 30 6 50 58 44 13 55 11 7 17 46 2 32 18 53 31 48 52 25 20 39 23 5 56
Random workstation permutation
5 8 23 21 17 18 13 4 19 6 15 3 14 1 9 16 11 20 24 12 10 22 7 2

Data added to instance `WEE-MAG`
Random task permutation
40 10 47 50 59 43 33 49 17 4 56 41 21 55 11 37 51 74 71 28 18 25 20 52 2 53 65 63 35 38 26 6 67 66 75 69 3 64 60 68 44 24 14 54 42 61 12 19 9 16 46 7 22 45 27 39 15 31 23 57 34 29 72 73 32 1 5 8 13 30 36 70 58 62 48
Random workstation permutation
18 1 8 34 5 20 10 12 17 31 33 15 30 38 36 4 22 14 25 2 23 24 43 35 27 7 13 39 40 28 6 11 45 44 21 16 9 41 32 42 37 26 29 3 19