A MIN-MAX PATH APPROACH FOR BALANCING ROBOTIC ASSEMBLY LINES WITH SEQUENCE-DEPENDENT SETUP TIMES

Y Lahrichi, L. Deroussi, N Grangeon, S Norre

13^{th} *International Conference on Modeling, Optimization and Simuation* - MOSIM*'20 - November 12-14, 2020*
Agadir - Morocco "New advances and challenges for sustainable and smart industries"

A MIN-MAX PATH APPROACH FOR BALANCING ROBOTIC ASSEMBLY LINES WITH SEQUENCE-DEPENDENT SETUP TIMES

Y. LAHRICHI

Université Clermont Auvergne
CNRS, LIMOS
63000 Clermont–Ferrand, France
youssef.lahrichi@uca.fr

L. DEROUSSI, N. GRANGEON, S. NORRE

Université Clermont Auvergne
LIMOS UMR CNRS 6158, antenne de l'IUT d'Allier
03100 Montluçon, France
laurent.deroussi@uca.fr, nathalie.grangeon@uca.fr, sylvie.norre@uca.fr

ABSTRACT: *We deal in this paper with SDRALBP-2, namely the Sequence-Dependent Robotic Assembly Line Balancing Problem of type 2. The problem is of industrial relevance due to the growing robotization of the assembly lines in the new Industry 4.0 era. Given a set of operations that are necessary to assembly a product and a set of robot types with different performances, the problem is concerned with addressing three decision problems simultaneously while minimizing a given objective. The first decision is to assign the operations to a given set of stations placed in a straight line [Balancing decision], the second decision is to sequence the operations in each station due the sequence-dependent setup times [Sequencing decision] and the third decision is to assign a robot to each station [Equipment selection decision]. We consider the objective of minimizing the cycle time, which is the maximum duration spent by a product in some station. We propose in this paper a method of type Sequence-First Balance-And-Select-Second. The proposed method embeds a dynamic programming algorithm (that solves a polynomial case) in a metaheuristic. Benchmark instances are used to evaluate the proposed method.*

KEYWORDS: *Line balancing, Robotic, Sequence-dependent setup times, Polynomial case, Min-max path, Metaheuristic.*

1 INTRODUCTION

We assist nowadays to the growing robotization of all manufacturing systems. This large-scale robotization is pushed by new industry 4.0 standards encouraging the use of cyber-physical systems. Cyber-physical systems are systems where robots or other physical components interact with cyber or software components in order to deliver a service or produce a good in ways that change with context. Besides, robots offer higher productivity and flexibility (US National Science Foundation).

Assembly lines follow the same robotization trend. More and more often, tasks in assembly lines are no longer performed by human operators (Nilakantan, Ponnambalam, Jawahar & Kanagaraj 2015), (Janardhanan, Li, Bocewicz, Banaszak & Nielsen 2019). Human operators are only concerned by supervising the production process while robots perform all the operations. We consider a straight assembly line. Such a line is a series of stations organized throw a straight line. In each station, a set of operations is performed on a product. The product is then moved from the current station to the next station and a new product is moved to the current station. The product is considered finished when it exits from the last station. A robotic assembly line is an assembly line where the operations are performed by robots. Balancing an assembly line is the problem of assigning the operations necessary to assemble a product to the stations. The maximum duration spent by a product on some station is called the cycle time.

Balancing a robotic assembly line raises two problems that are not usually considered jointly in literature:

- The equipment selection problem which is concerned with assigning a robot to each station. The relevance of the problem is justified by the different performances of the robots. Indeed, the duration of an operation depends on the type of robot used (Rubinovitz, Bukchin & Lenz 1993).
- The problem of sequencing the operations in each station. The latter is justified by the consideration of sequence-dependent setup. A setup time $t_{i,j,r}$ between operations i and j must be considered if operations j is performed just after op-

eration i by a robot r. The sequence-dependent setup times are considered to provide for the necessary tool change on the robot or handling on the product. We remark that the setup times are not only sequence-dependent but also robot-dependent.

The considered problem in then called the SDRALBP (*Sequence-Dependent Robotic Assembly Line Balancing Problem.*). If the considered objective is to minimize the cycle time given a fixed number of stations, then it is denoted SDRALBP-2.

We propose a method of type Sequence-First Balance-And-Select-Second to tackle this problem. This method relies on a novel algorithm, called minmax, that computes a min-max path in some auxiliary graph. We prove that the latter is optimal when a sequence of all operations is given. The select and balance subproblems can be polynomially solved thanks to minmax. The SDRALBP-2 is then reduced to finding the best sequence of operations. The sequencing subproblem is solved by metaheuristic.

The paper is organized as follows. The problem is described in section 2. A example is given in section 3. Our contribution is then positioned in literature in section 4. A mathematical formulation is described in section 5. The resolution approach is described in section 6 then tested on benchmark instances in section 7.

2 PROBLEM STATEMENT

Given a set N of operations, a set S of stations placed in a straight line and a set R of robot types, the SDRALBP, is concerned with addressing three decisions simultaneously:

- Balancing decision: Assign each operation to a station.
- (Equipment) Selection decision: Assign a robot to each station.
- Sequencing problem: Sequence the operations in each station.

The duration of an operation i depends on the type of robot r used and is denoted d_i^r. Operations are linked by precedence relations (when operation i precedes operation j, the station to which i is assigned should not be after that of j). Sequence-dependent setup times are also considered. A setup times $t_{i,i'}^r$ should be considered if operation i is performed just before operation i' in some station equipped by a robot of type r.

The workload of a station is the sum of durations and sequence-dependent setup times induced by the sequence of operations assigned to it. The cycle time stands for the maximum workload among the stations and is a key performance indicator of the assembly line.

n	Number of operations
N	Set of operations, indexed on $\{1, 2, \ldots, n\}$
s_{max}	Max. number of stations
S	Set of stations, indexed on $\{1, 2, \ldots, s_{max}\}$
n_r	Number of robot types available
R	Set of robot types, indexed on $\{1, 2, \ldots, n_r\}$
P	Set of couple$(i, j) \in N^2$ s.t. i precedes j
C	Cycle time
d_i^r $i \in N, r \in R$	Duration of operation i on robot of type r
$t_{i,i'}^r$ $i, i' \in N, r \in R$	Setup time between operations i and i' on a robot of type r

Table 1 – Notations used

In this study, the cycle time is the objective to minimize given a maximum number of station.

The same type of robot can be assigned to several stations without any limitation, i.e. we assume that we have enough robot units of each type of robot.

The notations introduced in table 1 are used all across the paper.

3 EXAMPLE

We illustrate the problem with a small instance. The different attributes of the instance are given as follows:

- Number of operations: $n = 10$.
- Number of types of robots: $n_r = 4$.
- Maximum number of stations: $s_{max} = 3$.

Precedence relations are illustrated in the precedence graph (figure 2). Durations and sequence-dependent setup times are given respectively in tables 2 and 3. A feasible solution is depicted in figure 1.

The solution is feasible since precedence relations are satisfied and the number of stations used does not exceed the maximum number of stations.

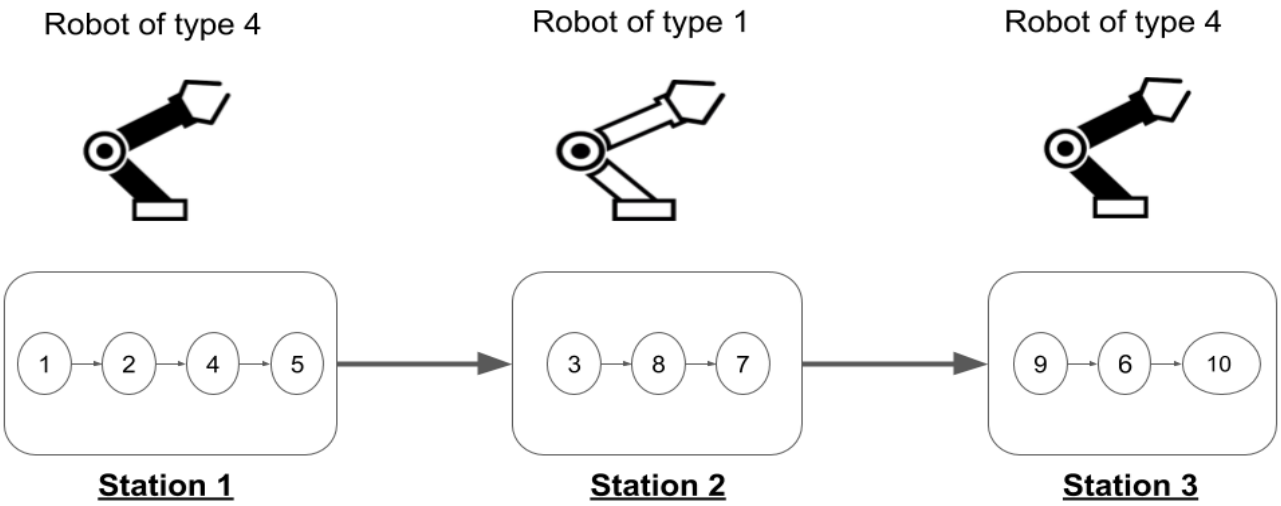

Figure 1 – Feasible solution

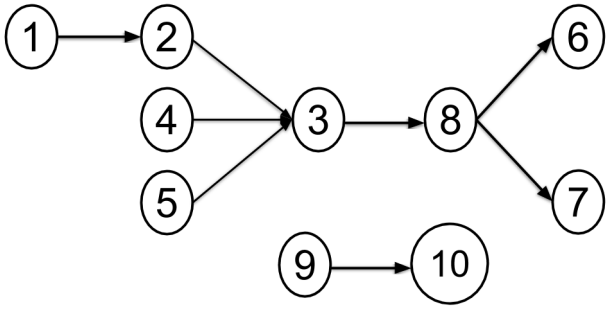

Figure 2 – Precedence graph

Op. \ Robot type	1	2	3	4
1	d_1^1 =3	4	4	2
2	$d_2^1 = 4$	3	3	2
3	1	3	2.3	1
4	3.4	7	4	2
5	3.5	5	1	1
6	2	5	4	1
7	1.3	8	2.4	3
8	5	5	3.1	2
9	3	5	2.4	2
10	5	7.5	1.1	3

Table 2 – Durations

The cycle time is obtained by computing the maximum among the workloads of the stations:

- On the first station, the workload is given by:

  $d_1^4 + t_{1,2}^4 + d_2^4 + t_{2,4}^4 + d_4^4 + t_{4,5}^4 + d_5^4 + t_{5,1}^4 = 7.6$

- On the second station, the workload is given by:

  $d_3^1 + t_{3,8}^1 + d_8^1 + t_{8,7}^1 + d_7^1 + t_{7,3}^1 = 9.3$

- On the third station, the workload is given by:

  $d_9^4 + t_{9,6}^4 + d_6^4 + t_{6,10}^4 + d_{10}^4 + t_{10,9}^4 = 7.6$

From the above calculation we can deduce the cycle time: $C = 9.3$.

4 STATE OF THE ART

The studied problem is a generalization of the Simple Assembly Line Balancing Problem. SDRALBP becomes a SALBP when $n_r = 1$ and $t_{i,j}^1 = 0, \forall i, j \in N$. SALBP is NP-Hard and has been extensively studied in literature. The interested reader can find a taxonomy of assembly line balancing problems in (Battaïa & Dolgui 2013).

The Sequence-Dependent Robotic Assembly line Balancing problem considers two particularities jointly:

- The consideration of the equipment selection problem: the problem is raised whenever the decider has to choose between different types of equipment with different performances in order to perform the assembly operations.

(a) Setup times of robots of type 1

Op.	1	2	3	4	5	6	7	8	9	10
1	0	$t^1_{1,2}$=0.9	0.5	0.3	0.2	0.2	0.6	0.4	0.6	0.4
2	0.4	0	0.6	0.1	0.3	0.4	0.1	0.6	0.6	0.4
3	0.2	0.1	0	0.1	0.1	0.2	0.7	0.8	0.6	0.2
4	0.1	0.2	0.4	0	0.1	0.2	0.1	0.2	0.6	0.3
5	0.5	0.5	0.9	0.4	0	0.7	2	0.7	0.6	0.3
6	0.2	0.1	0.2	0.2	0.2	0	0.3	0.7	0.6	0.3
7	0.7	0.5	0.4	0.2	0.7	1	0	0.2	0.6	0.4
8	0.1	0.1	0.5	0.5	0.1	0.2	0.8	0	0.6	0.3
9	0.7	0.5	0.4	0.2	0.7	1	0.1	0.2	0	0.4
10	0.1	0.1	0.1	0.5	0.1	0.2	0.5	0.2	0.6	0

(b) Setup times of robots of type 2

Op.	1	2	3	4	5	6	7	8	9	10
1	0	$t^2_{1,2}$=0.4	0.5	0.4	0.5	0.7	0.5	0.7	0.5	0.7
2	0.1	0	0.4	0.4	0.2	0.4	0.1	0.2	0.5	0.2
3	0.1	0.3	0	0.4	0.6	0.7	0.3	0.2	0.5	0.2
4	0.1	0.2	0.5	0	0.4	0.4	0.5	1.5	0.5	0.2
5	0.5	0.5	0.9	0.4	0	0.7	2	0.7	0.5	0.2
6	0.2	0.5	0.4	0.3	0.4	0	0.4	0.2	0.5	0.1
7	0.3	0.5	0.4	0.3	0.7	1	0	0.1	0.5	0.1
8	0.5	0.5	0.1	0.1	0.4	0.2	0.7	0	0.5	0.1
9	0.7	0.5	0.4	0.2	0.7	1	0.1	0.2	0	0.4
10	0.1	0.1	0.5	0.1	0.1	0.2	0.1	0.1	0.1	0

(c) Setup times of robots of type 3

Op.	1	2	3	4	5	6	7	8	9	10
1	0	$t^3_{1,2}$=0.1	0.9	0.7	0.4	0.4	0.5	0.7	0.5	0.1
2	0.2	0	0.4	0.4	0.3	0.1	0.1	0.2	0.5	0.1
3	0.2	0.3	0	0.1	0.4	0.6	0.3	0.2	0.5	0.2
4	0.3	0.2	0.1	0	0.5	0.4	0.4	0.5	0.5	0.2
5	0.5	0.1	0.4	0.9	0	0.1	0.1	0.3	0.5	0.2
6	0.1	0.2	0.5	0.4	0.3	0	0.3	0.1	0.5	0.7
7	0.8	0.1	0.5	0.2	0.7	0.5	0	0.1	0.5	0.2
8	0.7	0.3	0.2	0.2	0.1	0.3	0.4	0	0.5	0.7
9	0.7	0.5	0.4	0.2	0.2	1	0	0.2	0	0.4
10	0.1	0.1	0.2	0.2	0.1	0.2	0.2	0	0.6	0

(d) Setup times of robots of type 4

Op.	1	2	3	4	5	6	7	8	9	10
1	0	$t^3_{1,2}$=0.2	0.3	0.4	0.2	0.1	0.2	0.2	0.5	0.7
2	0.5	0	0.1	0.1	0.4	0.1	0.1	0.1	0.5	0.2
3	0.2	0.1	0	0.1	0.2	0.2	0.1	0.2	0.5	0.7
4	0.3	0.1	0.2	0	0.1	0.2	0.1	0.1	0.5	0.2
5	0.2	0.1	0.2	0.3	0	0.1	0.1	0.1	0.5	0.3
6	0.2	0.2	0.2	0.4	0.3	0	0.1	0.1	0.5	0.1
7	0.2	0.1	0.5	0.2	0.1	0.3	0	0.1	0.5	0.1
8	0.2	0.3	0.2	0.2	0.1	0.2	0.5	0	0.5	0.2
9	0.7	0.5	0.4	0.2	0.7	1	0.1	0.2	0	0.2
10	0.1	0.1	0.5	0.2	0.1	0.2	0.1	0.1	0.5	0

Table 3 – Setup times

The Robotic Assembly Line Balancing Problem (RALBP) is defined in (Rubinovitz et al. 1993). In the context of RALBP, different types of robots are available to perform the operations. The durations of the operations depend on the type of robot and the decider has to assign a single type of robot to each station. The RALBP has gained great importance due to its industrial relevance and due to the academic challenge it raises. The literature on RALBP is summarized in table 4. We can partition the literature on RALBP within two categories. Some authors consider that the same type of robot can be selected by multiple stations without any limitation (Rubinovitz et al. 1993), (Nilakantan et al. 2015), (Borba et al. 2018) whereas other authors consider that each robot can be selected by at most one station (Gao et al. 2009), (Janardhanan et al. 2019). The first assumption is the original assumption of RALBP as defined in (Rubinovitz et al. 1993), it is assumed in this paper.

- The consideration of sequence-dependent setup times: Sequence-dependent setup times in the context of assembly lines were introduced in (Andres, Miralles & Pastor 2008). Setup times are necessary to provide for tool change or product handling that can occur between two operations. The Sequence-Dependent Simple Assembly Line Balancing problem (SDSALBP) raises the decision of sequencing the operations in each station. Many authors have considered sequence-dependent setup times in the context of assembly lines: (Borisovsky, Delorme & Dolgui 2014), (Martino & Pastor 2010), (Lahrichi, Grangeon, Deroussi & Norre 2020).

The problem considered in this paper, SDRALBP-2, is concerned with minimizing the cycle time while addressing the balancing, selection and sequencing decisions simultaneously. To the best of our knowledge, the three previous decisions have never been addressed simultaneously in literature expect in (Janardhanan et al. 2019). Besides, (Janardhanan et al. 2019) assume that each robot can be selected at most once while we assume in this paper that each type of robot can be selected by multiple stations without any limitations. The results obtained in (Janardhanan et al. 2019) can not be compared directly to our results, they give an upper bound for our results.

The resolution approach described in this paper could be seen as a Sequence-First Balance-And-Select-Second algorithm. We propose a min-max path algorithm that addresses optimally the selection and the balancing decisions in polynomial time provided that a sequence of all operations is given. The later

Table 4 – Position of our study in the literature.

Article	Objectives			Sequence-dependent setup times
	Z_1	Z_2	Z_3	
(Rubinovitz et al. 1993)		✓		
(Levitin, Rubinovitz & Shnits 2006)	✓			
(Gao, Sun, Wang & Gen 2009)	✓			
(Yoosefelahi, Aminnayeri, Mosadegh & Ardakani 2012)	✓		✓	
(Nilakantan et al. 2015)	✓			
(Çil, Mete & Ağpak 2016)	✓	✓	✓	
(Borba, Ritt & Miralles 2018)	✓			
(Janardhanan et al. 2019)	✓			✓
Our study	✓			✓

Z_1: Cycle time, Z_2: Number of stations, Z_3: Cost of robots used

novel algorithm is then integrated in a metaheuristic framework.

5 MATHEMATICAL FORMULATION

To clarify the definition of the problem, we give a linear formulation based on the one from (Janardhanan et al. 2019). The latter is considering a limited number of robots per type of robot, we adapt it for the case of an unlimited number of robots by type. The formulation of (Janardhanan et al. 2019) is itself adapted from (Andres et al. 2008).

We use i to index an operation, s to index a station, j to index a position in the sequence of operations assigned to a station and r to index a type of robot. The following **variables** are used:

$$x_{i,s,j,r} = \begin{cases} 1 & \text{If the operation } i \text{ is assigned to the station } s \text{ at the } j\text{-th position of its sequence and performed by a robot of type } r. \\ 0 & \text{Otherwise.} \end{cases}$$

$$y_s = \begin{cases} 1 & \text{If at least one operation is assigned to station } s. \\ 0 & \text{Otherwise.} \end{cases}$$

$$v_{s,r} = \begin{cases} 1 & \text{If a robot of type } r \text{ is assigned to station } s. \\ 0 & \text{Otherwise.} \end{cases}$$

$$z_{i,i',s,r} = \begin{cases} 1 & \text{If operation } i \text{ is performed just before operation } i' \text{ at station } s \text{ by a robot of type } r. \\ 0 & \text{Otherwise.} \end{cases}$$

$$\mathrm{w}_{i,s} = \begin{cases} 1 & \text{If the operation } i \text{ is assigned to the last position in the sequence of station } s. \\ 0 & \text{Otherwise.} \end{cases}$$

C = Cycle time

We minimize the cycle time ($Min\ C$) under the constraints (1)-(12).

$$\sum_{s\in S}\sum_{j\in N}\sum_{r\in R} x_{i,s,j,r} = 1, \forall i \in N \tag{1}$$

$$\sum_{i\in N}\sum_{r\in R} x_{i,s,j,r} \leq 1, \forall s \in S, \forall j \in N \tag{2}$$

$$\sum_{i\in N} x_{i,s,j,r} \leq v_{s,r}, \forall s \in S, \forall j \in N, \forall r \in R \tag{3}$$

$$\sum_{r\in R} v_{s,r} = y_s, \forall s \in S \tag{4}$$

$$\sum_{i\in N} x_{i,s,j+1,r} \leq \sum_{i\in N} x_{i,s,j,r} \quad \forall s \in S, \forall j \in N - \{n\}, \forall r \in R \tag{5}$$

$$y_{s+1} \leq y_s, \forall s \in S - \{s_{max}\} \tag{6}$$

$$\sum_{s\in S}\sum_{j\in N}\sum_{r\in R}(n.(s-1)+j)x_{i,s,j,r} \leq \sum_{s\in S}\sum_{j\in N}\sum_{r\in R}(n.(s-1)+j)x_{i',s,j,r}, \forall (i,i') \in P \tag{7}$$

$$\sum_{i\in N}\sum_{j\in N}\sum_{r\in R} d_{i,r}.x_{i,s,j,r} + \sum_{i\in N}\sum_{i'\in N}\sum_{r\in R} t_{i,i',r}.z_{i,i',s,r} \leq C.y_s, \forall s \in S \tag{8}$$

$$x_{i,s,j,r} + x_{i',s,j+1,r} \leq 1 + z_{i,i',s,r}, \quad \forall i,i' \in N^2, i \neq i', \forall j \in N - \{n\}, \forall s \in S, \forall r \in R \tag{9}$$

$$x_{i,s,j,r} - \sum_{i' \in N; i' \neq i} x_{i',s,j+1,r} \leq w_{i,s} \quad \forall i \in N, \forall s \in S, \forall j \in N - \{n\} \tag{10}$$

$$x_{i,s,n,r} \leq w_{i,s}, \forall i \in N, \forall s \in S, \forall r \in R \tag{11}$$

$$w_{i,s} + x_{i',s,1,r} \leq 1 + z_{i,i',s,r} \quad \forall i \in N, i' \in N, i \neq i', \forall s \in S, \forall r \in R \tag{12}$$

The set of constraints (1) ensures that all operations must be assigned once and only once. (2) ensures that at most one operation can be assigned to the same position. (3) ensures that an operation is carried out by a robot of type r on a station s only if the station s is equipped by a robot of type r and (4) ensures that no more than one type of robot can be assigned to a station. (5) ensures that a position is only occupied by an operation if all of its previous positions are also occupied. (6) ensures that a station is only used if the previous stations are also used. (7) ensures that the precedence constraints are respected. (8) ensures that the cycle time constraints are respected on all stations. (9) ensures that $z_{i,i',s,r} = 1$ when i and i' follow each other on the station s (equipped by the robot r). (10) - (12) verify that $z_{i,i',s,r} = 1$ when i is the last operation assigned to the station s and i' the first operation assigned at the station s (equipped by the robot r).

6 RESOLUTION APPROACH

6.1 Overview and basic definitions

An algorithm of type Sequence-First Balance-And-Select-Second addresses the sequencing subproblem in the first step by giving a sequence of all operations called *giant sequence*. The balancing and the selection subproblems are then solved in the second step while respecting the giant sequence. We give some basic definitions and the algorithm used to obtain a giant sequence. Then, the algorithm to address the balancing and the selection decisions is described in subsection 6.2. This algorithm is then embedded in a metaheuristic (subsection 6.3).

Definition 6.1. *(Giant sequence) Given an instance of the SDRALBP, a giant sequence is a permutation of all its operations .*

Definition 6.2. *(A solution satisfying a giant sequence) A solution s of the SDRALBP is said to satisfy a giant sequence $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_n)$ if for all σ_i, σ_j such that $i < j$ either σ_i and σ_j are assigned to the same station in s or the station to which the operation σ_i is assigned must be before the station to which σ_j is assigned.*

Definition 6.3. *(Compatible giant sequence) A giant sequence σ is said to be compatible with respect to an instance of the SDRALBP if there exists at least a feasible solution satisfying σ.*

Theorem 6.4. *If we consider SDRALBP-2, a compatible giant sequence is simply a giant sequence respecting precedence constraints.*

Proof. Indeed, given a giant sequence respecting precedence constraints a feasible solution satisfying this giant sequence could be obtained by allocating all the operations of the giant sequence to a single station. □

Remark 1. *A compatible giant sequence respecting precedence constraints could be obtained thanks to Algorithm 1.*

Algorithm 1 Algorithm to build a giant sequence respecting precedence constraints

INPUT: An instance of the SDRALBP.
OUTPUT: A giant sequence σ respecting precedence constraints.

1: **while** σ is of size $< n$ **do**
2: Select randomly and uniformly i an operation without predecessor or such that all predecessors have already been included in σ.
3: Append i to σ
4: **end while**

6.2 Giant sequence fixed: polynomial case

We suppose in this subsection that the giant sequence is fixed. It is equivalent to the case where the precedence graph is a path of length n. We suppose without loss of generality that the giant sequence $\sigma = (1, 2, \ldots, n)$.

The resolution approach relies on an auxiliary graph $\mathcal{H}_{\mathcal{I}}(\sigma) = (V, A)$. The graph $\mathcal{H}_{\mathcal{I}}(\sigma) = (V, A)$ is directed, composed of the set of nodes V and the set of arcs A. The set of nodes is given by the set of operations plus an additional node corresponding to a fictitious operation, i.e $V = \{0\} \cup N$. An arc (i, j) from operation i to operation j refers to a station, the subsequence assigned to this station is $(i+1, i+2, \ldots, j)$ and the robot of type r selected for this station should minimize the workload induced by this subsequence: $\sum_{k=i+1}^{j} d_k^r + \sum_{k=i+1}^{j-1} t_{k,k+1}^r + t_{j,i+1}^r$. The arcs set A is given by all the arcs (i, j) such that $i < j$. Besides, $\mathcal{H}_{\mathcal{I}}(\sigma)$ is weighted. The weight of the arc (i, j) is given by:

$$c_{i,j} = Min_{r \in R}\{ \sum_{k=i+1}^{j} d_k^r + \sum_{k=i+1}^{j-1} t_{k,k+1}^r + t_{j,i+1}^r \}$$

$c_{i,j}$ corresponds to the minimum time required to perform the subsequence $(i + 1, \ldots, j)$. Any value of r

Algorithm 2 Minmax

INPUT ($\mathcal{I}$,σ) where $\mathcal{I}$ is an instance of the SDRALBP-2 problem and σ is a giant sequence respecting precedence constraints. We suppose without loss of generality that $\sigma = \{1, 2, ..., n\}$
OUTPUT S: An optimal solution (with the minimal cycle time C^*) respecting σ

```
1: Build the graph H_I(σ)
2: L_0 := {(0,0)}
3: for t=1 to n do
4:    L_t := ∅
5: end for
6: for t=0 to n-1 do
7:    for all i/(t,i) ∈ A (Propagate labels from L_t) do
8:       for all (a_t, b_t) ∈ L_t do
9:          if (b_t < s_max − 1 or i = n) then
10:            (a_i, b_i) := (Max{a_t, c_{t,i}}, b_t + 1)
11:            if (a_i, b_i) is not dominated by an element of L_i then
12:               L_i := L_i ∪ {(a_i, b_i)}
13:               if (a_i, b_i) dominates some element (a'_i, b'_i) ∈ L_i then
14:                  L_i := L_i \ {(a'_i, b'_i)}
15:               end if
16:            end if
17:         end if
18:      end for
19:   end for
20: end for
21: if L_n ≠ ∅ then
22:    C* := Min_{(a_i,b_i)∈L_n}(a_i)
23:    Decode the path of cost C* to build S
24: end if
```

that gives this min represents the type of robot assigned to this station. The construction of the auxiliary graph can be performed within time in $O(n^3.n_r)$.

The optimal solution among the solutions satisfying a giant sequence σ can be obtained by computing a path from 0 to n in $\mathcal{H}_{\mathcal{I}}(\sigma)$ that minimises the maximum weight of an arc **using no more than s_{max} arcs**. In graph theory, the problem is known as a *bottleneck path* or *min-max path* (Chechik, Kaplan, Thorup, Zamir & Zwick 2016) which consists in finding a path between a pair of vertices such as the weight of an arc of maximum weight is minimized. This problem is polynomial. Since the path should not exceed s_{max} arcs, we are dealing with a constrained min-max path which can also be computed in polynomial time thanks to Algorithm 2 that we call minmax. It could be seen as an adaptation of Bellman-Ford algorithm to solve the problem of finding a min-max path constrained not to exceed s_{max} arcs in the graph $\mathcal{H}_{\mathcal{I}}(\sigma)$. It uses a set of labels L_i for node i. Every label $l = (a, b)$ in L_i corresponds to a path (partial solution) between 0 and i where a denotes the cycle time used by the path represented by the label l and b denotes the number of stations used by this path (i.e. the number of arcs in the path).(a, b) is said to be dominated by (a', b') if $a' \leq a$ and $b' \leq b$. The dominance rule limits the number of labels per node to s_{max}.

Algorithm 2 starts with fictitious node 0 labelled $L_0 := \{(0, 0)\}$ and continues with the other nodes following the giant sequence. For every node t and every label $(a_t, b_t) \in L_t$, the algorithm explores every outgoing arc (t, i) and tries to propagate it (i.e add a label to the list of labels of node i denoted L_i) if $(Max\{a_t, c_{t,i}\}, b_t + 1)$ is not dominated by a label of L_i. If so, the label $(Max\{a_t, c_{t,i}\}, b_t + 1)$ is added to L_i and all labels dominated by $(Max\{a_t, c_{t,i}\}, b_t + 1)$ are deleted from L_i. The min-max path cost (cycle time) is stored in C^*. The path is decoded by creating a station for each arc $c_{i,j}$ which is part of the path. The subsequence $(i+1, i+2, \ldots, j)$ is assigned to this station. The type of robot selected for this station is any type of robot minimizing the workload induced by this subsequence. The algorithm runs in $O(n^4 + n^3.n_r)$. The obtained path represents an optimal balancing (and robot selection) solution.

6.3 An hybrid metaheuristic

The minmax algorithm could be used to solve the problem optimally given a giant sequence of operations. To solve the SDRALBP-2, we should deter-

mine the best giant sequence.

We use metaheuristic frameworks in order to explore the space of giant sequences. The minmax algorithm is then used as a decoding algorithm and an evaluation function.

We use Iterated local search (ILS). A complete description of ILS algorithm could be found in (Lourenço, Martin & Stützle 2010). A local search is iterated a number of times starting from a perturbation of the best know solution. The stopping criterion of the local search is the number of solutions visited while the stopping criterion of the ILS is the number of iterations of local search. The neighbours are chosen randomly. The neighbourhood move used is described below. The perturbation stands for applying the move three times.

We choose an insertion move that respects precedence constraints. A random operation is chosen on the giant sequence then it is re-inserted between the last operation that precedes it and the first operation that succeeds it with respect to precedence constraints.

7 EXPERIMENTATION

The experiments presented in this section are preliminary. Other experiments are being held. The instances are taken from (Janardhanan et al. 2019). They are partitioned within three classes:

- Instances with null setup times: they correspond to the same instances of (Gao et al. 2009).
- Instances with low setup times: they correspond to the instances of (Gao et al. 2009) to which setup times are added and generated randomly and uniformly within $[0, 0.25 * min_{i,r} d_{i,r}]$.
- Instances with high setup times: they correspond to the instances of (Gao et al. 2009) to which setup times are added and generated randomly and uniformly within $[0, 0.75 * min_{i,r} d_{i,r}]$.

We have $s_{max} = n_r$ for these instances. The only method available in literature for SDRALBP-2 is suggested in (Janardhanan et al. 2019). Their method gives an upper bound for ours because they don't allow themselves to use a type of robot several times in different stations. Our method can also apply for RALBP (SDRALBP with null setup times) by considering that the setup times are null. Since the RALBP is much more investigated than SDRALBP, we also compare our method on instances with null setup times with the better-performing algorithms from RALBP literature: (Nilakantan et al. 2015), (Borba et al. 2018). The RALBP does not raise any sequencing problem, it is only concerned with the balancing and robot selection decisions. Even if our method is not dedicated to RALBP, this comparison can give an idea about the performance of our method. Table 5 shows the experiments on instances with null setup times (RALBP) and table 6 shows experiments on instances with low and high setup times. In those tables, references denote the value of the cycle time obtained in those references while C and C_{10} denote the cycle time obtained respectively with a stochastic local search where 10 000 neighbours are visited and an ILS of 10 iterated local searches where 10 000 neighbours are visited in each. We do not have the results of (Janardhanan et al. 2019) for instances with $n \geq 89$.

CPU times needed to compute C are comparable with those from (Janardhanan et al. 2019) for $n = 11..70$ since they are generally below 50 seconds (i.e less than 5 microseconds necessary to run the minmax algorithm). The CPU time grows then drastically while increasing the value of n. The CPU times remain very reasonable up to $n = 148$. For $n = 148$ and $n_r = 29$, 0.1 seconds is needed to perform a single minmax run. For instances with $n = 297$, this value can rise up to 1.7 second for biggest n_r. In order to be used efficiently in a metaheuristic, the minmax algorithm should be applied many times to explore as much neighbors as needed. The minmax algorithm is very fast for instances with $n = 1..70$ (small instances) and reasonable for instances with $n = 89..148$ (medium and big instances). However for very big instances $n = 297$ with high number of possible robots, the CPU times of the minmax algorithm exceeds one second which limits the number of neighbours that can be visited in some metaheuristic. For this reason, results for instances with $n = 297$ and $n = 148$ (C_{10}) are not presented in this preliminary research.

Table 5 shows that even if our method is not dedicated to RALBP, it can retrieve most optimal values obtained by (Borba et al. 2018) and is far better than (Nilakantan et al. 2015) which is dedicated to RALBP.

Tables 5 and 6 show the benefit of the assumption allowing the use of the same robot type in multiple stations. Indeed, the cycle time C (and C_{10}) is much smaller than the cycle time from (Janardhanan et al. 2019) for most instances. C_{10} is better than C at the expense of 10 times higher CPU time. The tables also show that the consideration of setup times has a real impact on the cycle time. This justifies that the sequence-dependent setup times cannot be negligible and should be taken into consideration in the modelling/optimization step of the Robotic Assembly Line Balancing Problem.

Table 5 – Instances with null setup times

n	s_{max}, n_r	(Nilakantan et al. 2015)	(Borba et al. 2018)	(Janardhanan et al. 2019)	C	C_{10}
11	4	-	-	128	126	126
25	3	503	503*	503	**503**	**503**
	4	327	291*	327	294	**291**
	6	200	194*	213	195	**194**
	9	110	109*	121	**109**	**109**
35	4	341	341*	449	342	**341**
	5	332	329*	344	**329**	**329**
	7	211	201*	222	**201**	**201**
	12	103	93*	112	98	**93**
53	5	449	449*	559	**449**	**449**
	7	294	283*	320	284	**283**
	10	221	203*	239	213	**203**
	14	142	134*	162	137	**134**
70	7	430	391*	448	401	392
	10	264	233*	271	238	234
	14	194	170*	201	183	176
	19	140	121*	152	129	126
89	8	460	436*	-	446	445
	12	320	296*	-	309	301
	16	219	205*	-	211	207
	21	170	156*	-	164	161
111	9	523	468	-	491	472
	13	321	275	-	295	287
	17	240	212	-	230	224
	22	182	154	-	173	166
148	10	593	550	-	583	-
	14	419	351	-	376	-
	21	273	225	-	244	-
	29	189	154	-	171	-

*: optimal solution

Table 6 – Instances with low and high setup times

Instance		Low Setup			High Setup		
n	$s_{max}(= n_r)$	(Janardhanan et al. 2019)	C	C_{10}	(Janardhanan et al. 2019)	C	C_{10}
11	4	137	137	137	152	152	151
25	3	516	536	535	579	584	579
	4	346	303	303	380	343	343
	6	227	203	198	242	216	214
	9	131	116	116	142	125	121
35	4	462	352	352	494	376	374
	5	355	335	335	392	368	365
	7	237	208	208	261	225	224
	12	118	100	100	131	113	113
53	5	574	471	461	619	508	486
	7	334	286	286	359	319	308
	10	256	223	213	276	244	237
	14	170	146	143	185	155	155
70	7	469	426	408	507	466	448
	10	282	252	246	309	270	266
	14	211	189	182	233	206	202
	19	158	135	131	175	145	144
89	8	-	465	458	-	491	491
	12	-	318	308	-	344	344
	16	-	224	220	-	240	238
	21	-	166	163		184	181
111	9	-	517	495	-	541	521
	13	-	306	301	-	329	228
	17	-	233	233	-	257	255
	22	-	178	172	-	193	191
148	10	-	610	-	-	685	-
	14	-	391	-	-	431	-
	21	-	256	-	-	289	-
	29	-	182	-	-	202	-

8 CONCLUSION AND PERSPECTIVES

In this paper, a solvable polynomial case for the sequence-dependent robotic assembly line balancing problem is derived. It is solved thanks to a minmax path algorithm. We also give an interesting use case of the minmax proposed algorithm: embedding it on a metaheuristic. This findings introduce a new solution encoding in a metaheuristic that searches in a much smaller space than traditional encodings. The first experimentation are very promising. Several directions could be taken following this research :

- Reducing the CPU time of the minmax algorithm by limiting the number of labels thanks to upper bounds.
- Making more experiments on different sets of instances.
- Using more sophisticated metaheuristics embedding the minmax algorithm.
- Adapting the method for the case where each robot can be used at most once.
- Deriving other use cases of the minmax algorithm.

ACKNOWLEDGMENTS

The authors acknowledge the support received from the *Agence Nationale de la Recherche* of the French government through the program "Investissements d'Avenir"(16-IDEX-0001 CAP 20-25).

References

Andres, C., Miralles, C. & Pastor, R. (2008). Balancing and scheduling tasks in assembly lines with sequence-dependent setup times, *European Journal of Operational Research* **187**(3): 1212–1223.

Battaïa, O. & Dolgui, A. (2013). A taxonomy of line balancing problems and their solutionapproaches, *International Journal of Production Economics* **142**(2): 259–277.

Borba, L., Ritt, M. & Miralles, C. (2018). Exact and heuristic methods for solving the robotic assembly line balancing problem, *European Journal of Operational Research* **270**(1): 146–156.

Borisovsky, P. A., Delorme, X. & Dolgui, A. (2014). Balancing reconfigurable machining lines via a set partitioning model, *International Journal of Production Research* **52**(13): 4026–4036.

Chechik, S., Kaplan, H., Thorup, M., Zamir, O. & Zwick, U. (2016). Bottleneck paths and trees and deterministic graphical games, *33rd Symposium on Theoretical Aspects of Computer Science (STACS 2016)*, Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

Çil, Z. A., Mete, S. & Ağpak, K. (2016). A goal programming approach for robotic assembly line balancing problem, *IFAC-PapersOnLine* **49**(12): 938–942.

Gao, J., Sun, L., Wang, L. & Gen, M. (2009). An efficient approach for type ii robotic assembly line balancing problems, *Computers & Industrial Engineering* **56**(3): 1065–1080.

Janardhanan, M. N., Li, Z., Bocewicz, G., Banaszak, Z. & Nielsen, P. (2019). Metaheuristic algorithms for balancing robotic assembly lines with sequence-dependent robot setup times, *Applied Mathematical Modelling* **65**: 256–270.

Lahrichi, Y., Grangeon, N., Deroussi, L. & Norre, S. (2020). A new split-based hybrid metaheuristic for the reconfigurable transfer line balancing problem, *International Journal of Production Research* pp. 1–18.

Levitin, G., Rubinovitz, J. & Shnits, B. (2006). A genetic algorithm for robotic assembly line balancing, *European Journal of Operational Research* **168**(3): 811–825.

Lourenço, H. R., Martin, O. C. & Stützle, T. (2010). Iterated local search: Framework and applications, *Handbook of metaheuristics*, Springer, pp. 363–397.

Martino, L. & Pastor, R. (2010). Heuristic procedures for solving the general assembly line balancing problem with setups, *International Journal of Production Research* **48**(6): 1787–1804.

Nilakantan, J. M., Ponnambalam, S. G., Jawahar, N. & Kanagaraj, G. (2015). Bio-inspired search algorithms to solve robotic assembly line balancing problems, *Neural Computing and Applications* **26**(6): 1379–1393.

Rubinovitz, J., Bukchin, J. & Lenz, E. (1993). Ralb–a heuristic algorithm for design and balancing of robotic assembly lines, *CIRP annals* **42**(1): 497–500.

Yoosefelahi, A., Aminnayeri, M., Mosadegh, H. & Ardakani, H. D. (2012). Type ii robotic assembly line balancing problem: An evolution strategies algorithm for a multi-objective model, *Journal of Manufacturing Systems* **31**(2): 139–151.