

AMERICAN UNIVERSITY OF BEIRUT

Models for the Assembly Line Balancing Problem

by  
Antoinette Mouawad

A thesis  
submitted in partial fulfillment of the requirements  
for the degree of Master of Science in Business Analytics  
of the Olayan School of Business  
at the American University of Beirut

Beirut, Lebanon  
June 2020

# AMERICAN UNIVERSITY OF BEIRUT

## Models for the Assembly Line Balancing Problem

by  
Antoinette Mouawad

Approved by:

*Krzysztof Fleszar*

---

Dr. Krzysztof Fleszar, Professor

Advisor

Olayan School of Business

*Khalil Hindi*

---

Dr. Khalil S. Hindi, Professor

Member of Committee

Olayan School of Business

Date of thesis defense: June 22, 2020

# AMERICAN UNIVERSITY OF BEIRUT

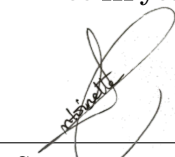
## THESIS, DISSERTATION, PROJECT RELEASE FORM

Student Name: Mouawad Antoinette  
Last First Middle

Master's Thesis       Master's Project       Doctoral Dissertation

I authorize the American University of Beirut to: (a) reproduce hard or electronic copies of my thesis, dissertation, or project; (b) include such copies in the archives and digital repositories of the University; and (c) make freely available such copies to third parties for research or educational purposes.

I authorize the American University of Beirut, to: (a) reproduce hard or electronic copies of it; (b) include such copies in the archives and digital repositories of the University; and (c) make freely available such copies to third parties for research or educational purposes after: **One \_\_\_ year from the date of submission of my thesis, dissertation or project.**  
**Two \_\_\_ years from the date of submission of my thesis , dissertation or project.**  
**Three \_\_\_ years from the date of submission of my thesis , dissertation or project.**

  
\_\_\_\_\_  
Signature

2020-07-06  
\_\_\_\_\_  
Date

This form is signed when submitting the thesis, dissertation, or project to the University Libraries

# Acknowledgements

I am really grateful to submit this work that couldn't have been possible without the continuous guidance, help and patience of Professor Krzysztof Fleszar. The opportunity of doing a Masters degree at a prestigious university like the American University of Beirut is an honor for me and I remain forever grateful for it.

# An Abstract of the Thesis of

Antoinette Mouawad for Master of Business Analytics  
Major: Business Analytics

Title: Models for the Assembly Line Balancing Problem

We review the literature models for the Assembly Line Balancing Problem (ALBP), present multiple variants of models for ALBP, and propose a new mixed-integer linear model that uses a network flow formulation to represent the load of each station. We use the network compression algorithm proposed by Brandão and Pedroso (2016) for bin packing to reduce the number of variables and constraints in the network flow formulation. We perform computational experiments to test the models and compare their performance on standard benchmark problem instances.

# Contents

<b>Acknowledgements</b>	<b>v</b>
<b>Abstract</b>	<b>vi</b>
<b>1 Introduction</b>	<b>1</b>
<b>2 Literature Review</b>	<b>3</b>
<b>3 Model Parameters</b>	<b>4</b>
<b>4 MILP Models</b>	<b>6</b>
4.1 Decision Variables . . . . .	6
4.2 Occurrence constraints . . . . .	6
4.3 Cycle time constraints . . . . .	6
4.4 Precedence constraints . . . . .	7
4.5 Objective function . . . . .	10
4.6 Dynamic latest station reduction . . . . .	14
4.7 Reduction of the number of precedence constraints . . . . .	14
4.8 Precedence network reversion . . . . .	15
<b>5 Our Model</b>	<b>16</b>
<b>6 Computational Experiments</b>	<b>21</b>
<b>7 Conclusion</b>	<b>27</b>
<b>A Abbreviations</b>	<b>28</b>

# List of Figures

- 4.1 Visualization of stations on which tasks  $i$  and  $h$  can be scheduled, where  $i$  is a predecessor of  $h$  and  $E_i = 3$ ,  $L_i = 8$ ,  $E_h = 6$ , and  $L_h = 12$  . . . . . 13
- 4.2 Scanning algorithm . . . . . 15
  
- 5.1 Network flow formulation algorithm . . . . . 18
- 5.2 Example of ALBP problem with  $n = 7$  . . . . . 18
- 5.3 Station 1 represented as a set of nodes or states . . . . . 19
- 5.4 Station 1 represented as a set of nodes or states . . . . . 19
  
- 6.1 Count of optimal solutions in function of variants types . . . . . 23
- 6.2 Performance comparison between models . . . . . 26

# List of Tables

4.1	Precedence constraints expressions for $l_{ih} = 0$ . . . . .	11
4.2	Precedence constraints expressions for $l_{ih} = 1$ . . . . .	12
5.1	Earliest and latest stations of tasks . . . . .	19
6.1	Precedence constraints recap . . . . .	22
6.2	Instances not solved by the MILP variants . . . . .	23
6.3	Results of best performing model (Precedence type = 3) . . . . .	24
6.4	Results of best performing model (Precedence type = 10) . . . . .	25
6.5	Results of the network flow model (Precedence type = 3) . . . . .	25
6.6	Results of the network flow model (Precedence type = 10) . . . . .	26



# Chapter 1

## Introduction

An *Assembly Line* is a mass production system that consists of a number of workstations (*station*, hereafter) arranged along a linear transport mechanism that is usually a conveyor. Work pieces are fed to the first station of the line at a predetermined constant feed rate (time interval of *cycle time*), and are moved from one station to the next. At each station, certain tasks (*task*, hereafter) are repeatedly performed. By definition, a task cannot be divided between two or more stations. In addition, tasks are subject to precedence constraints whereby a given task cannot start before the processing of all its predecessors. A work piece is completed when it leaves the last station, after all tasks required for its production have been completed.

The *Assembly Line Balancing Problem* (ALBP) is to optimally partition the tasks among the stations with respect to some objective. The two conflicting objectives are minimizing the total cost of resources used by the assembly line and maximizing the production rate achieved by the line. In the simplest case, the former is achieved through minimizing the number of stations, given a fixed cycle time (this is the case of *Simple Assembly Line Balancing Problem - type 1* (SALBP-1)), while the latter is achieved through minimizing the cycle time, given a fixed number of stations (this is the case of *Simple Assembly Line Balancing Problem - type 2* (SALBP-2)). SALBP-2 is equivalent to maximizing the work time among all stations. We define the production rate as below:

$$\text{Production rate} = \frac{\text{One completed work piece}}{\text{Cycle time}} \quad (1.1)$$

Our research focuses on the *Simple Assembly Line Balancing Problem - type 1* (SALBP-1), where the objective is to minimize the total number of stations needed to accomplish all tasks for a given production cycle time and precedence constraints among tasks. While most of the literature for this problem focuses on heuristics and meta-heuristics methods (see for example Sivasankaran and Shahabudeen (2014)), we will focus our attention on mathematical models for SALBP-1.

We will start with a review of Mixed Integer Linear Programming (MILP) models proposed in the literature for SALBP-1 so far. Then we will conduct experimental investigation of the performance of the different models, determine the best performing ones, and try to determine the features that improve the performance of the models. The performance will be measured in terms of speed to convergence to optimality as well as in terms of computational efforts. Finally, we will propose a new model for SALBP-1 that outperforms the existing models.

# Chapter 2

## Literature Review

In this chapter, we discuss briefly the literature where MILP models for ALBP have been introduced, without presenting the models in detail. The following chapters will present the various types of constraints and all possible resulting models.

First MILP models were introduced by Bowman (1960) and later modified by White (1961). They used binary variables  $x_{ij} = 1$  if task  $i$  is assigned to station  $j$ , 0 otherwise. No other variables were introduced. The objective was based on the observation that the makespan can be between a lower bound  $m_{\min}$  and an upper bound  $m_{\max}$  and that in order to minimize the makespan, it is sufficient to penalize scheduling tasks without successors on stations from  $m_{\min}$  to  $m_{\max}$ .

Patterson and Albracht (1975) introduced a MILP with a different objective function. They introduced a dummy finish task where all the tasks with no successors were made to be predecessors of the dummy finish task. Then, their objective minimized the station number where the dummy finish task is scheduled (in fact, their objective was a maximization, but effectively it worked as minimizing the station number of the dummy finish task).

Moreover, Patterson and Albracht (1975) introduced the concept of earliest  $E_i$  and latest  $L_i$  stations for task  $i$  that can be used to limit the number of binary variables. Furthermore, they used a different precedence constraint than Bowman (1960) and White (1961) (discussed later).

Baybars (1986) presented a review of the above MILP models as well as other exact methods for SALBP-1 and SALBP-2 and compared their computational performance.

Mathematical models for ALBP with stronger precedence constraints were proposed by Aghezzaf and Artiba (1995) and later by Ritt and Costa (2018).

# Chapter 3

## Model Parameters

Before we introduce any models, let us first define all necessary parameters:

- $n$  = total number of tasks
- $m^*$  = optimal number of stations needed
- $m_{\min}$  = lower bound of stations needed (known)
- $m_{\max}$  = upper bound of stations needed (known)
- $t_i$  = process time of task  $i$
- $c$  = cycle time
- $t_{\min} = \min\{t_i : i \in \{1, \dots, n\}\}$
- $t_{\max} = \max\{t_i : i \in \{1, \dots, n\}\}$
- $R = \{(h, i) \in \{1, \dots, n\}^2 : h \text{ is an immediate predecessor of } i\}$  = immediate precedence relationships
- $P(i) = \{h : (h, i) \in R\}$  = immediate predecessors of task  $i$
- $P_a(i)$  = all predecessors of task  $i$
- $S(i) = \{h : (i, h) \in R\}$  = immediate successors of task  $i$
- $S_a(i)$  = all successors of task  $i$
- $F = \{i : S(i) = \emptyset\}$  = tasks with no successors
- $E_i$  = earliest station of task  $i$
- $L_i$  = latest station of task  $i$

- $l_{ih}$  = station lag between tasks  $i$  and  $h$ , i.e., the minimum difference between station number for task  $i$  and station number for task  $h$

Patterson and Albracht (1975) calculated earliest (resp., latest) stations for all tasks based on the total time of an activity and all its predecessors (resp., successors):

$$E_i = \left\lceil \left( t_i + \sum_{h \in P_a(i)} t_h \right) / c \right\rceil \quad (3.1)$$

$$L_i = m_{\max} + 1 - \left\lceil \left( t_i + \sum_{h \in S_a(i)} t_h \right) / c \right\rceil \quad (3.2)$$

In general, both  $E_i$  and  $L_i$  are based on calculating lower bounds on a subset of activities  $\{i\} \cup P_a(i)$  and  $\{i\} \cup S_a(i)$ , respectively, so the above formulas can be equivalently expressed as:

$$E_i = \text{LB}(\{i\} \cup P_a(i)) \quad (3.3)$$

$$L_i = m_{\max} + 1 - \text{LB}(\{i\} \cup S_a(i)) \quad (3.4)$$

where  $\text{LB}(A)$  is a lower bound on the problem with a subset of tasks  $A$ . In the above case,  $\text{LB}(A) = \text{LB}_1(A)$ , which is calculated as:

$$\text{LB}_1(A) = \left\lceil \left( \sum_{i \in A} t_i \right) / c \right\rceil \quad (3.5)$$

In addition to  $\text{LB}_1(A)$ , we can use other lower bounds for SALBP-1. Following Sewell and Jacobson (2012) we will define  $\text{LB}(A)$  as the maximum of  $\text{LB}_1(A)$ ,  $\text{LB}_2(A)$ , and  $\text{LB}_3(A)$ , where the later two lower bounds are defined as follows:

$$\text{LB}_2(A) = \left| \left\{ i \in A : t_i > \frac{c}{2} \right\} \right| + \left\lceil \frac{1}{2} \left| \left\{ i \in A : t_i = \frac{c}{2} \right\} \right| \right\rceil \quad (3.6)$$

$$\begin{aligned} \text{LB}_3(A) = & \left| \left\{ i \in A : t_i > \frac{2c}{3} \right\} \right| + \\ & + \left\lceil \frac{2}{3} \left| \left\{ i \in A : t_i = \frac{2c}{3} \right\} \right| + \frac{1}{2} \left| \left\{ i \in A : \frac{c}{3} < t_i < \frac{2c}{3} \right\} \right| + \frac{1}{3} \left| \left\{ i \in A : t_i = \frac{c}{3} \right\} \right| \right\rceil \end{aligned} \quad (3.7)$$

# Chapter 4

## MILP Models

### 4.1 Decision Variables

Our decision variables are all binary variables defined as below:

- $x_{ij} = 1$  if task  $i$  is assigned to station  $j$ ,  $= 0$  otherwise, for all  $i \in \{1, \dots, n\}$  and  $j \in \{E_i, \dots, L_i\}$
- $y_j = 1$  if station  $j$  is used,  $= 0$  otherwise, for  $j \in \{m_{\min} + 1, \dots, m_{\max}\}$

### 4.2 Occurrence constraints

The occurrence constraint ensures that each task is assigned to exactly one station:

$$\sum_{j=E_i}^{L_i} x_{ij} = 1 \quad \forall i \in \{1, \dots, n\} \quad (\text{OCCUR})$$

The constraint was first used in the model of Bowman (1960) and White (1961) with the summation over all stations, i.e., from 1 to  $m_{\max}$ . Later, Patterson and Albracht (1975) restricted indexing to only stations to which task  $i$  could be assigned, i.e., from  $E_i$  to  $L_i$ .

### 4.3 Cycle time constraints

The following constraints ensure the workload on each station does not exceed the cycle time:

$$\sum_{i:j \in \{E_i, \dots, L_i\}} t_i x_{ij} \leq c \quad \forall j \in \{1, \dots, m_{\max}\} \quad (\text{CYCLE})$$

## 4.4 Precedence constraints

Precedence constraints are defined for all pairs of tasks  $(i, h) \in R$ , where  $i$  is an immediate predecessor of  $h$ . They must ensure that the station of  $i$  has an index smaller than or equal to the index of the station of  $h$ . Note that if  $t_i + t_h > c$ , i.e., if the two tasks cannot share a station, the station index of  $i$  must be less than the station index of  $h$  by at least one. In general, for  $(i, h) \in R$ , we can define the station lag  $l_{ih} = 1$  if  $t_i + t_h > c$  and 0 otherwise. Then, the precedence constraint between tasks  $(i, h) \in R$  must ensure that the station index of  $i$  is less than the station index of  $h$  by at least  $l_{ih}$ . Note that this constraint can also be used when  $i$  is a transitive predecessor of  $h$ , in which case  $l_{ih}$  may be higher than 1.

If earliest station  $E_i$  and latest stations  $L_i$  are provided for each task, observe that we can assume that  $E_i + l_{ih} \leq E_h$  and  $L_i + l_{ih} \leq L_h$ , for  $(i, h) \in R$ . If these conditions are not satisfied,  $E_h$  can be increased and  $L_i$  decreased until the conditions are satisfied.

Patterson and Albracht (1975) observed that precedence constraint for  $(i, h) \in R$  needs to be defined only if  $L_i \geq E_h$ . We observe that this condition can be strengthened to  $L_i > E_h$ , because only in this case, task  $i$  could be scheduled after task  $h$ , which the precedence constraint should forbid. With lag  $l_{ih}$ , the condition is  $L_i + l_{ih} > E_h$ . We assume  $l_{ih} = 1$  if  $t_i + t_h > c$  and  $l_{ih} = 0$  otherwise.

We consider three different types of precedence constraints. The first, due to Patterson and Albracht (1975), is:

$$\sum_{j=E_i}^{L_i} jx_{ij} \leq \sum_{j=E_h}^{L_h} jx_{hj} \quad \forall (i, h) \in R, L_i > E_h \quad (4.1)$$

The left side calculates the station number of task  $i$  and the right side the station number of task  $h$ . A more general version with multipliers different than station numbers  $j$  were earlier used by Thangavelu and Shetty (1971).

Taking into account the station lag  $l_{ih}$ , a stronger variant of the above constraint can be proposed:

$$\sum_{j=E_i}^{L_i} jx_{ij} + l_{ih} \leq \sum_{j=E_h}^{L_h} jx_{hj} \quad \forall (i, h) \in R, L_i + l_{ih} > E_h \quad (\text{PREC0})$$

The constraint can be further strengthened by observing that the left side value can be at most  $L_i + l_{ih}$ , so the variable multipliers  $j$  on the right side can be replaced by  $\min\{j, L_i + l_{ih}\}$ . Similarly, the right side value is at least  $E_h$ , so adjusting for  $l_{ih}$ , the multipliers  $j$  on the left side can be replaced by  $\max\{j, E_h - l_{ih}\}$ . The resulting final variant of the constraint is:

$$\sum_{j=E_i}^{L_i} \max\{j, E_h - l_{ih}\}x_{ij} + l_{ih} \leq \sum_{j=E_h}^{L_h} \min\{j, L_i + l_{ih}\}x_{hj} \quad \forall (i, h) \in R, L_i + l_{ih} > E_h \quad (\text{PREC1})$$

The second type of precedence constraint was introduced in the model of Bowman (1960) and White (1961). Taking into account the earliest and latest stations of tasks, this constraint can be expressed as follows:

$$x_{hk} \leq \sum_{j=E_i}^k x_{ij} \quad \forall (i, h) \in R, L_i > E_h, \forall k \in \{E_h, \dots, L_i - 1\} \quad (4.2)$$

The interpretation of this constraint is as follows: if task  $h$  is assigned to station  $k$  (if the left side equals one), then task  $i$  that is a predecessor of  $h$  must be assigned to station  $k$  or earlier (the right side must also equal one).

The third type of precedence constraint was first used in a model of Aghezzaf and Artiba (1995) and later by Ritt and Costa (2018). Taking into account the earliest and latest stations of tasks, this constraint can be expressed as follows:

$$\sum_{j=E_h}^k x_{hj} \leq \sum_{j=E_i}^k x_{ij} \quad \forall (i, h) \in R, L_i > E_h, \forall k \in \{E_h, \dots, L_i - 1\} \quad (4.3)$$

The interpretation of this constraint is similar to the previous constraint: if task  $h$  is assigned to station  $k$  or earlier, then task  $i$  that is a predecessor of  $h$  must be assigned to station  $k$  or earlier.

Improved versions of the above two types of constraints that take into account the station lag  $l_{ih}$  are:

$$x_{hk} \leq \sum_{j=E_i}^{k-l_{ih}} x_{ij} \quad \forall (i, h) \in R, L_i + l_{ih} > E_h, \forall k \in \{E_h, \dots, L_i - 1 + l_{ih}\} \quad (\text{PREC2-E})$$

$$\sum_{j=E_h}^k x_{hj} \leq \sum_{j=E_i}^{k-l_{ih}} x_{ij} \quad \forall (i, h) \in R, L_i + l_{ih} > E_h, \forall k \in \{E_h, \dots, L_i - 1 + l_{ih}\} \quad (\text{PREC3-E})$$

Exploiting the directional symmetry of the problem, we can introduce the following two new types of precedence constraints. The first is:

$$x_{ik} \leq \sum_{j=k}^{L_h} x_{hj} \quad \forall (i, h) \in R, L_i > E_h, \forall k \in \{E_h + 1, \dots, L_i\} \quad (4.4)$$

The interpretation of this constraint is as follows: if task  $i$  is assigned to station  $k$  (if left hand side equals one), then task  $h$  that is a successor of  $i$  must be assigned to station  $k$  or later.

The second new type of precedence constraint is:

$$\sum_{j=k}^{L_i} x_{ij} \leq \sum_{j=k}^{L_h} x_{hj} \quad \forall (i, h) \in R, L_i > E_h, \forall k \in \{E_h + 1, \dots, L_i\} \quad (4.5)$$



The interpretation of this constraint is as follows: if task  $i$  is assigned to station  $k$  or later, then task  $h$  that is a successor of  $i$  must be assigned to station  $k$  or later.

Taking into account station lags  $l_{ih}$ , the constraints can be written as follows:

$$x_{ik} \leq \sum_{j=k+l_{ih}}^{L_h} x_{hj} \quad \forall (i, h) \in R, L_i + l_{ih} > E_h, \forall k \in \{E_h + 1 - l_{ih}, \dots, L_i\} \quad (\text{PREC2-L})$$

$$\sum_{j=k}^{L_i} x_{ij} \leq \sum_{j=k+l_{ih}}^{L_h} x_{hj} \quad \forall (i, h) \in R, L_i + l_{ih} > E_h, \forall k \in \{E_h + 1 - l_{ih}, \dots, L_i\} \quad (\text{PREC3-L})$$

The advantage of using precedence constraints (PREC0) or (PREC1) is that they result in much fewer constraints. However, as pointed out by Ritt and Costa (2018), constraints (PREC3-E) dominate constraints (PREC0), (PREC1), and (PREC2-E) in terms of the strength of the linear relaxation. By symmetry, constraints (PREC3-L) will dominate constraints (PREC0), (PREC1), and (PREC2-L).

Although the constraints PREC3-E and PREC3-L dominate the constraints PREC0, PREC1, PREC2-E and PREC2-L in terms of the strength of the linear relaxation, however they will introduce more constraints compared to PREC0 and PREC1, and have more non-zero elements. This is why, we will be doing a full factorial analysis to check what trade-off will have a greater impact on the performances of the models.

Let precedence constraints PREC2-BETTER (resp, PREC3-BETTER) be defined as follows: if  $E_h - E_i \leq L_h - L_i$ , then constraints (PREC2-E) (resp., (PREC3-E)) will have be chosen, otherwise constraints (PREC2-L) (resp., (PREC3-L)) will be chosen. The criterion chooses the constraint type that results in fewer non-zero elements in the constraint matrix. In case of a tie, the E variant is chosen.

Additionally, let precedence constraints PREC2-BOTH (resp, PREC3-BOTH) be equivalent to using both constraints (PREC2-E) and (PREC2-L) (resp., (PREC3-E) and (PREC3-L)).

Using occurrence constraints (OCCUR), the right side of constraints (PREC3-E) can be modified, so that the resulting constraint is:

$$\sum_{j=E_h}^k x_{hj} \leq 1 - \sum_{j=k-l_{ih}+1}^{L_i} x_{ij} \quad \forall (i, h) \in R, L_i + l_{ih} > E_h, \forall k \in \{E_h, \dots, L_i - 1 + l_{ih}\} \quad (4.6)$$

The interpretation of this constraint is: if task  $h$  is on station  $k$  or earlier, than task  $i$  must not be on station  $k - l_{ih} + 1$  or later. Taking all variables to the left

side gives:

$$\sum_{j=E_h}^k x_{hj} + \sum_{j=k-l_{ih}+1}^{L_i} x_{ij} \leq 1 \quad \forall (i, h) \in R, L_i + l_{ih} > E_h, \forall k \in \{E_h, \dots, L_i - 1 + l_{ih}\}$$

(PREC3-NEW1)

The constraint will not allow task  $h$  to be on station  $k$  or earlier and at the same time task  $i$  to be on station  $k - l_{ih} + 1$  or later.

Alternatively, using occurrence constraints (OCCUR), the left side of constraints (PREC3-E) can be modified, so that the resulting constraint is:

$$1 - \sum_{j=k+1}^{L_h} x_{hj} \leq \sum_{j=E_i}^{k-l_{ih}} x_{ij} \quad \forall (i, h) \in R, L_i + l_{ih} > E_h, \forall k \in \{E_h, \dots, L_i - 1 + l_{ih}\}$$

(4.7)

The interpretation of this constraint is: if task  $h$  is not on station  $k + 1$  or later, than task  $i$  must be on station  $k - l_{ih}$  or earlier. Taking all variables to the left side gives:

$$\sum_{j=k+1}^{L_h} x_{hj} + \sum_{j=E_i}^{k-l_{ih}} x_{ij} \geq 1 \quad \forall (i, h) \in R, L_i + l_{ih} > E_h, \forall k \in \{E_h, \dots, L_i - 1 + l_{ih}\}$$

(PREC3-NEW2)

The two additional constraints added PREC3-NEW1 and PREC3-NEW2 might have an impact on the performance of the models in terms of creating better cuts to help converge faster to an optimal solution.

To illustrate the difference between all above defined precedence constraints, consider an example pair of tasks  $i$  and  $h$ , where  $i$  is a predecessor of  $h$  and  $E_i = 3$ ,  $L_i = 8$ ,  $E_h = 6$ , and  $L_h = 12$ . In this example, the precedence constraint existence condition ( $L_i + l_{ih} > E_h$  for  $l_{ih} = 0$  or  $1$ ) is satisfied. Figure 4.1 illustrates visualizes the stations to which each of the two tasks can be assigned. Tables 4.1 and 4.2 show the different precedence constraints for the pair of tasks.

## 4.5 Objective function

The objective of the problem is to find the minimum number of stations  $m^*$ , which must be between  $m_{\min}$  and  $m_{\max}$ . If  $m_{\min} = m_{\max}$ , then the problem reduces to a feasibility problem, checking if a solution with  $m_{\min} = m_{\max}$  exists, and no objective needs to be defined. If  $m_{\min} < m_{\max}$ , one of the following objectives is used.

The number of used stations can be minimized using objective:

$$\min z \tag{OBJ1}$$

Precedence type	Precedence constraint expression for $l_{ih} = 0$
PREC0	$\sum_{j=3}^8 jx_{ij} \leq \sum_{j=6}^{12} jx_{hj}$
PREC1	$\sum_{j=3}^8 \max\{j, 6\}x_{ij} \leq \sum_{j=6}^{12} \min\{j, 8\}x_{hj}$
PREC2-E	$x_{hk} \leq \sum_{j=3}^k x_{ij} \quad \forall k \in \{6, 7\}$
PREC3-E	$\sum_{j=6}^k x_{hj} \leq \sum_{j=3}^k x_{ij} \quad \forall k \in \{6, 7\}$
PREC2-L	$x_{ik} \leq \sum_{j=k}^{12} x_{hj} \quad \forall k \in \{7, 8\}$
PREC3-L	$\sum_{j=k}^8 x_{ij} \leq \sum_{j=k}^{12} x_{hj} \quad \forall k \in \{7, 8\}$
PREC3-NEW1	$\sum_{j=6}^k x_{hj} + \sum_{j=k+1}^8 x_{ij} \leq 1 \quad \forall k \in \{6, 7\}$
PREC3-NEW2	$\sum_{j=k+1}^{12} x_{hj} + \sum_{j=3}^k x_{ij} \geq 1 \quad \forall k \in \{6, 7\}$

Table 4.1: Precedence constraints expressions for  $l_{ih} = 0$

Precedence type	Precedence constraint expression for $l_{ih} = 1$
PREC0	$\sum_{j=3}^8 jx_{ij} \leq \sum_{j=6}^{12} jx_{hj}$
PREC1	$\sum_{j=3}^8 \max\{j, 6\}x_{ij} \leq \sum_{j=6}^{12} \min\{j, 8\}x_{hj}$
PREC2-E	$x_{hk} \leq \sum_{j=3}^{k-1} x_{ij} \quad \forall k \in \{6, 7, 8\}$
PREC3-E	$\sum_{j=6}^k x_{hj} \leq \sum_{j=3}^{k-1} x_{ij} \quad \forall k \in \{6, 7, 8\}$
PREC2-L	$x_{ik} \leq \sum_{j=k+1}^{12} x_{hj} \quad \forall k \in \{6, 7, 8\}$
PREC3-L	$\sum_{j=k}^8 x_{ij} \leq \sum_{j=k+1}^{12} x_{hj} \quad \forall k \in \{6, 7, 8\}$
PREC3-NEW1	$\sum_{j=6}^k x_{hj} + \sum_{j=k}^8 x_{ij} \leq 1 \quad \forall k \in \{6, 7, 8\}$
PREC3-NEW2	$\sum_{j=k+1}^{12} x_{hj} + \sum_{j=3}^{k-1} x_{ij} \geq 1 \quad \forall k \in \{6, 7, 8\}$

Table 4.2: Precedence constraints expressions for  $l_{ih} = 1$

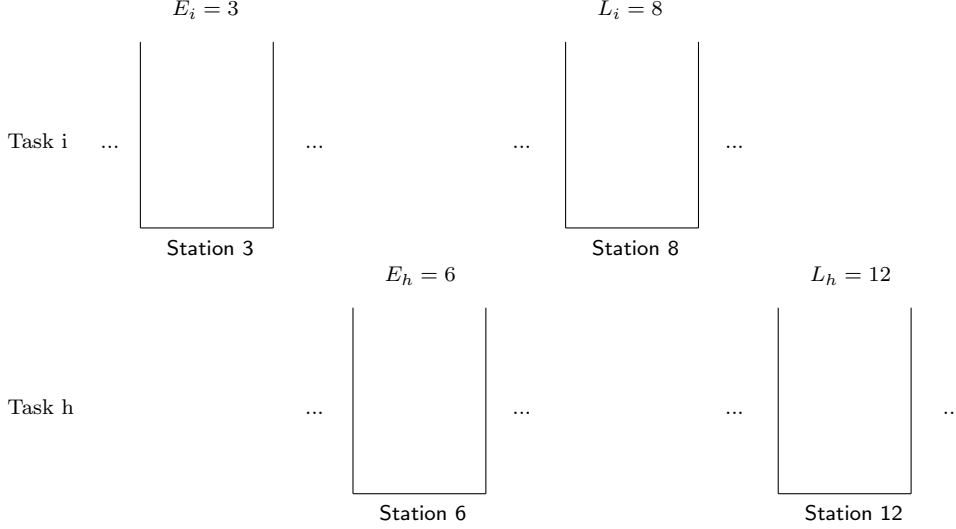


Figure 4.1: Visualization of stations on which tasks  $i$  and  $h$  can be scheduled, where  $i$  is a predecessor of  $h$  and  $E_i = 3$ ,  $L_i = 8$ ,  $E_h = 6$ , and  $L_h = 12$

and additional constraints:

$$\sum_{j=E_i}^{L_i} jx_{ij} \leq z \quad \forall i \in F \quad (\text{OBJ1-AUX})$$

where  $z$  denotes the number of used stations and  $F = \{i : S(i) = \emptyset\}$  is the set of finish tasks (without any successors). If the problem has only one finish task  $F = \{f\}$ , the above objective can be implemented without introducing variable  $z$  and additional constraints as follows:

$$\min \sum_{j=E_f}^{L_f} jx_{fj} \quad (4.8)$$

Alternatively, the objective can be based on the  $y_j$  variables, indicating whether station  $j$  is used, defined for  $j \in \{m_{\min} + 1, \dots, m_{\max}\}$ . Pastor and Ferrer (2009) propose to link  $y_j$  variables to  $x_{ij}$  variables by replacing the cycle time constraints (CYCLE) for stations  $m_{\min} + 1$  to  $m_{\max}$  with:

$$\sum_{i:j \in \{E_i, \dots, L_i\}} t_i x_{ij} \leq cy_j \quad \forall j \in \{m_{\min} + 1, \dots, m_{\max}\} \quad (\text{CYCLE-Y})$$

Then, they use the objective:

$$\min \sum_{j=m_{\min}+1}^{m_{\max}} jy_j \quad (\text{OBJ2})$$

This objective minimizes the number of used stations, but its value denotes the total cost of using stations above station  $m_{\min}$ . For symmetry breaking, the cost of using stations increases as station index increases.

Another objective based of the  $y_j$  variables which calculates the number of stations used is:

$$\min m_{\min} + \sum_{j=m_{\min}+1}^{m_{\max}} y_j \quad (\text{OBJ3})$$

However, in this case, symmetry-breaking constraints might be useful:

$$y_{j-1} \geq y_j \quad \forall j \in \{m_{\min} + 2, \dots, m_{\max}\} \quad (\text{OBJ3-AUX})$$

## 4.6 Dynamic latest station reduction

Pastor and Ferrer (2009) introduced the following constraints that strengthen the MILP formulation:

$$x_{i,L_i-q} \leq y_{m_{\max}-q} \quad \forall i \in \{1, \dots, n\}, \forall q \in \{0, \dots, m_{\max} - m_{\min} - 1\} \quad (4.9)$$

The interpretation of this constraint is as follows: if the last station is not used, then no task  $i$  can be assigned to its latest station  $L_i$  ( $q = 0$ ); if the second last station is not used, then no task  $i$  can be assigned to its second latest station  $L_i - 1$  ( $q = 1$ ); and so on.

The effect of this constraint is that it dynamically reduces the latest stations for all tasks during the solution process. When the incumbent solution in the MILP solver is improved, the latest stations for all tasks are reduced.

Ritt and Costa (2018) propose a stronger version of the above constraint:

$$\sum_{j=L_i-q}^{L_i} x_{ij} \leq y_{m_{\max}-q} \quad \forall i \in \{1, \dots, n\}, \forall q \in \{0, \dots, m_{\max} - m_{\min} - 1\} \quad (4.10)$$

## 4.7 Reduction of the number of precedence constraints

Following ideas of Dolgui and Gafarov (2017), we attempt to reduce the number of precedence relationships by introducing dummy tasks with zero duration and modifying the precedence network.

Suppose we have two disjoint subsets of tasks,  $A$  and  $B$ ,  $A \cap B = \emptyset$ , such that for each  $i \in A$  and  $h \in B$ ,  $i \in P_a(h)$ , i.e., every task in  $A$  is an immediate predecessor of every task in  $B$ . A dummy task  $d$  could be introduced such that all tasks in  $A$  are immediate predecessors of  $d$  and all tasks in  $B$  are immediate successors of  $d$ . In this case, immediate precedence relationships between all tasks

$(i, h)$ ,  $i \in A$  and  $h \in B$ , can be removed. We define a score of introducing the dummy task given sets  $A$  and  $B$  as the difference between the number of removed precedence relationships and the number of added precedence relationships:

$$\text{score}(A, B) = |\{(i, h) : i \in A, h \in B, i \in P(h)\}| - (|A| + |B|) \quad (4.11)$$

The scanning algorithm shown in Figure 4.2 can be used to detect a pair  $(A^*, B^*)$  with the largest score. If a pair  $(A^*, B^*)$  with a positive score is found, a dummy task can be added and the precedence network updated as described above. The algorithm can be called repeatedly until no pair of sets with a positive score can be found.

Figure 4.2: Scanning algorithm

```

1  $A^* \leftarrow \emptyset, B^* \leftarrow \emptyset, s^* \leftarrow 0;$ 
2  $\text{Scan}(1, \emptyset, \{1, \dots, n\});$ 
3 procedure  $\text{Scan}(i_0, A, B)$ 
4   | if  $|B| < 2$  then return;
5   |  $s \leftarrow \text{score}(A, B);$ 
6   | if  $s > s^*$  then  $(A^*, B^*, s^*) \leftarrow (A, B, s);$ 
7   | foreach  $i \in \{i_0, \dots, n\}$  do
8   |   |  $\text{Scan}(i + 1, A \cup \{i\}, B \cap S(i));$ 
9   | end
10 end

```

## 4.8 Precedence network reversion

If precedence relationships are reversed (set  $R$  is replaced by set  $R' = \{(h, i) : (i, h) \in R\}$  and sets  $P(i)$ ,  $S(i)$ ,  $P_a(i)$ , and  $S_a(i)$  are updated accordingly), the resulting problem is also SALBP-1. If it is solved and a solution with  $m^*$  stations is obtained, the solution of the original problem can be found by moving all tasks from station  $j$  to  $m^* - j + 1$ , for each  $j \in \{1, \dots, m^*\}$ .

# Chapter 5

## Our Model

We propose a new mixed-integer linear model that uses the network flow formulation to represent the load of each station. We use the network compression algorithm proposed by Brandão and Pedroso (2016) for bin packing to reduce the number of variables and constraints in the network flow formulation.

Our model has new sets of flow conservation constraints and demand constraints to ensure that each task  $i$  will be performed at a certain station  $j$ .

We replace the cycle time constraint CYCLE by the network flow formulation for bin packing problems proposed by Brandão and Pedroso (2016), and we take the below considerations:

- this a one-dimensional case, where the equivalent of the size/weight of an item  $i$  in the bin packing problem is the time  $t_i$  needed to perform the task  $i$ ,
- each station is equivalent to a bin with a constant capacity  $c$ ,
- the demand for each item, or task in our case, will be equal to 1,
- each node in the network represents a state that can be reached by scheduling a task  $i$  on the station  $j$ . One station is equivalent to a set of  $c + 1$  nodes,
- a network compression algorithm is then applied on the network to reduce the number of variables and constraints as we will describe in this section,
- all remaining constraints (precedence and occurrence) are kept in our final model.

Let's denote  $A_j$  the set of tasks  $i$  that can be scheduled on a station  $j$ .  $A_j$  is defined as the group of tasks  $i \in \{i | E_i \leq j \leq L_i\}$ . We introduce the below set of variables and constraints:



- flow variables ( $u$  denotes the source node and  $u + t_i$  the destination node):

$$f_{i,u,u+t_i,j} \quad \forall i \in A_j, \forall j \in \{1, \dots, m_{max}\}, \forall u \in \{0, \dots, c - t_i\}$$

$$0 \leq f_{i,u,u+t_i,j} \leq 1$$

- flow conservation constraints:

$$\sum_{i \in A_j: k+t_i \leq c} f_{i,k,k+t_i,j} \leq \sum_{i \in A_j: k \geq t_i} f_{i,k-t_i,k,j}, \quad \forall j \in \{1, \dots, m_{max}\}, \forall k \in \{1, \dots, c\}$$

(FLOW\_CONSERV)

The interpretation of this constraint is that the outflows should not exceed the inflows for each node except node 0.

$$\sum_{i \in A_j} f_{i,0,t_i,j} \leq 1, \quad \forall j \in \{1, \dots, m_{max}\} \quad (\text{FLOW\_CONSERV0})$$

The interpretation of this constraint is that we can only take one unit of outflow at node 0.

- occurrence constraints:

$$x_{ij} = \sum_{v=t_i}^c f_{iuv,j}, \quad \forall i \in A_j, \forall j \in \{1, \dots, m_{max}\} \quad (\text{OCCUR\_FLOW})$$

The interpretation of this constraint is to force having one flow equal to 1 on a station  $j$  where we schedule the task  $i$ .

The algorithm 5.1 shows how the network is built with the related constraints explained.

We present the example of the figure 5.2 to illustrate our model and we define the related flow variables and constraints for the first station  $j=1$ . We define the tasks precedence network in the figure 5.2 where we take  $n = 7$  for simplification.

Let's consider  $c = 10$  for this example, and  $m_{max} = 5$ . Using the equations 3.1 and 3.2 defined to calculate the earliest and latest stations for each task, we present the values for the tasks in our example in the table 5.1.

For illustration purposes, we will build the network flow model on the first station  $j=1$ . The station in our example is represented as a set of states as shown in the figure 5.3. Based on the table 5.1, we could schedule the tasks 1, 2 and 3 on the first station. The 5.4 shows the network flow model constructed based on the algorithm 5.1.

At station  $j=1$ , we write the different constraints as below:

- flow conservation at node 0:  $f_{3,0,6,1} + f_{1,0,4,1} + f_{2,0,3,1} - 1 \leq 0$

Figure 5.1: Network flow formulation algorithm

```

1 NetworkConstruction();
2 procedure NetworkConstruction()
3   foreach station  $j \in \{1, \dots, m_{max}\}$  do
4      $reachedCapacity \leftarrow 0$ ; //we start with an empty station
5     Set supply = 1; //At level 0, the inflow or supply is 1
6     Find tasks set  $A_j = \{i: E_i \leq j \leq L_i\}$ ; //tasks that can be scheduled on
       the station j
7     Sort tasks in  $A_j$  by decreasing time;
8     foreach task i in  $A_j$  do
9       foreach reachable state of station j do
10        if  $reachedCapacity + t_i \leq c$  then
11          Add flow variable  $f_{i, reachedCapacity, reachedCapacity+t_i, j}$ ; //flow
            variable at level reachedCapacity
12           $reachedCapacity = reachedCapacity + t_i$ ;
13          ;
14        end
15        Add occurrence constraint:  $x_{ij} = \sum_{v=t_i}^c f_{iuv}$ 
16      end
17      foreach reachable state in station j do
18        Set flow conservation constraint:  $\sum_{u=k} f_{iuv} - \sum_{v=k} f_{iuv}$ 
19      end
20    end
21 end

```

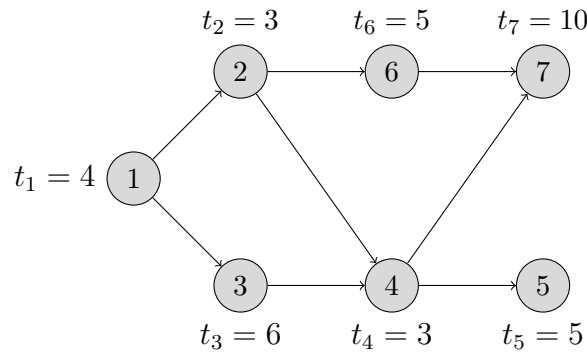


Figure 5.2: Example of ALBP problem with  $n = 7$

Task	Earliest station	Latest station
1	1	1
2	1	2
3	1	2
4	2	3
5	3	4
6	2	3
7	3	4

Table 5.1: Earliest and latest stations of tasks



Figure 5.3: Station 1 represented as a set of nodes or states

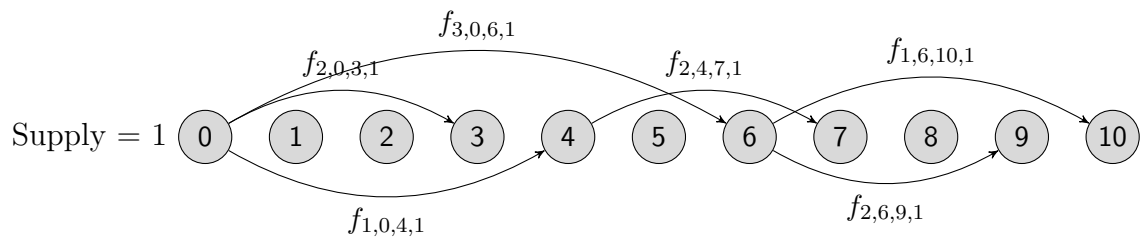


Figure 5.4: Station 1 represented as a set of nodes or states

- flow conservation at node 4:  $f_{2,4,7,1} - f_{1,0,4,1} \leq 0$
- flow conservation at node 6:  $f_{2,6,9,1} + f_{1,6,10,1} - f_{3,0,6,1} \leq 0$
- occurrence constraints:  $x_{3,1} = f_{3,0,6,1}$ ,  $x_{1,1} = f_{1,6,10,1} + f_{1,0,4,1}$  and  $x_{2,1} = f_{2,0,3,1} + f_{2,4,7,1} + f_{2,6,9,1}$

# Chapter 6

## Computational Experiments

We performed our experiments on a virtual machine with Windows 10 64-bit configured with 2 sockets, each with Intel Xeon E5-2643V2 3.50 GHz 6-core processor capable of running 12 threads. The virtual machine was configured with 10 cores available for processing and without hyper-threading. Heuristics, lower bounds, and procedures generating MILP models were programmed in Visual Studio C++ 2017. We used two solvers to solve MILP models: CPLEX 12.9 and Gurobi 8.1.

Before initiating a solver, we solved each instance with the Multi-Hoffmann heuristic proposed by Fleszar and Hindi (2003). We also calculated a lower bound as described in the above mentioned paper. If the heuristic solution was not proven to be optimal, it was solved using a MILP solver.

Tests were performed on one set of benchmark problem instances. This set, denoted BWL, described by Scholl and Klein (1997) can be downloaded from <https://assembly-line-balancing.de/salbp/benchmark-data-sets-1993/>. It includes 269 instances with  $n = 7$  to 297 tasks.

We started by performing the tests of the MILP models collected from the literature, while we included many variants of these models. The aim from this test is to identify the best performing combination in terms of convergence to the optimal solution.

The full factorial test was based on every possible combination of the below:

1. The objective function (OBJ1, OBJ2, OBJ3),
2. The precedence constraint type, as shown in table 6.1
3. Whether we introduce the dynamic latest station reduction (4.10) or not,
4. Whether we introduce the lag constraint or not,
5. Whether we set the value  $m_{max}$  to the best known value from the heuristic minus 1 or without decreasing by 1 as described in the section 4.6, denoted as mLess1 the remainder of the document,

Precedence constraint ID	Precedence constraint definition
0	PREC0
1	PREC1
2	PREC2-BETTER
3	PREC3-BETTER
4	PREC2-E
5	PREC3-E
6	PREC2-L
7	PREC3-L
8	PREC2-BOTH
9	PREC3-BOTH
10	PREC3-NEW1
11	PREC3-NEW2

Table 6.1: Precedence constraints recap

6. The type of optimizer used to solve the different instances (Cplex or Gurobi).

Based on the above, the combination of the different types of objectives and constraints will generate 576 variants of MILP models. This results in 154,944 tests ran on the 269 BWL instances to measure the performance of these 576 variants.

Following the tests, we note that the different variants of the MILP models show different performance, however none of them was able to solve all the BWL instances, as show in the table 6.2. We show in the table 6.2 the percentage of instances solved by the different variants we propose. Further statistical analysis, involving linear regression analysis of the results, shows no significant difference in terms of the models performances; we could only see few parameters that impacted the overall performance of a model, as seen in the figure 6.1.

The figure 6.1 represents the different distribution of the runs with respect to the different parameters that we changed. We see that precedence types 3 and 10 have higher impact than the other precedence types, the lag constraint is introducing enhancements, as well as the dynamic reduction of latest station. The usage of Cplex as optimizer has a higher impact than Gurobi. Finally, the objective type does not seem to have a high impact.

When checking further the models' performance, we notice that the below variants have the best performance, as they could solve 57% of the instances that were not solved by the heuristic. The models correspond to the below combination:

1. Optimizer is Cplex
2. Precedence constraint is either 3 or 10

group	cnt*	n	0	1	2	3	4	5	6	7	8	9	10	11
Arcus2	17	111	88%	88%	88%	88%	88%	88%	88%	88%	88%	88%	88%	88%
Barthol2	27	148	22%	22%	22%	22%	22%	22%	22%	22%	22%	22%	22%	23%
Lutz2	11	89	61%	62%	63%	68%	66%	69%	60%	62%	62%	61%	66%	66%
Scholl	26	297	11%	10%	9%	10%	8%	9%	9%	10%	8%	8%	10%	9%
Tonge	16	70	60%	61%	65%	69%	67%	69%	61%	61%	67%	67%	69%	65%
Warnecke	16	58	61%	57%	62%	66%	55%	56%	71%	74%	66%	66%	69%	67%
Wee-Mag	24	75	75%	75%	73%	74%	72%	72%	75%	74%	67%	68%	75%	76%

Table 6.2: Instances not solved by the MILP variants

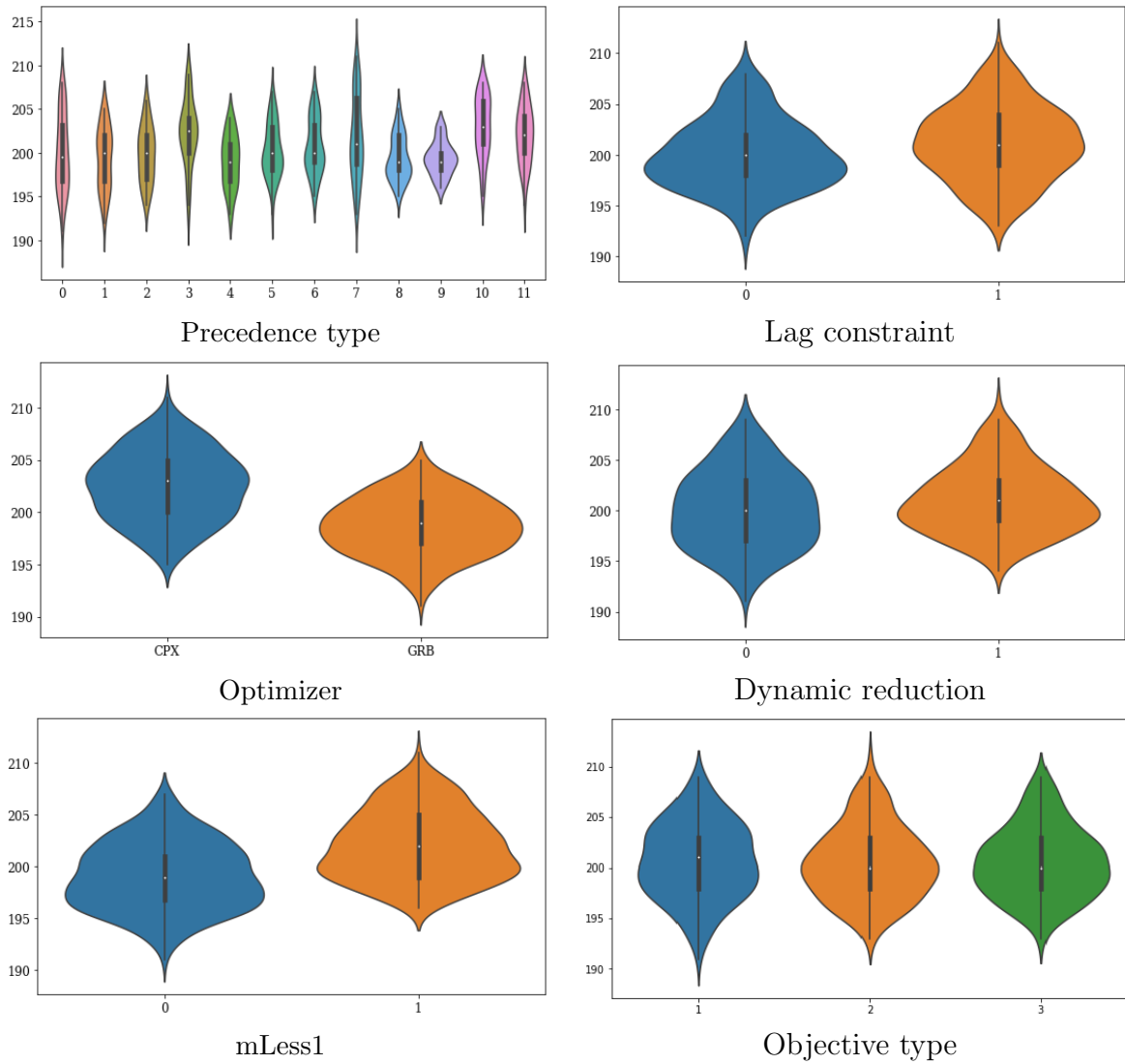


Figure 6.1: Count of optimal solutions in function of variants types

3. Lag constraint is introduced
4. Dynamic latest station is introduced (mLess1)

group	cnt	opt	lopt	gapm	%gapa	%gapm	$t_{avg}$	$t_{max}$	Constrs	Bin	Int	Contin	NZs	Nodes	LagConstr
Bowman	1	1	0	0	0	0	0.004	0.004							
Mansoor	3	3	0	0	0	0	0.005	0.005							
Mertens	6	6	0	0	0	0	0.003	0.003							
Jaeschke	5	5	0	0	0	0	0.004	0.004							
Jackson	6	6	0	0	0	0	0.005	0.005							
Mitchell	6	6	0	0	0	0	0.025	0.055	45	50	0	0	162	0	5
Heskiaoff	6	6	0	0	0	0	0.025	0.036							
Sawyer	9	9	0	0	0	0	0.107	0.308	170	161	0	0	1109	92	5
Kilbridge	10	10	0	0	0	0	0.042	0.047							
Tonge	16	12	4	1	1.56	8.33	4.617	10.224	401	450	0	0	2873	16693	13
Arcus1	16	16	0	0	0	0	0.307	0.756	364	394	0	0	2312	0	2
Arcus2	17	15	2	1	0.75	7.69	2.327	11.251	971	1043	0	0	10585	7523	10
Roszieg	6	6	0	0	0	0	0.016	0.018							
Buxey	7	7	0	0	0	0	0.067	0.124	134	126	0	0	739	0	6
Lutz1	6	6	0	0	0	0	0.042	0.088	55	64	0	0	165	0	1
Gunther	7	7	0	0	0	0	0.068	0.144	174	170	0	0	994	0	6
Hahn	5	5	0	0	0	0	0.129	0.215	62	72	0	0	152	0	0
Warnecke	16	13	3	1	0.69	3.85	4.614	10.206	700	612	0	0	7841	4457	15
Wee-Mag	24	19	5	4	1.29	11.76	4.032	10.640	2799	2335	0	0	86638	3156	34
Lutz2	11	8	3	4	1.65	8.7	4.362	10.327	1435	903	0	0	17026	1546	35
Lutz3	12	12	0	0	0	0	0.280	0.536	408	360	0	0	2855	493	6
Mukherje	13	13	0	0	0	0	0.193	0.217							
Barthold	8	8	0	0	0	0	0.424	0.474							
Barthold2	27	6	21	1	2.02	3.85	8.648	11.478	5441	4562	0	0	171034	2798	20
Scholl	26	3	23	1	2.4	3.7	14.081	16.089	5440	4797	0	0	105440	2351	6
All	269	208	61	4	0.8	11.76	3.529597	16.08892	2422	2088	0	0	57001	3785	14

Table 6.3: Results of best performing model (Precedence type = 3)

No clear impact of the objective function is observed. The tables 6.3 and 6.4 show the results for the top performing variants described above.

The same conditions of the best performing models were applied on our network flow model to check if it could solve the instances that were not solved by MILP top performing variants described above.

The different tests show a comparable result to the best MILP performing models identified above, for the same time limit = 10 seconds. Our model is able to solve 54% of the instances using the precedence type 3 or 10, as shown in the tables 6.5 and 6.6.

We suspect that the time consuming part is on the network flow construction and compression. When we increased the time limit, we could collect slightly better results as shown in the figure 6.2.

As seen in the figure 6.2, our model does not outperform the best of the existing variants we tested. However, we see that our model outperforms the best of the MILP models for Wee-Mag instance only.



group	cnt	opt	lopt	gapm	%gapa	%gapm	$t_{avg}$	$t_{max}$	Constrs	Bin	Int	Contin	NZs	Nodes	LagConstr
Bowman	1	1	0	0	0	0	0.004	0.004							
Mansoor	3	3	0	0	0	0	0.005	0.005							
Mertens	6	6	0	0	0	0	0.003	0.003							
Jaeschke	5	5	0	0	0	0	0.004	0.004							
Jackson	6	6	0	0	0	0	0.005	0.005							
Mitchell	6	6	0	0	0	0	0.026	0.057	45	50	0	0	162	0	5
Heskiaoff	6	6	0	0	0	0	0.025	0.036							
Sawyer	9	9	0	0	0	0	0.101	0.170	170	161	0	0	1101	0	5
Kilbridge	10	10	0	0	0	0	0.042	0.047							
Tonge	16	12	4	1	1.73	8.33	4.131	10.225	401	450	0	0	2848	16027	13
Arcus1	16	16	0	0	0	0	0.306	0.729	364	394	0	0	2300	0	2
Arcus2	17	15	2	1	0.75	7.69	2.350	11.268	971	1043	0	0	10485	8089	10
Roszieg	6	6	0	0	0	0	0.016	0.018							
Buxey	7	7	0	0	0	0	0.073	0.136	134	126	0	0	730	0	6
Lutz1	6	6	0	0	0	0	0.042	0.087	55	64	0	0	165	0	1
Gunther	7	7	0	0	0	0	0.080	0.172	174	170	0	0	981	0	6
Hahn	5	5	0	0	0	0	0.130	0.216	62	72	0	0	152	0	0
Warnecke	16	13	3	1	0.71	4	3.808	10.205	700	612	0	0	7745	4223	15
Wee-Mag	24	19	5	4	1.16	11.76	3.302	10.666	2799	2335	0	0	85376	3034	34
Lutz2	11	8	3	4	1.91	8.7	4.142	10.319	1435	903	0	0	16799	1096	35
Lutz3	12	12	0	0	0	0	0.318	0.517	408	360	0	0	2838	283	6
Mukherje	13	13	0	0	0	0	0.192	0.217							
Barthold	8	8	0	0	0	0	0.425	0.474							
Barthold2	27	6	21	1	2.02	3.85	8.656	11.168	5441	4562	0	0	169541	2749	20
Scholl	26	2	24	1	2.48	3.7	14.244	16.078	5440	4797	0	0	104652	2979	6
All	269	207	62	4	0.82	11.76	3.39872	16.077859	2422	2088	0	0	56480	3781	14

Table 6.4: Results of best performing model (Precedence type = 10)

group	cnt	opt	lopt	gapm	%gapa	%gapm	$t_{avg}$	$t_{max}$	Constrs	Bin	Int	Contin	NZs	Nodes	LagConstr
Bowman	1	1	0	0	0	0	0.004	0.004							
Mansoor	3	3	0	0	0	0	0.005	0.005							
Mertens	6	6	0	0	0	0	0.003	0.003							
Jaeschke	5	5	0	0	0	0	0.004	0.004							
Jackson	6	6	0	0	0	0	0.005	0.005							
Mitchell	6	6	0	0	0	0	0.037	0.106	158	50	0	249	863	0	5
Heskiaoff	6	6	0	0	0	0	0.025	0.036							
Sawyer	9	9	0	0	0	0	1.170	4.961	602	161	0	2029	7066	892	5
Kilbridge	10	10	0	0	0	0	0.042	0.047							
Tonge	16	4	12	1	4.76	9.09	9.925	13.835	4053	450	0	40646	123806	0	13
Arcus1	16	14	2	1	1.34	14.29	11.174	110.250	50162	394	0	1215695	3585889	0	2
Arcus2	17	9	8	1	2.91	7.69	121.571	300.124	143559	1043	0	3920624	11765891	0	10
Roszieg	6	6	0	0	0	0	0.016	0.018							
Buxey	7	7	0	0	0	0	0.545	2.439	504	126	0	1776	5954	517	6
Lutz1	6	6	0	0	0	0	0.342	1.448	1583	64	0	3636	9888	0	1
Gunther	7	7	0	0	0	0	0.325	1.359	709	170	0	2362	7929	43	6
Hahn	5	4	1	1	4	20	2.867	13.902	6911	72	0	42674	126352	0	0
Warnecke	16	5	11	2	3.72	7.69	9.139	11.410	2352	612	0	9757	35100	0	15
Wee-Mag	24	20	4	1	0.53	3.33	4.268	13.569	6559	2335	0	16560	134352	0	34
Lutz2	11	4	7	5	4.02	10.87	7.409	11.882	2772	903	0	5184	32007	841	35
Lutz3	12	7	5	1	2.73	9.09	8.344	12.025	2208	360	0	19360	60532	6	6
Mukherje	13	13	0	0	0	0	0.192	0.221							
Barthold	8	8	0	0	0	0	0.424	0.481							
Barthold2	27	6	21	1	2.02	3.85	21.842	28.929	14026	4562	0	221900	832709	0	20
Scholl	26	2	24	1	2.48	3.7	290.898	363.402	75723	4797	0	5232967	15753892	0	6
All	269	174	95	5	1.62	20	40.994507	363.402476	27490	2088	0	1270484	3856516	121	14

Table 6.5: Results of the network flow model (Precedence type = 3)

group	cnt	opt	lopt	gapm	%gapa	%gapm	$t_{avg}$	$t_{max}$	Constrs	Bin	Int	Contin	NZs	Nodes	LagConstr
Bowman	1	1	0	0	0	0	0.004	0.004							
Mansoor	3	3	0	0	0	0	0.005	0.005							
Mertens	6	6	0	0	0	0	0.003	0.003							
Jaeschke	5	5	0	0	0	0	0.004	0.004							
Jackson	6	6	0	0	0	0	0.005	0.005							
Mitchell	6	6	0	0	0	0	0.038	0.115	158	50	0	249	863	0	5
Heskiaoff	6	6	0	0	0	0	0.025	0.036							
Sawyer	9	9	0	0	0	0	0.891	3.958	602	161	0	2029	7058	394	5
Kilbridge	10	10	0	0	0	0	0.042	0.047							
Tonge	16	4	12	1	4.76	9.09	9.925	13.831	4053	450	0	40646	123781	0	13
Arcus1	16	14	2	1	1.34	14.29	11.188	110.350	50162	394	0	1215695	3585877	0	2
Arcus2	17	9	8	1	2.91	7.69	121.577	300.099	143559	1043	0	3920624	11765791	0	10
Roszieg	6	6	0	0	0	0	0.016	0.018							
Buxey	7	7	0	0	0	0	0.424	1.627	504	126	0	1776	5944	119	6
Lutz1	6	6	0	0	0	0	0.350	1.485	1583	64	0	3636	9887	0	1
Gunther	7	7	0	0	0	0	0.357	1.570	709	170	0	2362	7915	49	6
Hahn	5	4	1	1	4	20	2.870	13.924	6911	72	0	42674	126352	0	0
Warnecke	16	6	10	3	4	11.54	9.061	11.420	2352	612	0	9757	35003	0	15
Wee-Mag	24	21	3	1	0.39	3.23	4.210	13.435	6559	2335	0	16560	133090	0	34
Lutz2	11	4	7	5	4.02	10.87	7.423	11.907	2772	903	0	5184	31779	559	35
Lutz3	12	5	7	1	3.86	9.09	8.636	11.954	2208	360	0	19360	60515	2	6
Mukherje	13	13	0	0	0	0	0.192	0.216							
Barthold	8	8	0	0	0	0	0.422	0.471							
Barthold2	27	6	21	1	2.02	3.85	21.984	29.004	14026	4562	0	221900	831216	0	20
Scholl	26	2	24	1	2.48	3.7	291.404	364.345	75723	4797	0	5232967	15753103	0	6
All	269	174	95	5	1.67	20	41.051	364.35	27490	2088	0	1270484	3855996	61	14

Table 6.6: Results of the network flow model (Precedence type = 10)

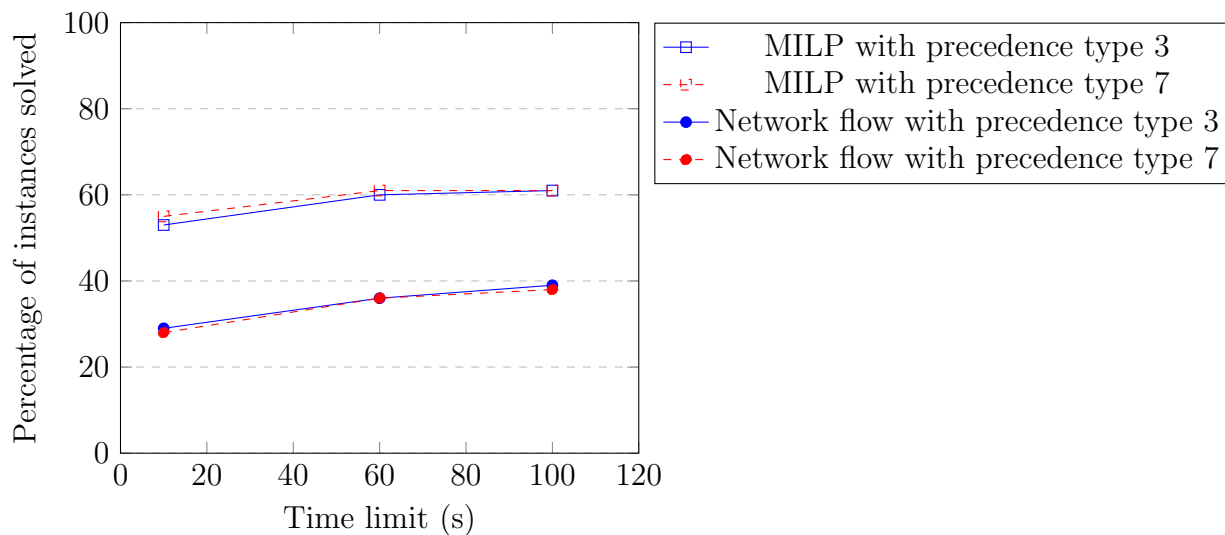


Figure 6.2: Performance comparison between models

# Chapter 7

## Conclusion

In this thesis, we did full factorial tests of the existing MILP models, and we introduced enhancements to the constraints as well as we proposed two new constraints (PREC3-NEW1 and PREC3-NEW2), and one of them turned out to have an impact on the convergence to optimality for some instances. This can be considered as an enrichment of the models that we enhanced from the literature. The network flow model we proposed to solve the SALBP-1 instances used in our tests does not outperform the best MILP variant we got. Our algorithm is taking time to build the network flow before it starts with solving the model itself. However, we saw that our model outperforms the existing MILP models for one instance (Wee-Mag). Additional configurations we could try are to implement the network compression proposed by Brandão and Pedroso (2016), as well as to test the model on a different set of data, denoted OTTO. This set was systematically generated by Otto et al. (2013) and is available online at <https://assembly-line-balancing.de/salbp/benchmark-data-sets-2013>.

# Appendix A

## Abbreviations

ALBP	Assembly Line Balancing Problem
LP	Linear Programming
MILP	Mixed-Integer Linear Programming
CP	Constraint Programming

# Bibliography

- E. H. Aghezzaf and A. Artiba. A Lagrangian relaxation technique for the general assembly line balancing problem. *Journal of Intelligent Manufacturing*, 6(2): 123–131, 1995. ISSN 09565515. doi: 10.1007/BF00123684.
- İlker Baybars. A Survey of Exact Algorithms for the Simple Assembly Line Balancing Problem. *Management Science*, 32(8):909–932, 1986. ISSN 0025-1909. doi: 10.1287/mnsc.32.8.909. URL <http://pubsonline.informs.org/doi/abs/10.1287/mnsc.32.8.909>.
- E. H. Bowman. Assembly-Line Balancing by Linear Programming. *Operations Research*, 8(3):385–389, 1960. ISSN 0030-364X. doi: 10.1287/opre.8.3.385.
- Filipe Brandão and Joo Pedro Pedroso. Bin packing and related problems: General arc-flow formulation with graph compression. *Computers and Operations Research*, 69:56–67, 2016. ISSN 03050548. doi: 10.1016/j.cor.2015.11.009.
- Alexandre Dolgui and Evgeny Gafarov. Some new ideas for assembly line balancing research. *IFAC-PapersOnLine*, 50(1):2255–2259, 7 2017. ISSN 2405-8963. doi: 10.1016/J.IFACOL.2017.08.189.
- Krzysztof Fleszar and Khalil S. Hindi. An enumerative heuristic and reduction methods for the assembly line balancing problem. *European Journal of Operational Research*, 145(3):606–620, 2003. ISSN 03772217. doi: 10.1016/S0377-2217(02)00204-7.
- Alena Otto, Christian Otto, and Armin Scholl. Systematic data generation and test design for solution algorithms on the example of SALBPGen for assembly line balancing. *European Journal of Operational Research*, 228(1):33–45, 2013. ISSN 03772217. doi: 10.1016/j.ejor.2012.12.029. URL <http://dx.doi.org/10.1016/j.ejor.2012.12.029>.
- Rafael Pastor and Laia Ferrer. An improved mathematical program to solve the simple assembly line balancing problem. *International Journal of Production Research*, 47(11):2943–2959, 6 2009. ISSN 0020-7543. doi: 10.1080/00207540701713832.

- James H. Patterson and Joseph J. Albracht. Technical Note Assembly-Line Balancing: Zero-One Programming with Fibonacci Search. *Operations Research*, 23(1):166–172, 2 1975. ISSN 0030-364X. doi: 10.1287/opre.23.1.166.
- Marcus Ritt and Alysson M. Costa. Improved integer programming models for simple assembly line balancing and related problems. *International Transactions in Operational Research*, 25(4):1345–1359, 7 2018. ISSN 09696016. doi: 10.1111/itor.12206. URL <http://doi.wiley.com/10.1111/itor.12206>.
- Armin Scholl and Robert Klein. SALOME: A bidirectional branch-and-bound procedure for assembly line balancing. *INFORMS Journal on Computing*, 9(4):319–334, 1997. ISSN 10919856. doi: 10.1287/ijoc.9.4.319.
- E. C. Sewell and S. H. Jacobson. A Branch , Bound , and Remember Algorithm for the Simple Assembly Line Balancing Problem. *INFORMS Journal on Computing*, 24(3):433–442, 2012.
- P Sivasankaran and P Shahabudeen. Literature review of assembly line balancing problems. *International Journal of Advanced Manufacturing Technology*, 73(9-12):1665–1694, 2014. ISSN 14333015. doi: 10.1007/s00170-014-5944-y. URL <https://link.springer.com/content/pdf/10.1007%2Fs00170-014-5944-y.pdf>.
- S. R. Thangavelu and C. M. Shetty. Assembly line balancing by zero-one integer programming. *AIIE Transactions*, 3(1):61–68, 1971. ISSN 05695554. doi: 10.1080/05695557108974787.
- William W White. Letter to the Editor: Comments on a Paper of Garg. *Operations Research*, 9(2):274–276, 1961. doi: 10.1023/b:scie.0000018533.22139.9c.