HEC MONTRÉAL

Optimization Models for Lot-sizing with Multiple Storage Locations

---

by

Chi Xu

Thesis director

Raf Jans

A thesis submitted in partial fulfillment

of the requirement for the degree of Master of Science

November 2019

Declaration by student
Ethical requirements:
Research on human
subjects

Office
of the Registrar

3000 chemin de la Côte-Sainte-Catherine
Montreal, Quebec, Canada H3T 2A7

HEC MONTRÉAL

Research not requiring approval from the Research Ethics Board (REB)

This form is required for thesis or supervised projects, if one of the following conditions apply:

1) a case study;
2) a research conducted with employees of a specific organisation, used exclusively for organisation assessment, management or improvement purposes.

Or, the thesis or supervised project does not include any of the following situations:

1) data collection involving human participants (interviews, observation, questionnaires, experimentation, etc.);
2) secondary use of not publicly available information involving human subjects;
3) linkage of information involving human subjects, notwithstanding whether it is publicly available or not (data linkage refers to the pairing of distinct datasets enabling linkage of specific information).

Title of research project: OPTIMIZATION MODELS FOR LOT-SIZING WITH MULTIPLE STORAGE LOCATIONS

Student name: Chi Xu

Signature:

Date: 2019-11-19

Director name: Raf Jans

Signature:

Date: 2019-11-19

Please turn in this form duly completed and signed with the initial submission of your thesis or your supervised project report.

For additional information, please contact cer@hec.ca.

ABSTRACT

Traditional lot-sizing problems have been studied for decades. In more recent studies, researchers have further proposed mixed integer programming models for various extensions, in an effort to bring step by step the mathematical problem closer to the complex reality.

In this thesis, we focus on modeling the multi-item capacitated lot-sizing problem with multiple storage locations, which we denote as CLSP-MSL. In the CLSP-MSL, the storage space is divided into several individual locations. Inventory is assigned to the storage locations according to specific rules and conditions, and can be relocated between two time periods. In addition to the cost elements in the traditional lot-sizing models, we further include three new types of costs: storage fixed cost, handling cost, and item relocation cost.

A general model is proposed for the CLSP-MSL, as well as a transportation reformulation (TP), which provides a better LP relaxation bound. Next, using simulated data sets, we carry out a series of computational tests on both the general CLSP-MSL model, and the TP formulation.

In order to analyze the impact of the various parameters, we perform a sensitivity analysis. Through this analysis, we find that results show that breaking the storage space into several storage locations helps lower the total cost. Allowing the inventory to be relocated at a cost also improves the efficiency of inventory management, although this effect appears to be very small in our tests. It is also shown that the symmetry between these storage locations can greatly add to the computational burden.

Keywords: Lot-sizing, Production planning, Mixed integer programming, Storage fixed cost, compatibility, Item relocation

ACKNOWLEDGEMENT

I would like to thank my sincere gratitude to my supervisor Raf Jans for all his valuable comments, remarks and engagement through the research and the writing of this master thesis. Without him, this work would not have been possible.

I must also express the deepest gratitude to my family. My caring wife, Xiaoye Ma, help me through many difficulties with her unconditional support. My father, Chenguang Xu, made my study financially possible while taking on the great responsibilities in the family. And especially, my dearest mother, Jiang Xue, who passed away before the completion of this work, had made countless sacrifices during her fight against the disease.

TABLE OF CONTENTS

Chapter 1. Introduction

1.1 Lot-sizing problems

The lot-sizing problem is one of the most studied problems of modern production planning. For the purpose to determine the optimal quantity for each production batch, an abundance of theories, models, and heuristics have been proposed throughout the century. At the heart of the production planning problems is the trade-off between setup costs and holding costs: large batches mean infrequent setups but high overall holding costs, whereas small batches lead to frequent setups but low overall holding costs. From the famous Economic Order Quantity model (EOQ) by Harris (1913) which assumes unlimited capacity and a constant demand on an infinite time horizon, to the numerous Mixed Integer Programming (MIP) models for the dynamic lot-sizing problem with a finite planning horizon in which the demand varies as well as the optimal lot size from period to period, more and more characteristics of industrial production environment are taken into consideration to create more realistic, solid and applicable tools.

When looking into the dynamic lot-sizing problem, one may begin from its most basic form, i.e. the single-item uncapacitated lot-sizing problem (ULSP) in which only one product is produced to fulfill the known demand for a finite time horizon. This basic model has been extended to take into account the limited capacity of various resources. Typically, according to Trigeiro, Thomas and McClain (1989) and Jans and Degraeve (2007), constraints have been imposed on the production capacity taking into consideration both the variable production time and the setup times. Since businesses, large or small, are rarely providing only one product, many extensions are made to include multiple items. The Capacitated Multi-Item Lot Sizing Problem (CLSP) is a big-bucket lot-sizing problem that addresses more than one type of item sharing an overall production capacity in each period, with each item having its own unit consumption of capacitated resource. This production capacity can take different forms in different situations and industries: processing power of the machine, limited number of workers, limited capital, fixed amount of material available for each period, etc. Although the holding cost is

included in the objective function, this basic model does not impose any restriction on the maximum inventory to be held, assuming an unlimited storage space.

In more recent studies, researchers have also been further extending the lot-sizing models with capacity constraints on storage, which brings the problem one step closer to the reality. Some other extended studies focus on a fixed charge on stock, generated by inventory carried over from the previous period, resembling the fixed cost per batch in the production phase.

1.2 Research objective

While researchers have spent great efforts to model the production planning problem for a more realistic context by considering multiple items, production and storage capacities, or fixed costs on setups and stock, little attention has been paid to cases where multiple storage locations are available. In this M.Sc. thesis, we will look into this problem with multiple storage locations, hereafter denoted by CLSP-MSL.

In CLSP-MSL, we break into the black box of inventory storage to examine in detail the activities happening in the storage area, which we will later refer to as the warehouse. We will consider material handling and storage fixed costs. The aim is to create a better tool for production planning in harmony with the warehouse operations by taking these considerations into account.

The simplest example of such a problem might be the production and storage of liquid products. Different liquids cannot be mixed, and therefore have to be stored in different tanks. Production plans should accommodate not only the demand for each item, but also the storage capacity of each tank. The typical approach of considering a global shared storage capacity no longer remains valid, since the empty space in each tank cannot be used to contain a second type of product.

In other cases, certain items cannot be stored together, so that the storage space has to be divided physically and logically into different zones. It is therefore evident that each of these zones will have their respective limited capacities. These kinds of restrictions are

commonly seen in the chemical industry. For instance, acids and bases are usually required to be separated by chemically stable materials; flammable matters must also be stored away from other hazard classes, especially oxidizers and toxics. Below are several other examples of specialized storage conditions that generate more costs and complexities to the planning activity.

a) Food: due to their different natures and perishability, food materials and products often impose various requirements on their storage conditions and environments, such as temperature, moisture and ventilation. This leads to higher fixed and variable storage cost.
b) Pharmaceuticals: while most pharmaceutical products should be stored under temperature control, they also often require a sterilized environment, which translates into even higher variable and fixed cost on stocks.
c) Chemicals: apart from controlled temperature, moisture, and exposure to light, etc., other special conditions have to be strictly obeyed to maintain the stability of many chemicals, such as vacuum or oil sealing. In addition, when multiple items share the same storage location, it is also important to consider their mutual compatibility and divide the warehouse into zones allowing a separate storage of different items.

In order to minimize the costs related to these complex storage conditions, managers need a tool to help them scientifically manage the inventory. The objective of this research is to incorporate the requirements with respect to complex storage conditions in the production planning models. More specifically, we will consider multiple capacitated storage locations. This will typically require modeling extra constraints, while incorporating additional decisions such as the assignment of items to specific storage locations. The models proposed in this thesis will also serve as such a tool to support the decision-making process.

1.3 Methodology

In this thesis, we aim to model the CLSP-MSL using the mixed integer programming (MIP) method. We assume a finite planning horizon with dynamic deterministic demand.

We will generate instances with different characteristics for the key input parameters. The proposed models will be tested using a general-purpose MIP solver (CPLEX) in computational tests. We will analyze the structure of the solutions. The CPU time and the quality of the solution in terms of optimality gap is also important especially in this case when we increase the complexity of the model because the size of the instances also contributes greatly to the computational burden.

After this introductory chapter follows the literature review which summarizes the previous works and studies in this field, providing an insight into the various types of lot-sizing problems and their formulations. In Chapter 3 and Chapter 4, we introduce the lot-sizing problem with multiple storage locations and propose formulations first for a single-item system, then for multi-item problems (i.e. CLSP-MSL). Chapter 5 explains the method and data sets of our computational tests and contains the analysis of the results. Chapter 6 concludes the thesis.

Chapter 2. Literature Review

There exists an abundance of literature on the optimization of lot-sizing problems, covering a large diversity of problem variations and model extensions for different industrial environments (Pochet and Wolsey, 2006). Our review of the literature begins with the single-item lot-sizing problems, and then extends to multi-item systems. Special attention is paid to problems with relatively complex cost structures and capacities with respect to the production and storage.

2.1 Single-item lot-sizing problem

In the single-item lot-sizing problem, only one item is produced. The goal is to design a production plan to fulfill given demands of a series of discrete time periods over a finite planning horizon. In this work, the demand that we consider is deterministic and dynamic i.e. the demand is known in advance, but may vary over time.

2.1.1 Uncapacitated lot-sizing problem

The uncapacitated lot-sizing problem (ULSP) under dynamic demand was first proposed by Wagner and Whitin (1958). To formulate ULSP, we use the following notations:

Set:

T Set of all periods within the planning horizon,

Parameters:

vc_t Variable production cost per unit of produced item in period *t*,
f_t Setup cost of each production batch in period *t*,
hc_t Inventory holding cost per unit in period *t*,
d_t Demand to be fulfilled in period *t*,
m Number of planning periods considered,

Decision variables:

x_t Production level in number of units in period *t*,

y_t Setup decision: $y_t = 1$ if any item is produced in period *t*; otherwise $y_t = 0$,

s_t Inventory level in number of units at the end of period *t*.

ULSP Model:

Minimize: $$\sum_{t \in T} (vc_t x_t + f_t y_t + hc_t s_t) \quad (1)$$

Subject to: $$s_{t-1} + x_t = d_t + s_t \quad \forall t \in T \quad (2)$$

$$x_t \leq \sum_{k=t}^{m} d_k \, y_t \quad \forall t \in T \quad (3)$$

$$s_0 = 0 \quad (4)$$

$$x_t, s_t \geq 0; \; y_t \in \{0, 1\} \quad \forall t \in T \quad (5)$$

The objective function (1) minimizes the total cost including the variable production cost, production setup cost and the variable holding costs under the following constraints. Constraints (2) balance the flow of goods by specifying that the demand in period *t* must be fulfilled with items produced in the same period or with the inventory carried over from the previous period, and any excess items will be kept in inventory.

Constraints (3) are the setup forcing constraints. Such constraints can also take the form of $x_t \leq M y_t$, where the 'big M' is a coefficient large enough for the model to choose appropriate values for the x_t variables when $y_t = 1$ without excluding any feasible solution. However, a 'big M' value that is too large will lead to a poor LP relaxation value and add to the computational burden to solve the problem. In the ULSP, we can set the 'big M' equal to the sum of remaining demands. This is valid because, given positive holding costs, there always exists an optimal solution in which the inventory at the end of the horizon is zero. Constraint (4) indicates that there is no initial inventory. Finally, constraints (5) impose the non-negativity and binary conditions.

We can see that in the ULSP, three kinds of costs are considered: the variable production cost, the setup cost of production, and the variable holding cost. On condition that the demand, if any, must be fulfilled without any backlog, the trade-off here is similar as in the classical EOQ model.

2.1.2 Production capacity

The first extension made to the ULSP is an upper bound on the production, since obviously, no business can produce an unlimited number of products within a certain period. The production capacity can be easily incorporated in the model by adding extra constraints specifying the upper bounds. Using P_t to denote this production capacity in period *t*, the capacity constraints can be written as follows:

$$x_t \leq min\left(\sum_{k=t}^{m} d_{ik}\,, P_t\right) \quad \forall t \in T \tag{6}$$

The upper bounds for production quantity depend not only on the capacity P_t, but also on the total demand of the remaining periods (including period *t*), since, in certain periods, the latter may provide tighter bounds.

2.1.3 Storage capacity

Just as the production is limited in the real world by time, manpower, and means of production, the inventory cannot accumulate infinitely. One of the most obvious limitations is the space used to store this inventory. Once a warehouse is acquired or rented and the physical feature of the product is given, the maximum number of units that can be stored is fixed.

Author(s)	Problem Characteristics			
	Demand	**Production**	**Inventory**	**Other features**
Love (1973)	DD	Capacitated	Upper & Lower bounds	
Erenguc and Aksoy (1990)	DD	Capacitated	Upper bounds	
Jaruphongsa, Çetinkaya and Lee (2004)	DD	Capacitated	Upper bounds	Multi-echelon Delivery time windows
Wolsey (2006)	DD	Capacitated/ Uncapacitated	Upper bounds	Delivery time windows
Guan and Liu (2010)	SD	Capacitated/ Uncapacitated	Upper bounds	
Hwang and van den Heuvel (2012)	DD	Capacitated	Upper bounds	Storage shared between production and inventory
Chu *et al.* (2013)	DD	Capacitated	Upper & Lower bounds	With backlogging and outsourcing

DD=deterministic dynamic SD=stochastic dynamic

Table 2.1-1 Summary of literature on single-item lot-sizing problems with bounded inventory

Table 2.1-1 summarizes the previous works on the single-item lot-sizing problems with bounded inventory. In most of the lot-sizing problems previously studied, the inventory level is limited by the storage capacity as upper bounds, while Love (1973) and Chu *et al.* (2013) applied also lower bounds. In this thesis, we only consider upper bounds on inventory while not allowing any backlogging. Using H to denote the storage capacity in each period, the inventory upper bounds can be imposed by adding the following constraints to the model:

$$s_t \leq min\left(\sum_{k=t+1}^{m} d_{ik}\,, H\right) \quad \forall t \in T \tag{7}$$

In the constraints (7), the inventory is limited not only by the storage capacity H, but also by the sum of the remaining demand (excluding period t). For some time periods, the former might provide tighter bounds than the latter.

In a single-item problem, since only one item is involved, the physical feature of the warehouse can be easily translated into the maximum number of units to be stored. However, in a multi-item problem, the dimensions might vary according to the different types of item. In that case, the value of H signifies the overall available space in the storage facility and the constraints should be modified accordingly as well.

2.1.4 Storage fixed cost

Fixed charges on inventory is another important extension to the previous model. Similar to the production setup cost, the fixed cost on stock does not vary according to the amount of inventory. It is only generated by the decision to keep the storage space open to hold inventory. If the inventory in a period is strictly positive, then the storage fixed cost has to be paid. In business operations, this fixed cost on stock may take the form of warehouse rental, salaries, maintenance fee, or other overhead costs. Table 2.1-2 summarizes the related literature on single-item lot-sizing problem with fixed cost on stock.

Author(s)	Problem Characteristics			
	Demand	**Production**	**Storage fixed cost**	**Other features**
Van Vyve and Ortega (2004)	DD	Capacitated	Linear	
Atamtürk and Küçükyavuz (2005)	DD	Capacitated	Linear	Possible initial stock
Atamtürk and Küçükyavuz (2008)	DD	Capacitated	Linear	
Di Summa and Wolsey (2010)	DD	Capacitated	Only on initial inventory	Bounded initial stock
Wolsey (2015)	DD	Capacitated	Linear	

Table 2.1-2 Summary of literature on single-item lot-sizing problems with fixed storage cost

Based on the ULSW, a capacitated version of the model can be easily formulated with the following elements added:

Additional Parameters:

g_t fixed cost of having a positive inventory level at the end of period *t*,

Additional Decision variables:

z_t Binary variable, $z_t = 1$ if the inventory is positive at the end of period *t*; $z_t = 0$ if otherwise,

Single-item CLSP with upper bounds and fixed cost on stock:

Minimize:
$$\sum_{t=1}^{m} (vc_t x_t + f_t y_t + hc_t s_t + g_t z_t) \quad (8)$$

Subject to:
$$s_{t-1} + x_t = d_t + s_t \quad \forall t \in T, \quad (9)$$

$$x_t \leq min\left(\sum_{k=t}^{m} d_k, P\right) y_t \quad \forall t \in T \quad (10)$$

$$s_t \leq min\left(\sum_{k=t+1}^{m} d_k\,, H\right) z_t \quad \forall t \in T \tag{11}$$

$$s_0 = 0, \tag{12}$$

$$x_t, s_t \geq 0;\ y_t, z_t \in \{0, 1\} \quad \forall t \in T. \tag{13}$$

In the objective function (8), all three types of costs previously considered are present as well. In addition, $g_t z_t$ represents the fixed cost on stock in period *t*: in case of positive inventory at the end of period *t*, z_t takes the value of 1, therefore generating a cost of g_t; otherwise, $z_t = 0$, meaning that no inventory is held at the end of the period.

Since the discussion here involves only one type of item, stored in one location, the production setup-forcing constraints can be integrated in the production capacity constrains (10) through the multiplication of the upper bounds with the setup binary variables y_t. Thus, the production quantity is not only limited by the capacity and the remaining demands, but also by the setup decision: if no setup occurs, even if the capacity is enough, the right-hand side of the inequality equals 0, therefore disabling the production. The same logic applies for the storage capacity and storage fixed cost constraints (11). Constraints (9) and (12) remain the same, while the new binary variable z_t is added to the domain constraints (13).

2.2 Multiple items

To include more than one item in our production, there are generally 2 kinds of models, each addressing a different type of production environment (Pochet and Wolsey, 2006): the large bucket models in which multiple items can be produced using the same resource within a given time period; and the small bucket model, where only one item can be processed in a given time period. Moreover, since we assume that all items are produced on the same machine, it is then inevitable to include the production capacity.

We assume the same demand characteristics, zero initial inventory and zero ending inventory in our discussion for the multi-item lot-sizing problem as for the single-item problems.

2.2.1 Production capacity for multi-item problems

In this thesis, we will focus on the multi-item CLSP, which can be categorized as a large bucket model. This means that within one time period, multiple types of items can be produced. To distinguish each type of item, the related parameters and decision variables are indexed with *i*, while *I* is the set all types. Additional parameters and the problem formulation are as follows:

Additional Set:

I Set of all types of items to be produced,

Additional Parameters:

vt_i Consumption of capacity to produce 1 unit of item *i*,

P Production capacity,

Multi-item CLSP Model:

Minimize:
$$\sum_{i\in I}\sum_{t\in T}(vc_{it}x_{it} + f_{it}y_{it} + hc_{it}s_{it}) \quad (14)$$

Subject to:
$$s_{i,t-1} + x_{it} = d_{it} + s_{it} \quad \forall i \in I, \forall t \in T, \quad (15)$$

$$x_{it} \le min\left(\sum_{k=t}^{m} d_{ik}, \frac{P}{vt_i}\right) y_{it} \quad \forall i \in I, \forall t \in T, \quad (16)$$

$$\sum_{i\in I} vt_i x_{it} \le P \quad \forall t \in T, \quad (17)$$

$$s_{i0} = 0 \quad \forall i \in I, \quad (18)$$

$$x_{it}, s_{it} \ge 0;\ y_{it} \in \{0,1\} \quad \forall i \in I, \forall t \in T. \quad (19)$$

The objective function (14) and the demand constraints (15) are adapted to include multiple items. Constraints (18) and (19) are similar to their respective counterparts in the single-item problem.

The setup forcing constraints (16) is modified as well. In the multi-item problem, the production capacity can no longer be measured directly by the number of units, since not all items have the same requirement with respect to the production time. In this case, P_t represents the capacity of the shared resource measured in units of time available. Therefore, in the capacity constraints (17), the left-hand side is a sum of total resource usage over all items.

2.2.2 Storage capacity and fixed cost for multi-item problems

It has been stated in Van Vyve and Ortega (2004) that "*Fixed charges on the stocks may be necessary to model holding costs. Another situation where fixed charges on the stocks arise naturally are models involving several items linked by combinatorial constraints on stocks.*". However, the authors focused solely on a single-item model without further exploration down this direction.

Later on, the storage capacity has been researched as well for multiple items by numerous researchers. The following tables summarizes the related literature on multi-item lot-sizing problems with storage bounds.

Author(s)	Problem Characteristics				
	Demand	**Production**	**Storage capacity**	**Storage fixed cost**	**Other features**
Minner (2009)	DD	Capacitated	Shared	None	
Coppens (2013)	DD	Capacitated	Shared	None	
Akbalik, Penz and Rapine (2014)	DD	Uncapacitated	Shared	None	
Akbalik, Penz and Rapine (2015)	DD	Capacitated	Shared	None	
Melo and Ribeiro (2016)	DD	Capacitated	Shared	None	
Cunha and Santos (2017)	DD	Capacitated	Multiple tanks	None	Alternative recipes By-products
Cunha *et al.* (2018)	DD	Capacitated	Multiple tanks	None	Multi-purpose tanks

Table 2.2-1 Summary of literature on multi-item lot-sizing problems

Similar to the big bucket model for the production capacity, most models proposed in the previous studies all assume that different types of item are stored in one shared space. In Cunha and Santos (2017) and Cunha et al. (2018), the inventory is stored in several capacitated tanks, but neither study considers any sort of fixed costs generated by the storage equipment.

However, although researchers have paid much attention to the inventory bounds, a discussion on the storage fixed cost within a multi-item context appears to be absent. To incorporate the storage capacity as well as fixed cost in the multi-item CLSP model, we need to introduce new parameters to denote the per-unit consumption of space for each item.

Additional Parameters:

st_i Consumption of storage capacity to store 1 unit of item *i*,

Multi-item CLSP Model with inventory bounds and fixed cost:

Minimize: $$\sum_{i\in I}\sum_{t\in T}(vc_{it}x_{it} + f_{it}y_{it} + hc_{it}s_{it}) + \sum_{t\in T} g_t z_t \quad (20)$$

Subject to: $$s_{i,t-1} + x_{it} = d_{it} + s_{it} \quad \forall i \in I, \forall t \in T, \quad (21)$$

$$x_{it} \le min\left(\sum_{k=t}^{m} d_{ik}, \frac{P}{vt_i}\right) y_{it} \quad \forall i \in I, \forall t \in T, \quad (22)$$

$$\sum_{i\in I} vt_i x_{it} \le P \quad \forall t \in T, \quad (23)$$

$$s_{it} \le min\left(\sum_{k=t+1}^{m} d_{ik}, \frac{H}{st_i}\right) z_t \quad \forall i \in I, \forall t \in T, \quad (24)$$

$$\sum_{i\in I} st_i s_{it} \le H \quad \forall t \in T, \quad (25)$$

$$s_{i0} = 0 \quad \forall i \in I, \quad (26)$$

$$x_{it}, s_{it} \ge 0;\ y_{it}, z_t \in \{0, 1\} \quad \forall i \in I, \forall t \in T. \quad (27)$$

Compared to the single-item CLSP, a storage fixed cost is added to the objective function (20). Constraints (21) – (23), (26), and (27) remain the same as in the previous models. Constraints (24) function in the same way of the production setup forcing constraints (22). In the storage capacity constraints (25), the left-hand side of the inequality is the total space occupied by the inventory.

2.3 Cost structure

Early studies on lot-sizing problems, such as Wagner and Whitin (1958), consider only the most basic trade-offs in extremely simplified scenarios. In this thesis, we continue to include these costs in our models:

- Variable production cost: this cost represents the consumption of resources, such as raw material, energy, and human labor etc., during the production activities. It usually depends on the nature of the product as well as the production process, and is assumed to be proportional to the number of units produced. When these costs are time-invariant, and when all demand must be exactly satisfied, these costs can be left out of the objective function since they represent a fixed total, independent of the lot-sizing decisions.
- Fixed production cost (setup cost): This usually includes the costs of preparing, changing, testing, or cleaning the equipment before a batch is being produced. This cost is fixed regardless of the number of units to be produced, but may vary according to the type of item.
- Variable holding cost: We adopt the conventional definition for this cost, as the cost per period for a unit held in stock. This cost typically include a cost related to the cost of capital (opportunity cost), as well as a physical holding cost. In different contexts, this cost may be composed of different parts such as depreciation, insurance, and obsolescence etc. Since one of our focus is to discuss the storage location assignments, the space rental and personnel costs are treated differently in comparison with many other simplified structures. Existing literature on this topic is reviewed in the following chapter.
- Variable holding cost: In classical lot-sizing models, inventory generates only consider a unit holding cost per unit per time period, typically as a percentage of the value of the product. This cost represents the opportunity cost generated according to the value of products held as inventory, as well as the physical holding cost such as rent, depreciation and obsolescence.

In reality, the cost structure is more complex, e.g. fixed costs related to the storage location, handling cost, order preparation cost, etc. Researchers have investigated the various activities throughout the whole production and storage process, trying to draw a better picture of the cost structure. An accurate mathematical description of the operations allows us to modify the model to obtain results that better reflect the reality. Kaplan and Bruns (1987) first brought the idea of activity-based costing (ABC) into the manufacturing industry. Horngren *et al.* (2010) defined ABC as "…*(to) calculate the costs of individual activities and assign costs to cost objects such as products and services on the basis of the activities undertaken to produce each product or service*". Lin *et al.* (2001) examine the adoption of ABC in supply chain management, and recommended a series of implementation procedures and techniques while also pointing out several drawbacks. Berling (2008) looked specifically into the inventory holding cost which is assumed to increase linearly with a rate equal to a percentage of the product value. By applying the concept of ABC, the authors proposed a model incorporating individual cost-generating activities in the inventory holding operation. Azzi *et al.* (2014) carried out a case study and summarize the different sources of storage costs into two different groups as shown in Table 2.3-1.

Cost structure A	Cost structure B
Evident costs	*Investment costs*
1. Floor space	1. Machines
2. Energy	2. Racks
3. Cleaning	3. Control system
4. Surveillance	4. Fire protection
5. Insurances	5. Aisle equipment
6. Taxes	6. WHMS and HAS equipments (HW and SW)
7. Material handling/storage equipments	7. Installation and testing
8. WHMS and HAS equipments	8. Training
9. Maintenance	9. Conveyor systems
10. Direct labor	10. Lands and building
Semi-evident costs	*Operation costs*
1. Obsolescence	1. Indirect labor
2. Product damage	2. Supervision
3. Product depreciation	3. Energy
4. Product deterioration/expiration	4. Maintenance
5. Indirect labor and supervision	5. Product damage/depreciation and deterioration
6. Stock list execution	6. Lost sales or backlog
Hidden costs	
1. Inspection and counting during the year	
2. Remanufacturing	
3. Repackaging and relabelling	
4. Lost sales or backlog	

Table 2.3-1 Storage cost structures in two groups. From Azzi et al. (2014), Table I

The authors specify that the cost structure A corresponds to the traditional approach of holding cost as a percentage of the stored product value, while the cost structure B consists of a storage cost parameter *"expressed as a function of the total number of pallet positions available in the warehouse (or other specific stock keeping unit (SKUs)), without including the opportunity cost value"*.

In this thesis, we are inspired by the ABC perspective and develop our major models by including the following additional components in the cost structure:

- Fixed storage cost: Similar to a production batch, a storage location may need preparation as well. This part includes the fixed part of the rental fee, operational charges and salary of employees to prepare and maintain storage locations in a specific period. For example, there are fixed costs involved to keep a storage location at a specific temperature or to keep it sterile.
- Variable handling cost: The variable handling cost in this work specifically reflects the effort to move a unit from the production line to the assigned storage location. This cost only has to be paid when the units are moved into (and out of) the storage location, and is hence different from the variable holding cost, which has to be paid for each period in which the item is in storage.
- Variable relocation cost: In a later part of this study, we will consider a situation where stored items are allowed to be moved from one storage location to another at a certain cost. This cost is related to the size and weight of the item as well as the distance between the origin and destination. It is considered to consist of the manpower and energy required by the movements.

By adopting such a cost structure which include the traditional and new cost elements listed above, we are able to balance multiple trade-offs among the six types of costs at the same time under a holistic perspective and thus build better tools for decision makers.

2.4 Link to current research

As revealed by our review on related literature, there is an absence of focus on the multi-item lot-sizing problems with inventory bounds and storage fixed cost. We will make extensions to include these considerations in a multi-item context. Moreover, we will further look into the warehouse which will be divided into several storage locations to which the inventory is assigned considering cost and storage requirements.

In our research, although we still adopt the traditional concept of the variable holding cost, we will look into the details related to the inventory holding activities. More specifically, in addition to the costs considered in the classical lot-sizing problems, we will go further by including the unit handling cost, storage fixed cost, and unit relocation cost in CLSP-MSL.

In light of all these previous works, we will develop several mixed integer programming (MIP) models for CLSP-MSL with different characteristics, therefore requiring the formulation to change accordingly.

Chapter 3. Single-item CLSP-MSL

The literature review indicates that the ideas of inventory bounds and fixed costs on stock have already been proposed. We extend the existing work by considering multiple storage locations. This also means that we must explicitly include decisions on where to stock specific items, i.e. the storage location assignment.

In this chapter, we start by building new models for the CLSP within a single-item production environment with multiple storage locations.

In the literature on single-item lot-sizing problem, fixed charges on stock and stock capacity limits were only considered for one storage location. As such, many common business practices in the real world are omitted, such as multiple warehouses, self-storage units, and even server rental or cloud data storage. To address this issue, we will consider multiple storage locations, each with its own limited capacity and fixed cost.

As a concrete example, imagine a producer of a chemical solvent with several tanks at its disposal. Instead of constantly producing exactly the demand in each period, the manager decides to take advantage of the well determined data of future demands and the available storage capacity consisting of multiple tanks, hoping to cut some costs through a better planned production.

3.1 Homogeneous Locations

We first assume that the storage locations are homogeneous, i.e., they are identical in terms of capacity and fixed cost. In order to model this, we expand the value range of z_t to non-negative integers. Thus, by allowing z_t to take integer values greater than 1, we enable the option to open multiple storage locations, each at the same fixed cost g_t. However, in this case, H stands for the capacity of an individual storage location. The total storage capacity can therefore be expressed as Hz_t. By modifying the related constraints, the single-item CLSP with homogeneous storage locations can be formulated as follows:

Additional parameter:

n Total number of available locations

Problem formulation:

Minimize: $$\sum_{t=1}^{m}(vc_t x_t + f_t y_t + s_t hc + g_t z_t) \quad (28)$$

Subject to: $$s_{t-1} + x_t = d_t + s_t \quad \forall t \in T, \quad (29)$$

$$x_t \leq min\left(\sum_{k=t}^{m} d_{ik}, P\right) y_t \quad \forall t \in T \quad (30)$$

$$s_t \leq min\left(\sum_{k=t+1}^{m} d_{ik}, H\right) z_t \quad \forall t \in T \quad (31)$$

$$z_t \leq n \quad (32)$$

$$s_0 = 0 \quad (33)$$

$$x_t, s_t \geq 0;\ y_t, z_t \in \mathbb{N}^0 \quad \forall t \in T. \quad (34)$$

This model is largely similar to model (8)-(13). Without modifying the objective function and constraints (31), we allow z_t to take any non-negative integer value to incorporate multiple homogeneous locations. As the objective is to minimize the total cost, $z_t$will be limited to the minimal necessary integer, meaning that the fewest possible locations are opened to hold all the inventory. Constraints (32) limit the total number of available locations. Constraints (33) imposes zero initial inventory. The domain constraints (34) are adjusted accordingly as well.

When we increase the inventory, the new inventory will be first added to the partially used storage location (if any). Similarly, inventory retrieved will first be taken also from the partially used locations (if any).

3.2 Heterogeneous Locations

Just as in production planning, where there might be several machines with different characteristics to complete the same task, this heterogeneity can also be relevant for the

storage locations. Even if we still produce only one item, the decision of putting the inventory in one location or another can have a different impact on the total costs.

Again, consider the example used in previous discussions. This chemical company expanded and has acquired some newer and larger tanks. Yet, the older tanks are not to be discarded, and can still be used. This brings more complexity into our considerations. Suppose that, due to the limited production capacity, the inventory accumulates through several periods and, before shipping to the customer, exceeds the maximum volume of a single old tank. In some cases, it might have been more reasonable to use the larger tank from the start. The decision of which tanks to use hence has to be incorporated in the mathematical formulation.

To characterize the different storage locations, our model needs to be expanded with a new index *l* signifying a specific location, while *L* is the set of all available locations. And the related elements of the model are accordingly modified as follows:

Minimize:
$$\sum_{t=1}^{m}(vc_t x_t + f_t y_t) + \sum_{l\in L}\sum_{t\in T}(s_{lt} hc_l + g_{lt} z_{lt}) \quad (35)$$

Subject to:
$$\sum_{l\in L} s_{l,t-1} + x_t = d_t + \sum_{l\in L} s_{lt} \quad \forall t \in T, \quad (36)$$

$$x_t \le min\left(\sum_{k=t}^{m} d_{ik}, P\right) y_t \quad \forall t \in T \quad (37)$$

$$s_{lt} \le min\left(\sum_{k=t+1}^{m} d_{ik}, \frac{H_l}{st}\right) z_{lt} \quad \forall l \in L, \forall t \in T \quad (38)$$

$$s_0 = 0, \quad (39)$$

$$x_t, s_t \ge 0;\ y_t, z_{lt} \in \{0,1\} \quad \forall t \in T. \quad (40)$$

Since the storage capacity, variable and fixed inventory cost can vary from location to location, the related parameters are marked by an index *l*. In the objective function (35), the variable and fixed costs for inventory are now summed not only over time periods, but also over locations. The demand balance constraints (36) are modified accordingly as well to take into account the multiple locations.

In this case of heterogenous storage locations for a single item, the model assumes that inventory can be moved without any cost from one location to another between two time periods. This means that we only need to consider the inventory balance at the global level as done in (36). Considering the fixed cost of storage location, it may be beneficial to change location when inventory is increased or reduced. Although, in this chapter, we assume that no extra cost is generated by such a relocation, such a cost will be taken into consideration in a later part of this thesis as a further extension.

Chapter 4. Multi-item CLSP-MSL

In modern industries, a manufacturer rarely produces only one type of product. To better plan the production of multiple items, it is important to see their different characteristics such as requirement of materials and occupation of storage space. Having discussed several extensions of the single-item version in the previous parts, we now move on to the multi-item capacitated lot-sizing problem with multiple capacitated storage locations and fixed storage cost.

4.1 Homogeneous locations for multiple items

We first assume that all storage locations are homogeneous. Different types of items are indexed by *i*, while *I* is the set of all types. Each type of item is characterized by its variable production cost (vc_i), fixed production cost (f_i), and holding cost (hc_i).

In addition, the different natures of these items also determine the way in which they share the production and storage capacities. For a certain item *i*, we use vt_i to represent the per-unit utilization of production capacity P, and st_i to represent the per-unit utilization of storage capacity *H*. Since each location has the same individual capacity *H*, the warehouse as a whole is thus naturally capacitated by $z_t H$. The model can be formulated as follows:

Minimize:
$$\sum_{t\in T}\sum_{i\in I}(vc_i x_{it} + f_i y_{it} + hc_i s_{it}) + \sum_{t\in T} g z_t \tag{41}$$

Subject to:
$$s_{i,t-1} + x_{it} = d_{it} + s_{it} \quad \forall i \in I, \forall t \in T \tag{42}$$

$$x_{it} \le min\left(\sum_{k=t}^{m} d_{ik}, \frac{P}{vt_i}\right) y_{it} \quad \forall i \in I, \forall t \in T \tag{43}$$

$$\sum_{i\in I} vt_i x_{it} \le P \quad \forall t \in T \tag{44}$$

$$\sum_{i \in I} st_i s_{it} \leq min\left(\sum_{i \in I} \sum_{k=t+1}^{m} st_i d_{ik}, H\right) z_t \quad \forall t \in T \tag{45}$$

$$s_{i1} = 0 \quad \forall i \in I \tag{46}$$

$$x_t, s_t \geq 0; \;\; y_{it} \in \{0,1\}; \;\; z_t \in \mathbb{N}^0 \quad \forall t \in T \tag{47}$$

Note that in the objective function (41), for each time period, the fixed cost on stock now stands apart from the sum of other types of costs, because this fixed cost is item-independent and occurs only once per period. With related variables and parameters adapted to include multiple items, the flow balance constraints (42) functions in the same way as (29). Since the physical dimension of the items may vary, in constraints (45), we use the total required space to impose the storage capacity and setup constraints. Different from the single-item model, the production capacity in constraint (44) is expressed, not as upper bounds on the number of units produced, but as the total amount of used time or the resource shared among different items, since each item has a different production requirement vt_i. As a result, to enforce the production setup, we now need a separate constraint (44), where, for a specific item and time period, we use the lesser between the maximum number of units allowed by the current capacity and the total demand in the remaining periods. Although this provides us with a better LP relaxation bound, the formulation typically still results in a large positive LP gap, which drives researchers to find alternative formulations with better bounds on this value. In later parts of this work, we also try to improve the formulation by reformulating the problem using the Transportation Formulation and test this reformulation to examine its performance. Finally, constraints (47) define the domain for the decision variables.

4.2 Heterogeneous locations for multiple items

In this scenario, the warehouse disposes of a finite set of storage locations indexed by *l*. The individual storage capacity and fixed cost for each location is now location-dependent, thus respectively denoted by H_l and g_l. As a result of the differences among the storage locations, it is now necessary to go further into details and monitor the inventory in each location. Several changes are made to the model accordingly:

1. The decision variable z_{lt} is again binary and location-dependent, indicating whether or not a storage location is used in period *t*.
2. The storage capacity is now managed at the location level. When the inventory in each location respect its own individual capacity, the overall storage capacity is naturally bounded by the finite set of locations *L*.
3. A new decision is integrated into the model to assign different types of item to storage locations. This decision is denoted by the binary variable w_{ilt}, which equals 1 if item *i* is stored in location *l* in period *t*, otherwise 0.
4. Considering the heterogeneity of storage locations, two binary input parameters are needed to describe the related compatibilities. α_{il} indicates the compatibility between each item and each location, taking the value of 1 if item *i* is allowed to be stored in location *l*; otherwise, 0. β_{ij} indicates the compatibility between two items *i* and *j*. If they can coexist in the same location, $\beta_{ij} = 1$; otherwise, $\beta_{ij} = 0$.

This problem can be formulated as follows:

Minimize: $$\sum_{i\in I}\sum_{t\in T}(vc_{it}x_{it} + f_{it}y_{it}) + \sum_{i\in I}\sum_{l\in L}\sum_{t\in T} hc_{il}s_{ilt} + \sum_{l\in L}\sum_{t\in T} g_l z_{lt} \quad (48)$$

Subject to: $$\sum_{l\in L} s_{il,t-1} + x_{it} = d_{it} + \sum_{l\in L} s_{ilt} \quad \forall i \in I, \forall t \in T \quad (49)$$

$$x_{it} \le min\left(\sum_{k=t}^{m} d_{ik}, \frac{P_t}{vt_i}\right) y_{it} \quad \forall i \in I, \forall t \in T \quad (50)$$

$$s_{il0} = 0 \quad \forall i \in I, \forall l \in L \quad (51)$$

$$\sum_{i\in I} vt_i x_{it} \le P_t \quad \forall t \in T \quad (52)$$

$$st_i s_{ilt} \le min\left(\sum_{k=t+1}^{m} st_i d_{ik}, H_l\right) w_{ilt} \quad \forall i \in I, \forall l \in L, \forall t \in T \quad (53)$$

$$\sum_{i \in I} st_i s_{ilt} \leq H_l z_{lt} \quad \forall l \in L, \forall t \in T \tag{54}$$

$$w_{ilt} \leq \alpha_{il} \quad \forall i \in I, \forall l \in L, \forall t \in T \tag{55}$$

$$w_{ilt} + w_{jlt} \leq \beta_{ij} + 1 \quad \forall i, j \in I, \forall i \leq j, \forall l \in L, \forall t \in T \tag{56}$$

$$x_{it}, s_{ilt} \geq 0; \;\; w_{ilt}, y_{it}, z_{lt} \in \{0, 1\} \quad \forall t \in T \tag{57}$$

In constraints (49) the flow of an item is balanced using the consolidated inventory across all locations, without further tracking the specific numbers of units going in and out of each location. This implies that, from period to period, inventory of an item can be moved between locations for the sake of other newly stocked items, or with the purpose of economizing space and fixed cost. In this model, these movements between different storage locations, called inventory relocation, are allowed at no extra cost, and hence they are not modeled. In the next section we will present a more general formulation that models inventory relocation and tracks inventory at the storage location level.

In constraints (53), we model the inventory allocation: the inventory variable is linked to the binary decision variable w_{ilt}, meaning that the inventory of item *i* in location *l* can only exist if we assign item *i* to location *l*. As for the storage capacity, constraints (54) impose that on one hand, products can be stored in location *l* only when this location is opened ($z_{lt} = 1$), while on the other hand, the total space occupied by all types of items in it cannot exceed its capacity H_l.

At last, constraint (55) and (56) make sure that all items are stored with respect to their location compatibility and product compatibility rules. Constraints (57) impose the domain for the decision variables.

4.3 General model with inventory relocation

As a result of the item-location and item-item compatibilities introduced in our previous discussion, items have to be carefully assigned to different locations – not only to respect the compatibility, but also to reach optimality by avoiding unnecessary cost.

In the previous formulation, we assumed that the inventory can be relocated within the warehouse in every time period at no additional cost. Such a relocation might be beneficial in order to reduce the total fixed inventory cost or maintain feasibility. However, if items are moved to different locations within the warehouse from one period to the next, which we will refer to as inventory relocation, this will typically entail an additional processing cost. We will analyze this issue in more detail taking into account the extra cost of inventory relocation. This will also require to track the inventory at the individual storage location level instead of at an aggregate level. This results in the most general model.

4.3.1 Logic and process

Prior to mathematically formulating this problem, in order to clarify the premises and assumptions, let us first examine the various movements and the interaction among the production, demand, and inventory.

We are given a series of demand over a finite time horizon, based on which we have to design an economic production and storage plan. For a certain time period, we have to make decisions considering not only the future demand, but also other factors such as the traditional production and holding costs, the fixed costs on production and storage, the capacities, and the newly introduced storage compatibility requirements, etc.

In each period, on one hand, the produced items (if any) can be either shipped to the customers or stored in our warehouse. On the other hand, the demand of that period can be fulfilled with products coming either directly from the production line or from the inventory accumulated in previous periods. The resulting interaction between the production and inventory in each period includes three scenarios:

Under-production: in the current period, the amount produced is less than the demand. Products have to flow out of the warehouse to compensate this lack, lowering the inventory level.

Exact-production: in the current period, the amount produced equals the demand, and is directly shipped to the customer without going into or out of the warehouse. There is no change in inventory level.

Over-production: in the current period, the amount produced is more than the current demand. These extra items flow into the warehouse, raising the inventory level.

We use the following terminology to describe the related movements of goods:

Inflow: this is the flow of products entering the warehouse to be stored as inventory as a result of over-production.

Outflow: this is the flow of products leaving the warehouse to satisfy the demand as a result of over-production.

Direct flow: this is the flow of finished products going from the production line directly to the customers, without entering the warehouse.

Relocation: this is the movement within the warehouse, carrying products from one storage location to another.

These different flows should be considered in the optimization model with their related costs as explained next. With respect to the handling costs of inflow and outflow, since we assume zero inventory at the end of the planning horizon, all items that go into the warehouse will leave eventually. Without loss of generality, we can aggregate the handling costs of both directions and apply them only on the inflow. Within the warehouse, the relocation cost mainly represents the effort to move the products from the original location to its new destination. This cost depends on the travel distance and the product feature such as dimension and weight. The direct flow of goods, i.e. goods that do not go in storage, do not have any warehouse handling cost associated with them. Assuming a transportation cost per unit would result in a fixed cost since there is no ending inventory and all units produced (whether directly shipped or first stored) need to be transported. Therefore, we do not include this transportation cost in our model.

Under these assumptions, the optimal movement of items can be depicted more precisely with the following characteristics:

- **On the global inventory level:** In case of an over-production, there should be only inflow but no outflow, since the demand of that period should be fully satisfied directly from the production, i.e. using direct flow. Similarly, in case of an under-production, there should be only an outflow but no inflow. In other words, an inflow and an outflow can by no means coexist within the same period.
- **On the local inventory level:** In a specific period, the relocation of any type of item does not involve any intermediate location. In other words, no item should pass through a third location other than its origin and destination.

4.3.2 Problem formulation

To incorporate the relocation in our model, we use v_{iklt} to denote the number of item *i* moved from location *k* to location *l* in period *t*, each at a unit moving cost r_{ikl} depending on the type of item and the distance between the two locations. Thus, in period *t*, the total number of item relocated to location *l* equals $\sum_{k\in L} v_{iklt}$, while the total number of item *i* relocated from location *l* to other locations equals to $\sum_{k\in L} v_{ilkt}$.

With respect to the inflow (i.e. the movement from the production line into the warehouse) and the outflow (i.e., the movement from the warehouse to the customer), we define δ^+_{ilt} as the inflow of item *i* into location *l* and δ^-_{ilt} as the outflow of item *i* extracted from location *l*. As claimed in the previous part, a unit handling cost, denoted by ha_{il}, is linked to δ^+_{ilt}. Both types of variables are non-negative, and can be mathematically defined as follows:

$$\sum_{l\in L} \delta^+_{ilt} - \sum_{l\in L} \delta^-_{ilt} = x_{it} - d_{it} \quad \forall i \in I, \forall t \in T \tag{58}$$

In case of an over-production ($x_{it} - d_{it} > 0$), we have $\sum_{l\in L} \delta^+_{ilt} > \sum_{l\in L} \delta^-_{ilt}$. Considering the related cost, the minimization of objective will yield a positive inflow and zero out

flow. The same logic applies to the case of under-production, resulting in a positive outflow and zero inflow. At last, both the inflow and outflow are zero in case of an exact-production.

Accordingly, the flow balance for each location can be expressed by the following equation:

$$s_{il,t-1} + \delta^{+}_{ilt} + \sum_{k \in L} v_{iklt} = s_{ilt} + \delta^{-}_{ilt} + \sum_{k \in L} v_{ilkt} \quad \forall i \in I, \forall t \in T, \forall l \in L \tag{59}$$

Although, for the two types of movement, both directions appear in the same constraint, the minimization process will eliminate the possibility of coexisting opposite movements of the same type. More specifically, $\sum_{k \in L} v_{iklt}$ and $\sum_{k \in L} v_{ilkt}$ will not be positive at the same time. If both are positive, a certain amount of item *I* is moved in and out of location *l* within the same period. However, a relocation route including any intermediate location will by nature induce more cost, which then translates into the Characteristic 3 stated in the previous part. Similarly, if both δ^{+}_{ilt} and δ^{-}_{ilt} are positive, some amount of item *i* enters and leaves the warehouse within the same period. Since there is a cost associated to the inflow, a solution in which both δ^{+}_{ilt} and δ^{-}_{ilt} are positive will not be optimal, as both amounts can be reduced simultaneously. The general formulation of the problem is thus as follows:

Minimize:
$$\sum_{i \in I} \sum_{t \in T} (vc_i x_{it} + f_i y_{it}) + \sum_{i \in I} \sum_{l \in L} \sum_{t \in T} hc_i s_{ilt} + \sum_{i \in I} \sum_{l \in L} \sum_{t \in T} ha_{il} \delta^{+}_{ilt} + \sum_{l \in L} \sum_{t \in T} g_l z_{lt} + \sum_{i \in I} \sum_{k \in L} \sum_{l \in L} \sum_{t \in T} r_{ikl} v_{iklt} \tag{60}$$

Subject to:
$$x_{it} - d_{it} = \sum_{l \in L} (\delta^{+}_{ilt} - \delta^{-}_{ilt}) \quad \forall i \in I, \forall t \in T \tag{61}$$

$$s_{il,t-1} + \delta^{+}_{ilt} + \sum_{k \in L} v_{iklt} = s_{ilt} + \delta^{-}_{ilt} + \sum_{k \in L} v_{ilkt} \tag{62}$$
$$\forall i \in I, \forall t \in T, \forall l \in L$$

$$s_{il0} = 0 \quad \forall i \in I, \forall l \in L \tag{63}$$

$$x_{it} \leq min\left(\sum_{k=t}^{m} d_{ik}, \frac{P_t}{vt_i}\right) y_{it} \quad \forall i \in I, \forall t \in T \tag{64}$$

$$\sum_{i \in I} vt_i x_{it} \leq P_t \quad \forall t \in T \tag{65}$$

$$s_{ilt} \leq min\left(\sum_{k=t+1}^{m} d_{ik}, \frac{H_l}{st_i}\right) w_{ilt} \quad \forall i \in I, \forall l \in L, \forall t \in T \tag{66}$$

$$\sum_{i \in I} st_i s_{ilt} \leq H_l z_{lt} \quad \forall l \in L, \forall t \in T \tag{67}$$

$$w_{ilt} \leq \alpha_{il} \quad \forall i \in I, \forall l \in L, \forall t \in T \tag{68}$$

$$w_{ilt} + w_{jlt} \leq \beta_{ij} + 1 \quad \forall i \in I, \forall j \in I | j \geq i, \forall l \in L, \forall t \in T \tag{69}$$

$$x_{it}, s_{ilt}, \delta^+_{ilt}, \delta^-_{ilt} \geq 0; \quad u_{it}, w_{ilt}, y_{it}, z_{lt} \in \{0,1\} \quad \forall i \in I, \forall l \in L, \forall t \in T \tag{70}$$

Constraints (61) define the inflow and outflow at the global level while constraints (62) balance the flow of products at each storage location. At the beginning of period *t*, location *l* has an inventory level of $s_{il,t-1}$. It may receive additional products from the production (δ^+_{ilt}) or from other locations ($\sum_{k \in L} v_{iklt}$). Its inventory may be extracted to fulfill the demand (δ^-_{ilt}) or be relocated to other locations ($\sum_{k \in L} v_{ilkt}$). The remainder is kept as the inventory at the end of the period (s_{ilt}). Constraints (63) imposes that there is no initial inventory at the beginning of the planning horizon. The remaining constraints (64) to (70) resemble the corresponding constraints in the previous model.

We regard this as a general model since the models given in previous parts can be seen as special cases derived from this one. When set *I* contains only one element, the problem becomes a single-item problem. When H_l and g_l are uniform, all locations become homogeneous. When all α_{il} and β_{ij} variables are set to 1, the compatibility no longer

limits the optimization. We can also force $v_{iklt} = 0$ to unable the relocation between storage locations while still keeping track of the flows at each location level.

4.3.3 Example of inventory relocation

In this part, we design and solve a simple problem to better illustrate the mechanism of item relocation. Table 4.3-2 shows the demand and cost parameters of the problem.

Demand	Item A	Item B	Item C	Item D
Period 1	1	1	1	0
Period 2	0	1	1	1
Period 3	4	3	0	2
Variable Cost (vc)	1	1	1	1
Fixed Cost (f)	10	10	10	10
Holding Cost (hc)	1	1	1	1
Stock Fixed (g)	1	1	1	1

Table 4.3-2 Relocation example - demand and cost parameters

α	Item A	Item B	Item C	Item D
Location 1	1	0	1	1
Location 2	1	1	1	1
Location 3	1	1	1	1
β	**Item A**	**Item B**	**Item C**	**Item D**
Item A	1	1	0	1
Item B	1	1	1	0
Item C	0	1	1	1
Item D	1	0	1	1

Table 4.3-1 Relocation example – compatibility matrix

As shown in Table 4.3-1, the demand of 4 items are given for 3 periods. All items have the same cost structure. Each item consumes the same amount of space in the storage location and uses the same amount of capacity for production. The total production capacity is 12 units while the storage capacity is 3 units for each of the three locations. According to the compatibility relationships shown in *Table 4.3-2*, item A and item C cannot coexist in the same location, just as item B and item D, while item B cannot be stored in location 1.

This simplified example can be easily solved to optimality, allowing us to analyze the result through observation. This problem is modeled according to the model with relocation, then solved in IBM ILOG CPLEX Optimization Studio (Version: 12.8.0.0). The optimal solution is illustrated in Figure 4-1.

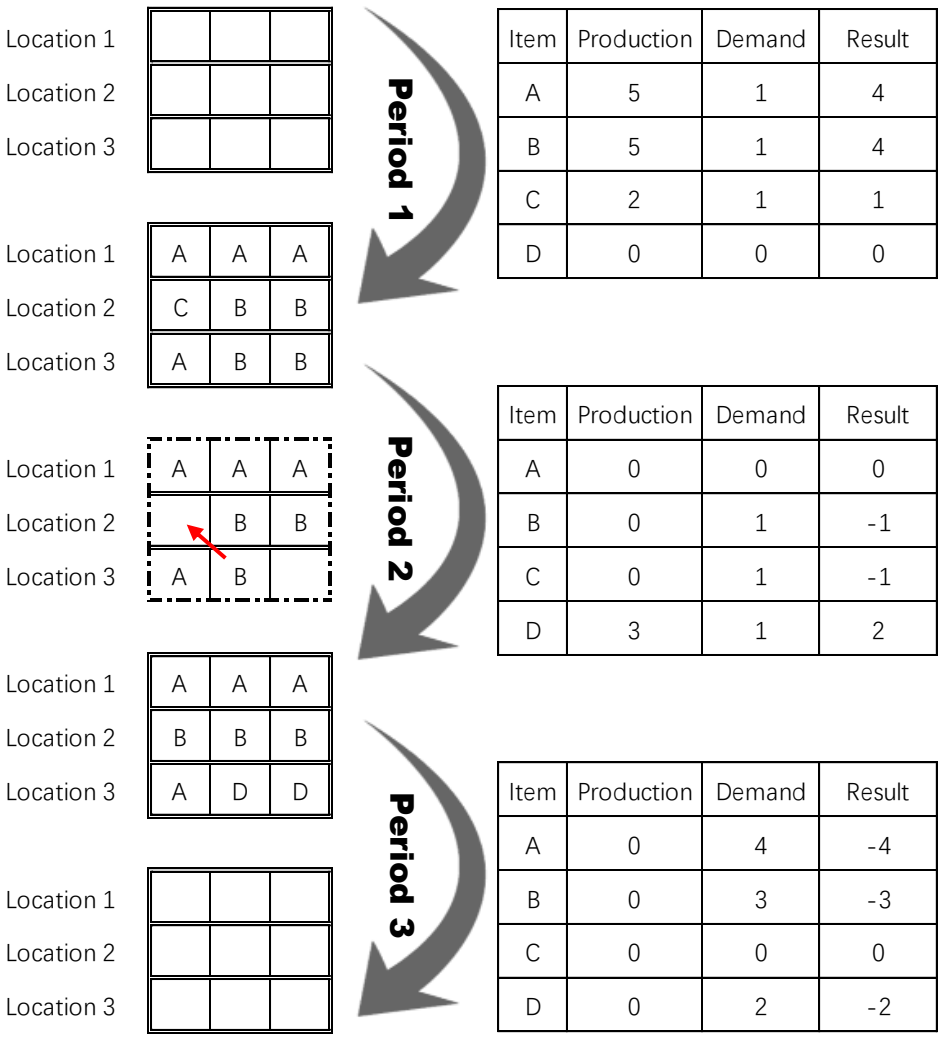

Figure 4-1 Relocation example - optimal solution

Due to the relatively high production fixed cost, it is desirable to produce as few batches as possible, in this specific case, only 1 batch for each item. At the end of period 1, we have to carefully assign the inventory to each location to accommodate the compatibilities and future storage. Since item A is incompatible with item C, and since at least two locations are needed to store 4 unites of item A, this one unit of item C can only be put into the other location without A, in this case, location 2. The rest of the storage space is filled by 4 units of item B.

In the following period, 1 unit of item C and 1 unit of item B are taken out from storage and shipped to the customers, while 2 units of the new item D enter the warehouse. According to the current situation, our next question then becomes which unit of item B to ship out. Again, considering the compatibilities, item B cannot coexist with item D in the same location. A relocation between location 2 and location 3 must be carried out to make room for the incoming item D. To minimize the cost, it is therefore ideal to move 1 unit of item B in the second period as indicated by the red arrow in Figure 4-1.

Finally, all inventory is used to meet the demand in period 3.

4.4 Transportation problem formulation of the general model

According to Krarup and Bilde (1977), reformulating the CLSP as a transportation problem provides a tighter LP relaxation. Such a formulation can possibly also find better upper bounds if an instance cannot be solved within the time limit. Define p_{itu} as the number of units produced in period *t* to satisfy the demand of item *i* in period *u*, where $t \leq u$, while other notations remain the same. Using these new variables, we can reformulate the CLSP-MSL problem. The reformulation is as follows:

Minimize:

$$\sum_{i\in I}\sum_{t\in T}\sum_{u=t}^{m} p_{itu}\, vc_i + \sum_{i\in I}\sum_{l\in L}\sum_{t\in T} hc_i s_{ilt} + \sum_{i\in I}\sum_{l\in L}\sum_{t\in T} \delta^{+}_{ilt} ha_{il} + \sum_{i\in I}\sum_{t\in T} f_i y_{it} + \sum_{l\in L}\sum_{t\in T} g_l z_{lt} + \sum_{i\in I}\sum_{k\in L}\sum_{l\in L}\sum_{t\in T} r_{ikl} v_{iklt} \tag{71}$$

Subject to:

$$s_{il,t-1} + \delta^{+}_{ilt} + \sum_{k\in L} v_{iklt} = s_{ilt} + \delta^{-}_{ilt} + \sum_{k\in L} v_{ilkt} \tag{72}$$

$$\forall i \in I, \forall t \in T, \forall l \in L$$

$$\sum_{t=1}^{u} p_{itu} = d_{iu} \quad \forall i \in I, \forall u \in T \tag{73}$$

$$\sum_{u=t}^{m} p_{itu} - d_{it} = \sum_{l \in L} (\delta_{ilt}^{+} - \delta_{ilt}^{-}) \quad \forall i \in I, \forall t \in T \tag{74}$$

$$\delta_{il1}^{-} = 0 \quad \forall i \in I, \forall l \in L \tag{75}$$

$$p_{itu} \leq d_{iu} y_{it} \quad \forall i \in I, \forall t, u \in T, t \leq u \tag{76}$$

$$\sum_{i \in I} \sum_{u=t}^{m} p_{itu} v t_i \leq P_t \quad \forall t \in T \tag{77}$$

$$s_{ilt} \leq min\left(\sum_{u=t+1}^{m} d_{iu}, \frac{H_l}{st_i} \right) w_{ilt} \quad \forall i \in I, \forall l \in L, \forall t \in T \tag{78}$$

$$\sum_{i \in I} st_i s_{ilt} \leq H_l z_{lt} \quad \forall l \in L, \forall t \in T \tag{79}$$

$$w_{ilt} \leq \alpha_{il} \quad \forall i \in I, \forall l \in L, \forall t \in T \tag{80}$$

$$w_{ilt} + w_{jlt} \leq \beta_{ij} + 1 \quad \forall i \in I, \forall j \in I | j \geq i, \forall l \in L, \forall t \in T \tag{81}$$

$$p_{itu}, s_{ilt} \geq 0; \quad w_{ilt}, y_{it}, z_{lt} \in \{0, 1\} \quad \forall t \in T \tag{82}$$

Different from the x_{it} variables signifying the production amount of product *i* in period *t*, the total production of item *i* in period *t* is given by $\sum_{u=t}^{m} p_{itu}$, with *m* signifying the total number of time periods. Related expressions are modified accordingly in the objective function (71) and in the constraints.

Constraints (73) impose the demand fulfillment by specifying that the accumulated production $\sum_{t=1}^{u} p_{itu}$ targeting period *u* equals to the demand in period *u*. Constraints (74) provide the new definition of inflows and outflows: in period *t*, if the total production $\sum_{u=t}^{m} p_{itu}$ is larger than the demand of the same period, there is a positive inflow; otherwise, a positive outflow. Constraints (76) are the new production setup forcing constraints.

Chapter 5. Computational experiments

The general CLSP-MSL model incorporating handling costs and item relocation discussed in Chapter 4.4 has been implemented using IBM ILOG Studio (ver.12.8.0) and ILOG OPL Script. The computation is carried out using a 64-bit operating system with an Intel® Core™ i7-6700HQ CPU of 2.60GHz. During the computation, we allow CPLEX the access to all 8 threads of the CPU and a total RAM of 10GB. In case that this working memory runs out, we also allow the software to move a group of nodes to temporary files on the hard disk. There is no limit on the maximal number of nodes to be processed before the computation is forced to stop. We instead limit the computation time to 30 minutes per instance since we deem that beyond this point, the optimizing process would be impractical for production planning purposes. While the solver incorporates its own symmetry breaking function, we set this option to default to let the software decide.

In all our experiments, we try to solve the problem to optimality. The computation will not stop until CPLEX finds an optimal solution or reaches the time limit. CPLEX uses a default optimality tolerance of 1e-5. We keep the optimality tolerance at this default level. In the computational experiments, we will indicate the number of instances which end within this tolerance, but with a non-zero gap.

5.1 Data description

Since our literature review shows an absence of existing data sets with the various locational parameters required to perform the tests, our test data are generated according to preliminary tests. A base case is thus established with 20 instances, of which the output data serve as the benchmark in our comparisons with other alternative parameter settings.

Number of items: According to the result of similar experiments through the literature, having additional types of item is supposed to considerably complicate the problem. Hence, to obtain valid results for multi-item problems while avoiding unnecessary extreme complexities, we include 5 items in all the instances to be tested.

Planning horizon: A longer planning horizon significantly adds to the computational burden. In our tests, while the base case consists of 15 periods, we also test the problem with 12 and 20 periods to observe the impact of different planning horizons.

Demand: For each item, the demand in each period is randomly generated from a uniform distribution between 40 and 60, with an average of 50 units.

Per-unit utilization of capacity/space: In order to facilitate the computational tests, we assume that all items require the same capacity to produce, while they occupy the same amount of storage space. Thus, the capacities of production and storage directly translate into the number of units.

Production capacity: Since the average demand for each item is 50 units per period, a total of 250 units should be produced on average during each period. To ensure the basic flexibility and to avoid infeasible problems, especially when no inventory exists at the beginning periods, we set the production capacity to 500 units in the base case. Comparisons on this parameter include another two scenarios: 375 units and 750 units.

Production setup cost: The fixed cost of production for each item is randomly generated following a uniform distribution between 80 and 120, with an average of 100. The influence of this cost is examined with an alternative case where the setup costs are halved.

Variable production cost: Since we do not allow backlogs or lost sales, the variable production costs always add up to a constant amount according to the total demand. Without any loss of generality, this type of cost is therefore omitted in our tests.

Storage Locations: Different from the production capacity, the total storage capacity is decided by the types and numbers of location used in each period. Across all instances, we designed three types of locations depending on the average demand. A small location holding inventory up to one period (50 units); two periods for a medium location (100 units); and four periods for a large location (200 units). In the base case, 4 medium locations are available.

Storage fixed cost: The storage fixed costs varies according to the type of location. In order to have a reasonable balance between the production setup cost and the inventory system costs, we set the storage fixed cost based on the following reasoning.

In our base case, given a production capacity of 500 units and an average demand of 250 units per period, the production set-up needs to occur at least once every two periods. The resulting average total setup cost is 250 for all five items.

In order to establish links between the storage fixed cost and the production setup cost, we set this average total setup cost as a base cost. Since we have 4 medium locations, the base cost per location is therefore 62.5. Having carried out several preliminary tests, we found 20% of this base cost per location (12.5) to be reasonable for a medium location.

To observe the impact of combining different types of storage locations, we test two alternative combinations, both with the same total storage capacity as the base case. The fixed costs of the different types of location are proportional with their individual capacity. Table 5.1-1 shows the two alternative combinations.

	Small	Medium	Large	Total
Fixed Cost	*6.25*	*12.5*	*25*	Capacity
Base case	0	4	0	400
Combination 1	2	1	1	400
Combination 2	0	0	2	400

Table 5.1-1 Test settings - location combinations

Cost structure for storage fixed cost: To observe the results of fixed costs exhibiting economies of scale (EOS) or diseconomies of scale (DOS), we established another two alternative scenarios, both using the Combination 1, i.e. 2 small, 1 medium and 1 large location. In case of EOS, a location twice as large generates 1.5 times the cost instead of 2 times; while in case of DOS, it is 2.5 times. Table 5.1-2 summarizes the fixed costs of different types of locations:

	Small	Medium	Large
Combination 1	6.25	12.5	25
Economy of scale	8.33	12.5	18.75
Diseconomy of scale	5	12.5	31.25

Table 5.1-2 Test settings - fixed costs per location

Variable holding cost: In order to balance the trade-off between production and holding, we adopt the concept of time between orders (TBO) derived from the EOQ model to set the holding cost. Let d be the average demand, f if the fixed production setup cost, and hc the holding cost per unit per period, deriving from $EOQ = \sqrt{\frac{2df}{hc}}$ and $TBO = \frac{EOQ}{d}$, we have $hc = \frac{2f}{TBO^2 d}$. We test three scenarios where the TBO takes the value of 2, 4 or 8. The holding cost is therefore respectively calculated to be 1, 0.25 or 0.0625.

Relocation cost: We assume that the item relocation allows a better organization of inventory but generates certain costs depending on the relocated weight, volume, and distance. However, to facilitate the tests, we adopt a uniformed relocation cost of 0.0625 per unit per movement, regardless of the distance between each pair of the four locations.

Handling cost: In order incorporate the trade-off between the two types of movements – relocation (within the warehouse) and handling (into the warehouse), the unit handling cost is set to be 10 times the unit relocation cost. We also have also carried out tests on this ratio (2, 6, 14 times) to examine the impact of different ratios between handling and relocation cost.

Item-item/item-location compatibility: As another major feature added to our models, the compatibilities are introduced using binary variables, with 0 indicating incompatible, and 1 indicating compatible. To facilitate the analysis, we measure the level of compatibility of each instance with the percentage of non-zero variables over the total number of variables within the compatibility matrix.

β	Item 1	Item 2	Item 3	Item 4	Item 5
Item 1	-	1	1	1	0
Item 2	1	-	1	0	1
Item 3	1	1	-	1	0
Item 4	1	0	1	-	1
Item 5	0	1	0	1	-

Table 5.1-3 An example of the compatibility matrix

For example, the matrix of item-item variable shown in Table 5.1-3 indicates the compatibility between each pair of two items. Among all 20 variables listed, 14 takes the value of 1. The level of item-item compatibility for this instance therefore equals $\frac{14}{20} \times 100\% = 70\%$. The level of item-location compatibility is calculated in the same way.

In our computational tests, we assume five different levels of compatibility: 100% (perfect compatibility), 90%, 80%, 70%, and 60%. Table 5.1-4 illustrates the 9 alternative cases being tested:

	Base Case	90% Compatibility			80% Compatibility			70% Compatibility			60% Compatibility		
		Overall	Item Only	Location Only	Overall	Item Only	Location Only	Overall	Item Only	Location Only	Overall	Item Only	Location Only
Item-item	100%	90%	90%	100%	80%	80%	100%	70%	70%	100%	60%	60%	100%
Item-product	100%	90%	100%	90%	80%	100%	80%	70%	100%	70%	60%	100%	60%

Table 5.1-4 Test settings - levels of compatibility

In the following parts, we first present and analyze the results of the base case. Then, after introducing the complete structure of the computational tests, we continue to show the results and sensitivity analysis of each test groups.

5.2 Base case analysis

Table 5.2-1 illustrates the results of all 20 instances in the base case. The notions used in our tables are explained as follows:

Status:	Final status of the computation. (1=Solved to optimality; 11=Exceeding time limit)
Gap:	Relative optimality gap. Equals to 0% if solved to optimality.
RlaxedOBJ:	The final objective value of the linear relaxation of the original problem
Holding:	Total variable holding cost
Setup:	Total production setup cost
StockFixed:	Total storage fixed cost
Handling:	Total handling cost
Relocation:	Total relocation cost
Handled:	Total number of handled units (i.e. the total number of units entering the warehouse)
Relocated:	Total number of relocated units
NSetups:	Total number of production setups
NLocations:	Total number of activated locations
Total Space:	Total storage space provided by all activated locations
Used Space:	Total storage space occupied by the inventory
Utilization:	The percentage of space occupied by inventory over the total opened space
Objective:	The final value of the objective function at the end of the calculation
Time:	The time consumed during the calculation (in seconds)
Best Bound:	The best overall lower bound at the end of the computation
Nodes:	Total number of nodes explored at the end of the computation

	Average Results						Relaxation			Separated Costs				
Instance	Objective	Time	Best Bound	Nodes	Status	Gap	RelaxedOBJ	Time	Status	Holding	Setup	StockFixed	Handling	Relocation
1	5606.13	203.31	5606.13	569923	1	0%	1734.73	0.008	1	13.91%	53.10%	8.03%	24.96%	0.00%
2	5500.25	217.24	5500.25	636782	1	0%	1691.46	0.008	1	12.95%	55.60%	7.50%	23.95%	0.00%
3	5405.00	326.28	5405.00	814547	1	0%	1604.79	0.007	1	13.35%	53.67%	7.40%	25.58%	0.00%
4	5516.88	576.32	5516.88	1631471	1	0%	1665.73	0.008	1	14.83%	51.17%	8.38%	25.61%	0.00%
5	5268.25	25.61	5268.25	86745	1	0%	1556.69	0.008	1	12.70%	55.39%	6.88%	25.03%	0.00%
6	5407.50	143.61	5407.50	453767	1	0%	1656.93	0.008	1	12.03%	57.48%	6.93%	23.56%	0.00%
7	5635.38	358.04	5635.38	1249872	1	0%	1773.70	0.007	1	14.99%	51.57%	8.21%	25.23%	0.00%
8	5345.25	258.90	5345.25	1165304	1	0%	1590.72	0.007	1	13.28%	54.61%	7.25%	24.86%	0.00%
9	5475.25	111.40	5475.25	221663	1	0%	1704.00	0.007	1	13.35%	54.85%	7.76%	24.04%	0.00%
10	5466.25	336.93	5466.25	828329	1	0%	1688.76	0.006	1	12.88%	55.65%	7.32%	24.15%	0.00%
11	5483.62	554.69	5483.62	1440029	1	0%	1629.69	0.010	1	13.59%	53.67%	7.98%	24.77%	0.00%
12	5471.13	230.94	5471.13	531637	1	0%	1629.63	0.006	1	12.48%	55.84%	7.31%	24.37%	0.00%
13	5547.50	76.08	5547.50	210417	1	0%	1701.10	0.007	1	14.42%	51.64%	8.11%	25.82%	0.00%
14	5693.63	1800.04	5680.00	1811985	11	0.24%	1755.30	0.010	1	12.08%	57.94%	6.81%	23.17%	0.00%
15	5824.50	261.20	5824.50	565017	1	0%	1835.91	0.006	1	14.80%	51.78%	8.58%	24.83%	0.00%
16	5422.00	523.22	5422.00	1722733	1	0%	1632.76	0.006	1	13.38%	54.37%	7.61%	24.64%	0.00%
17	5645.75	354.17	5645.75	1226418	1	0%	1760.82	0.006	1	13.32%	54.93%	7.53%	24.22%	0.00%
18	5519.38	60.65	5519.38	157217	1	0%	1691.64	0.007	1	12.19%	57.45%	7.02%	23.34%	0.00%
19	5504.25	705.93	5504.25	2366383	1	0%	1700.91	0.006	1	14.41%	51.80%	8.40%	25.39%	0.00%
20	5429.88	256.91	5429.88	798960	1	0%	1690.71	0.006	1	13.98%	53.02%	8.06%	24.94%	0.00%
Average	5508.39	369.07	5507.71	924459.95	-	-	1684.80	0.007	-	13.45%	54.27%	7.66%	24.62%	0.00%

Table 5.2-1 Results of the base case

	Other Parameters							
Instance	Handled	Relocated	NSetups	NLocations	Total Space	Used Space	Utilization	Total Demand
1	2239.00	0.000	29	36	3600.00	3119.00	86.64%	3777.00
2	2108.00	0.000	31	33	3300.00	2849.00	86.33%	3760.00
3	2212.00	0.000	31	32	3200.00	2886.00	90.19%	3835.00
4	2261.00	0.000	29	37	3700.00	3273.00	88.46%	3848.00
5	2110.00	0.000	32	29	2900.00	2676.00	92.28%	3777.00
6	2038.00	0.000	32	30	3000.00	2603.00	86.77%	3772.00
7	2275.00	0.000	28	37	3700.00	3380.00	91.35%	3733.00
8	2126.00	0.000	32	31	3100.00	2840.00	91.61%	3845.00
9	2106.00	0.000	30	34	3400.00	2924.00	86.00%	3696.00
10	2112.00	0.000	31	32	3200.00	2817.00	88.03%	3732.00
11	2173.00	0.000	31	35	3500.00	2980.00	85.14%	3846.00
12	2133.00	0.000	32	32	3200.00	2732.00	85.38%	3824.00
13	2292.00	0.000	29	36	3600.00	3200.00	88.89%	3859.00
14	2111.00	0.000	32	31	3100.00	2751.00	88.74%	3817.00
15	2314.00	0.000	28	40	4000.00	3449.00	86.23%	3865.00
16	2138.00	0.000	31	33	3300.00	2901.00	87.91%	3798.00
17	2188.00	0.000	30	34	3400.00	3009.00	88.50%	3784.00
18	2061.00	0.000	32	31	3100.00	2691.00	86.81%	3770.00
19	2236.00	0.000	29	37	3700.00	3173.00	85.76%	3782.00
20	2167.00	0.000	29	35	3500.00	3036.00	86.74%	3667.00
Average	2170.00	0.000	30.400	33.750	3375.00	2964.45	87.84%	3789.35

Table 5.2-1 Results of the base case (continued)

The first group of data includes the basic information of the computational tests. With a maximum of 5824.500 and a minimum of 5268.250, the objective values are relatively stable across all instances. However, not all instances are solved to optimality. Instance 14 is marked by a status code 11, which indicates that the computation for this instance exceeded the time limit of 30 minutes. Its computation time of 1,800 seconds delivers the same message. Only in this instance, the best bound is different from the objective value. More generally, when the computation of an instance ceases before reaching optimality, the best bound is the best overall lower bound at the end of the computation. Since our objective function minimizes the total cost, a best bound value is always less than (not optimal) or equal to (optimal) the final objective value.

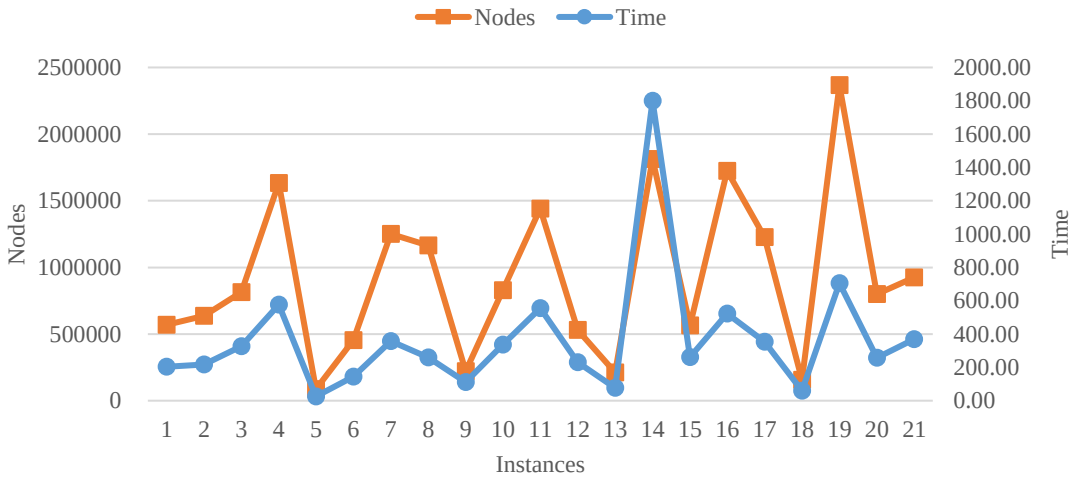

Chart 5.2-1 Base case- number of nodes and computational time

The huge variance in the computational time can be partly explained by the number of nodes explored by the solver before it stops or reaches optimality. Chart 5.2-1 indicates a strong positive correlation between the total number of explored nodes and the computational time.

Using the linear relaxation of the original model, all instances are all solved to optimality within a very short time of no more than 10ms. The relaxed problems yield an average objective value of 1684.799, i.e. 30.58% of the average objective value of the MIP problems. The huge differences in computational time and objective value indicate the complexity and burden introduced by the integrality conditions.

The second group of data, as illustrated in Chart 5.2-2, consists of five kinds of costs being balanced during the optimization process, except the variable production cost which is constant as previously explained. The following chart illustrates the relative compositions of total cost.

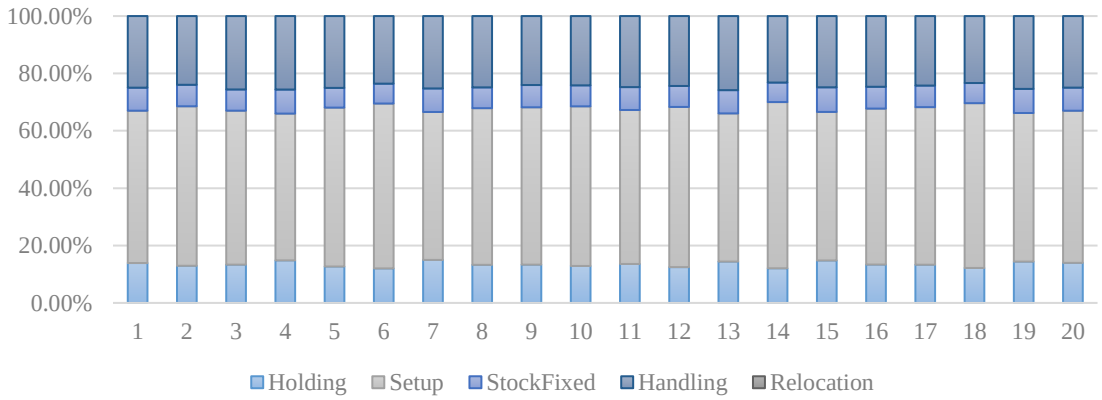

Chart 5.2-2 Base case - separated costs

Despite minor variations, the proportions of each type of cost remains stable across all 20 instances. Contributing the most to the total cost, the production setup cost varies around an average of 54.27%. The handling cost comes second with an average of 24.62%, indicating that a considerable number of units are put into the warehouse. The variable holding cost accounts for 13.45% of the total cost, while the 7.66% is paid as the fixed cost for storage locations. In the base case, the conditions are not strict enough to trigger the relocation. Hence, the relocation cost remains 0 for all the 20 instances.

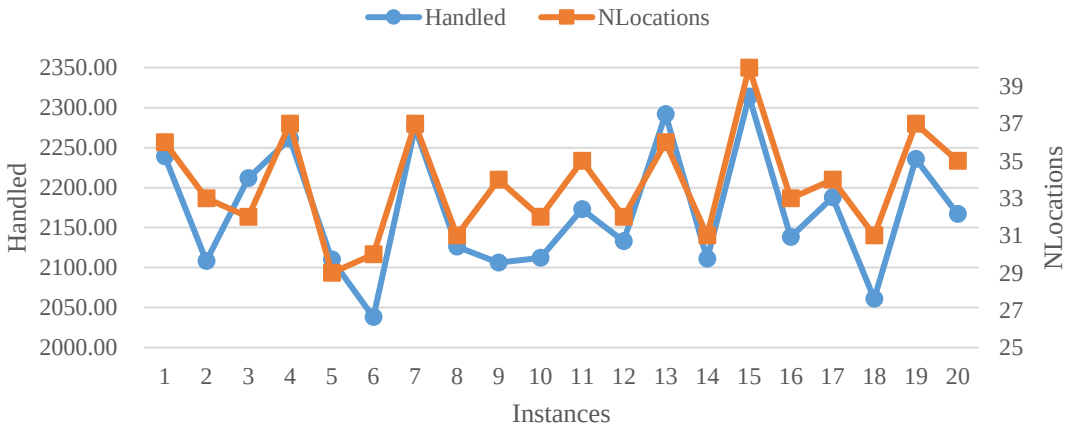

Chart 5.2-3 Base case - Handled units and number of locations

According to the total handled units and the total demand illustrated in Chart 5.2-3, on average, 57.26% of the products are stored in the warehouse before being sent to the customers. We count on average 30.4 production batches, which, considering the production capacity of 500 units, translates into 15,200 units of cumulated capacity. In comparison with the total demand of 3789.35 units, this indicates that each batch only uses an average of 24.93% of the production capacity.

5.3 Sensitivity analysis

We now present the results of the computational experiments. Considering the numerous parameters involved in the general model, we allow only one parameter to vary during each test to observe its influence on the performance and results of our model. The results of different parameter settings are compared to those of the base case, which allow us to carry out the related sensitivity analysis.

	Average Results								Relaxation	
Condition	**IP**	**IPTime**	**Best LB**	**S.O.[1]**	**W.T.[2]**	**Nodes**	**C.Gap[3]**	**A.Gap[4]**	**RelaxedOBJ**	**Time**
General Formulation										
Base case	5508.39	369.07	5507.71	1	0	924459.95	0.24%	0.012%	1684.80	0.007
Planning Horizon										
12 periods	4421.50	62.26	4421.50	0	0	263449.50			1537.65	0.008
20 periods	7322.97	1314.85	7317.39	11	0	1693739.40	0.14%	0.075%	1935.84	0.014
Production Setup										
Half setup cost	3617.00	3.25	3617.00	0	0	9743.30			851.46	0.008
Production Capacity										
375	5561.65	204.18	5561.65	0	0	609607.85			1801.61	0.007
750	5476.15	299.27	5476.10	1	0	872740.25	0.02%	0.001%	1626.76	0.009
Variable Holding										
0.0625	4837.37	322.04	4837.37	0	0	542744.85			1631.26	0.010
1	6942.44	13.26	6942.44	0	0	25097.95			1702.91	0.009
Handling Cost										
ha = 2x reloc	4357.83	452.09	4357.41	1	0	1006118.10	0.20%	0.010%	1585.18	0.007
ha = 6x reloc	4946.71	688.59	4945.75	3	0	1538050.40	0.13%	0.019%	1646.85	0.008
ha = 14x reloc	6038.49	277.83	6038.49	0	1	945808.45			1699.99	0.007
Location Combinations										
2S 1M 1L EOS	5415.51	12.41	5415.51	0	0	15793.35			1680.00	0.008
2S 1M 1L Normal	5480.49	31.65	5480.49	0	0	54871.45			1684.80	0.009
2S 1M 1L DOS	5492.50	40.23	5492.50	0	0	38723.55			1681.89	0.008
2L	5555.84	59.88	5555.84	0	0	119212.90			1684.80	0.007
Symmetry										
Integral storage	5724.79	1.64	5724.79	0	0	5300.65			1684.80	0.006
Initial symmetry breaking	5508.39	334.46	5508.39	0	0	989695.50			1684.80	0.008
Compatibility										
90% Overall	5508.99	327.34	5508.91	1	0	747518.35	0.03%	0.001%	1684.80	0.007
90% item-item	5508.99	367.30	5508.80	1	2	864081.15	0.07%	0.003%	1684.80	0.008
90% item-location	5508.39	287.96	5508.16	1	0	718273.40	0.08%	0.004%	1684.80	0.008
80% Overall	5515.55	720.15	5514.50	4	0	838247.50	0.10%	0.019%	1684.80	0.009
80% item-item	5515.48	1030.70	5512.90	7	1	1506278.45	0.13%	0.047%	1684.80	0.009
80% item-location	5508.39	100.57	5508.39	0	0	217649.50			1684.80	0.008
70% Overall	5523.52	816.44	5520.00	5	2	598691.00	0.25%	0.063%	1684.80	0.011
70% item-item	5519.20	1800.17	5508.79	20	0	1347052.95	0.19%	0.189%	1684.80	0.011
70% item-location	5509.13	79.77	5509.13	0	0	162615.45			1684.80	0.008
60% Overall	5572.39	821.76	5567.09	6	1	554675.60	0.32%	0.095%	1685.23	0.010
60% item-item	5528.41	1800.40	5498.46	20	0	770209.25	0.54%	0.539%	1684.80	0.015
60% item-location	5540.86	28.70	5540.86	0	0	45988.15			1685.23	0.008
No Relocation										
Base case No relocation	5508.39	145.15	5508.39	0	0	424077.60			1684.80	0.008
60% Overall No relocation	5575.69	666.00	5570.75	5	0	521057.20	0.35%	0.088%	1685.23	0.012
2S 1M 1L DOS No relocation	5496.68	58.54	5496.68	0	0	55003.45			1681.89	0.009

1 S.O.=Suboptimal, i.e. number of instances not solved to optimality within the time limit
2 W.T.=Within tolerance, i.e. number of instances finishing within the default tolerance of 1e-5
3 C.GAP=Conditional gap, i.e. average optimality gap among S.O. instances
4 A.GAP=Average gap, i.e. average optimality gap among all 20 instances

Table 5.3-1 Average results of all computational tests

	Costs Structure					Other Parameters						
Condition	**Holding**	**Setup**	**Stock Fixed**	**Handling**	**Relocation**	**Handled**	**Relocated**	**NSetup**	**NLocation**	**Total Space**	**Used Space**	**Utilization**
General Formulation												
Base case	13.45%	54.27%	7.66%	24.62%	0.000%	2170.00	0.00	30.40	33.75	3375.00	2964.45	87.84%
Planning Horizon												
12 periods	13.41%	54.49%	7.65%	24.45%	0.000%	1730.00	0.00	24.50	27.05	2705.00	2371.30	87.66%
20 periods	14.02%	52.82%	7.92%	25.24%	0.000%	2957.65	0.00	39.30	46.40	4640.00	4105.95	88.49%
Production Setup												
Half setup cost	6.05%	75.44%	3.37%	15.14%	0.000%	876.00	0.00	55.90	9.75	975.00	876.00	89.85%
Production Capacity												
375	13.43%	54.87%	7.52%	24.18%	0.000%	2151.90	0.00	30.95	33.45	3345.00	2987.15	89.30%
750	14.13%	52.33%	7.95%	25.58%	0.000%	2241.10	0.00	29.10	34.85	3485.00	3096.15	88.84%
Variable Holding												
0.0625	5.42%	50.86%	11.91%	31.81%	0.000%	2143.49	0.00	25.00	46.10	4610.00	4197.75	91.06%
1	18.75%	66.73%	2.80%	11.72%	0.000%	1626.89	0.00	47.25	15.55	1555.00	1301.95	83.73%
Handling Cost												
ha = 2x reloc	21.17%	60.17%	11.79%	6.87%	0.000%	2395.90	0.00	26.55	41.10	4110.00	3690.15	89.78%
ha = 6x reloc	17.06%	55.80%	9.65%	17.49%	0.000%	2307.15	0.00	28.00	38.20	3820.00	3375.70	88.37%
ha = 14x reloc	10.93%	53.07%	6.23%	29.77%	0.000%	2054.55	0.00	32.55	30.10	3010.00	2640.05	87.71%
Location Combinations												
2S 1M 1L EOS	14.15%	54.07%	6.31%	25.46%	0.011%	2206.30	4.68	29.75	21.80	3347.50	3064.75	91.55%
2S 1M 1L Normal	13.72%	53.90%	7.38%	24.99%	0.010%	2191.20	4.20	30.00	28.95	3237.50	3006.90	92.88%
2S 1M 1L DOS	12.73%	56.19%	6.83%	24.21%	0.050%	2127.15	22.02	31.35	37.85	3022.50	2796.65	92.53%
2L	12.75%	55.62%	7.69%	23.93%	0.000%	2127.30	0.10	31.40	17.10	3420.00	2833.85	82.86%
Symmetry												
Integral storage	16.02%	46.23%	12.01%	25.73%	0.000%	2357.00	0.00	26.85	13.75	5500.00	3669.45	66.72%
Initial symmetry breaking	13.45%	54.27%	7.66%	24.62%	0.000%	2170.00	0.00	30.40	33.75	3375.00	2964.45	87.84%
Compatibility												
90% Overall	13.41%	54.34%	7.66%	24.60%	0.000%	2167.95	0.00	30.45	33.75	3375.00	2955.20	87.56%
90% item-item	13.41%	54.34%	7.66%	24.60%	0.000%	2167.95	0.00	30.45	33.75	3375.00	2955.20	87.56%
90% item-location	13.45%	54.27%	7.66%	24.62%	0.000%	2170.00	0.00	30.40	33.75	3375.00	2964.45	87.84%
80% Overall	13.57%	54.00%	7.74%	24.69%	0.001%	2178.85	0.55	30.30	34.15	3415.00	2993.30	87.65%
80% item-item	13.57%	54.00%	7.74%	24.69%	0.000%	2178.85	0.00	30.30	34.15	3415.00	2993.30	87.65%
80% item-location	13.45%	54.27%	7.66%	24.62%	0.000%	2170.00	0.00	30.40	33.75	3375.00	2964.45	87.84%
70% Overall	13.36%	54.40%	7.68%	24.54%	0.017%	2168.50	7.42	30.55	33.95	3395.00	2951.60	86.94%
70% item-item	13.37%	54.34%	7.71%	24.58%	0.000%	2170.65	0.05	30.50	34.05	3405.00	2952.05	86.70%
70% item-location	13.45%	54.27%	7.66%	24.62%	0.000%	2170.25	0.00	30.40	33.75	3375.00	2964.20	87.83%
60% Overall	13.02%	54.98%	7.95%	24.02%	0.025%	2141.80	11.05	31.05	35.45	3545.00	2901.45	81.85%
60% item-item	13.58%	53.75%	7.95%	24.72%	0.005%	2186.90	2.07	30.20	35.15	3515.00	3002.85	85.43%
60% item-location	13.13%	55.15%	7.55%	24.16%	0.009%	2141.85	4.10	31.05	33.45	3345.00	2911.05	87.03%
No Relocation												
Base case	13.45%	54.27%	7.66%	24.62%	0.000%	2170.00	0.00	30.40	33.75	3375.00	2964.45	87.84%
60% Overall	12.80%	55.41%	7.90%	23.89%	0.000%	2132.81	0.00	31.30	35.25	3525.00	2854.20	80.97%
2S 1M 1L DOS	12.66%	56.37%	6.82%	24.15%	0.000%	2125.52	0.00	31.45	37.90	3025.00	2782.45	91.98%

Table 5.4 1 Average results of all computational tests (continued)

5.3.1 Test framework

In order to facilitate the comparisons and analysis, we group our tests according to the tested parameters or features. Table 5.3-2summarizes the logic and structure of the tests:

	Base case	Sensitivity analysis
Planning horizon	15	12, 20
Production setup cost	[80, 120]	[40, 60]
Production capacity	500	375, 750
Variable holding cost	0.25	0.0625, 1
Handling cost	0.625	0.125, 0.375, 0.875
Location combinations	4M	2S 1M 1L, 2L
Cost structure for storage fixed cost	—	2S 1M 1L EOS (Economies of scale) 2S 1M 1L Normal (Proportional) 2S 1M 1L EOS (Diseconomies of scale)
Symmetry	4M (identical locations)	Integral storage, Initial symmetry breaking
Compatibilities	Perfect compatibility (100%)	90% overall, 90% item-item, 90% item-location 80% overall, 80% item-item, 80% item-location 70% overall, 70% item-item, 70% item-location 60% overall, 60% item-item, 60% item-location
No relocation	Allowing relocation	Base case no relocation, 60% overall compatibility no relocation, 2S 1M 1L DOS no relocation

Table 5.3-2 Test framework

Additionally, we also test the transportation formulation to observe its performance. The results are compared with those of the general formulation in the same selected scenarios. In this group of tests, we include the following data settings: Base case, 2S 1M 1L Normal, 90% Overall compatibility, 80% Overall compatibility, 70% Overall compatibility and 60% Overall compatibility.

5.3.2 Planning horizon

Although the preliminary tests indicate that 15 periods appear to be appropriate for our analysis, we decide to compare the results of 12 periods and 20 periods with the base in order to demonstrate the impact of changing the planning horizon.

	Average Results							
Condition	IP	IPTime	Best LB	S.O.	W.T.	Nodes	C.Gap	A.Gap
12 periods	4421.50	62.26	4421.50	0	0	263449.50		
Base case (15 periods)	5508.39	369.07	5507.71	1	0	924459.95	0.24%	0.012%
20 periods	7322.97	1314.85	7317.39	11	0	1693739.40	0.14%	0.075%

Table 5.3-3 Planning horizon - average results

Since all our data settings share the same demand, the total demand is positively related to the planning horizon. The resulting objective value therefore follows an increasing trend. As the problem size grows, the computation takes more time in order to explore the additional nodes, leading to more S.O. instances as well. In the case of 20 periods, although the S.O. instances have a smaller conditional average gap, the global average gap across all instances appear to be larger.

	Separated Costs				
Condition	Holding	Setup	StockFixed	Handling	Relocation
12 periods	13.41%	54.49%	7.65%	24.45%	0.000%
Base case (15 periods)	13.45%	54.27%	7.66%	24.62%	0.000%
20 periods	14.02%	52.82%	7.92%	25.24%	0.000%

	Other Parameters						
Condition	Handled	Relocated	NSetup	NLocation	TotalSpace	UsedSpace	Utilization
12 periods	1730.00	0.00	24.50	27.05	2705.00	2371.30	87.66%
Base case (15 periods)	1185.78	0.00	30.40	33.75	3375.00	2964.45	87.84%
20 periods	2957.65	0.00	39.30	46.40	4640.00	4105.95	88.49%

Table 5.3-4 Planning horizon - separated costs and other parameters

As shown in Table 5.3-4, as we extend the planning horizon, the production setup cost represents a smaller percentage of the total cost, while all the three inventory-related costs occupies increased proportions. This indicates that allows us to further benefit from the inventory by reducing the production frequency and raising the inventory level. Another proof is the increasing utilization rate of space.

5.3.3 Production setup cost

One of the major trade-offs in the production planning problems is between the production frequency and the size of inventory. In order to observe the impact of different production frequencies on our model, we test a scenario in which the production setup cost is halved with regard to the base case.

	Average Results							
Condition	IP	IPTime	Best LB	S.O.	W.T.	Nodes	C.Gap	A.Gap
Base case	5508.38	369.07	5507.70	1	0	924459.95	0.24%	0.012%
Half setup cost	3617.00	3.24	3617.000	0	0	9743.30		

Table 5.4-2 Production setup cost – average results

As shown in the Table 5.4-2, the greatest impact of the lowered setup cost is on the computational time and the number of nodes. 98.94% less nodes are explored by the solver before reaching optimality, resulting in 99.12% of computational time saved. The only instance exceeding the time limit in the base case is now solved to optimality as well. The reduction in production setup cost also leads to a 34.34% drop in the total cost.

This large reduction in computational burden might be explained by the production-inventory trade-off. Given a much higher production frequency, a larger part of our demand is fulfilled through direct flows. Consequently, for this part of demand, there is no longer a need to decide to which locations we should assign the products, or from which locations we should take out the extract inventory.

	Separated Costs				
Condition	Holding	Setup	StockFixed	Handling	Relocation
Base case	13.45%	54.27%	7.66%	24.62%	0.00%
Half setup cost	6.06%	75.44%	3.37%	15.13%	0.00%

	Other Parameters						
Condition	Handled	Relocated	NSetup	NLocation	TotalSpace	UsedSpace	Utilization
Base case	2170.00	0.00	30.40	33.75	3375.00	2964.45	87.84%
Half setup cost	876.00	0.00	55.90	9.75	975.00	876.00	89.85%

Table 5.3-5 Production setup cost - separated cost and other parameters

Taking a closer look at the cost structure, we can see that reducing the production setup cost results in an increase in the total number of setups and a significant drop in the variable holding cost. Moreover, other inventory-related costs have been reduced as well.

Since the handling cost is linked to the overall inflow, the total handled units reflect directly the total amount of products put into the storage. As a result of a higher production frequency, the number of total stored units is reduced by 59.63%. However, the total storage space used indicates that the overall inventory has reduced by 70.45% compared to the base case. The average inventory level has seen a larger decrease than the average flow rate indicated by the total stored units. According to Little's Law, the inventory storage can be seen as a process, in which $Flow\ time = \frac{Inventory\ level}{Flow\ rate}$. Considering the different percentages of decrease in the inventory level and in the average flow rate, reducing the production setup cost also results in a shorter inventory flow time.

At last, a decrease of 71.11% can also be observed in the total number of locations used throughout the planning horizon, leading to an economy of the same percentage in the stock fixed cost. Since all four storage locations are medium locations, which is the same as in the base case, the total available space is always 100 times the number of locations except in later scenarios with alternative location combinations.

5.3.4 Production capacity

In comparison with the base case where the production capacity is set to 500, we tested two alternative settings: low capacity (375) and high capacity (750).

	Average Results							
Condition	IP	IPTime	Best LB	S.O.	W.T.	Nodes	C.Gap	A.Gap
Low (375)	5561.65	204.18	5561.65	0	0	609607.85		
Base case (500)	5508.39	369.07	5507.71	1	0	924459.95	0.24%	0.012%
High (750)	5476.15	299.27	5476.10	1	0	872740.25	0.02%	0.001%

Table 5.3-6 Production capacity - average results

By expanding the production capacity, we loosen the constraints for the production planning. It is therefore reasonable to see the objective value and the best bounds improve as the production capacity increases.

A higher production capacity also facilitates the calculation process. By exploring less nodes, the solver saves 18.9% of time on each instance. There is still one instance not solved to optimality, but showing a smaller gap than in the base case. Unexpectedly, the stricter condition in the low capacity case does not add computational burden to the calculation. On the contrary, in this case, the solver ran through the least nodes among all three cases, spending only 204.18 seconds per instance.

	Separated Costs				
Condition	Holding	Setup	StockFixed	Handling	Relocation
Low (375)	13.43%	54.87%	7.52%	24.18%	0.000%
Base case (500)	13.45%	54.27%	7.66%	24.62%	0.000%
High (750)	14.13%	52.33%	7.95%	25.58%	0.000%

	Other Parameters						
Condition	Handled	Relocated	NSetup	NLocation	TotalSpace	UsedSpace	Utilization
Low (375)	2151.90	0.00	30.95	33.45	3345.00	2987.15	89.30%
Base case (500)	2170.00	0.00	30.40	33.75	3375.00	2964.45	87.84%
High (750)	2241.10	0.00	29.10	34.85	3485.00	3096.15	88.84%

Table 5.3-7 Production capacity - separated costs and other parameters

A high production capacity allows larger batches at a lower frequency, bringing down the total number of setups and the total setup cost. In general, this extra capacity contributes to a higher inventory instead of direct flows, which is indicated by the increases in all the inventory-related costs and parameters except the space utilization rate. Following the same logics, a low production capacity impacts the results in an opposite way.

5.3.5 Variable holding cost

We test three different values for the per-unit variable holding cost.

	Average Results							
Condition	IP	IPTime	Best LB	S.O.	W.T.	Nodes	C.Gap	A.Gap
0.0625	4837.37	322.04	4837.37	0	0	542744.85		
Base case (0.25)	5508.39	369.07	5507.71	1	0	924459.95	0.24%	0.012%
1	6942.44	13.26	6942.44	0	0	25097.95		

Table 5.3-8 Variable holding cost - average results

The results shown in Table 5.3-8 indicates that the variable holding cost has a significant positive correlation with the total cost. However, it impacts the computational burden differently. While lowering the per-unit variable holding cost results in a shorter CPU

time and fewer nodes explored, raising this parameter leads to a much larger reduction in the computational time and in the number of nodes.

	Separated Costs				
Condition	Holding	Setup	StockFixed	Handling	Relocation
0.0625	5.42%	50.86%	11.91%	31.81%	0.000%
Base case (0.25)	13.45%	54.27%	7.66%	24.62%	0.000%
1	18.75%	66.73%	2.80%	11.72%	0.000%

	Other Parameters						
Condition	Handled	Relocated	NSetup	NLocation	TotalSpace	UsedSpace	Utilization
0.0625	2143.49	0.00	25.00	46.10	4610.00	4197.75	91.06%
Base case (0.25)	2170.00	0.00	30.40	33.75	3375.00	2964.45	87.84%
1	1626.89	0.00	47.25	15.55	1555.00	1301.95	83.73%

Table 5.3-9 Variable holding cost - separated costs and other parameters

A closer look at the detailed parameters may help us explain the above-mentioned phenomenon. Since it cost more to keep inventory, the total number of handled units has dropped by approximately 25% as compared to the base case. This lower overall inventory level requires less than half the number of locations, which has an positive impact in the CPU time. Besides, as a consequence of a lower inventory level, the case of high per-unit variable holding cost is also characterized by a more frequent production.

5.3.6 Handling cost

As the second largest among all costs, the handling cost is another factor to consider when deciding whether to build up a higher inventory. The unit handling cost in the base case is set to 10 times the unit relocation cost. Alternative settings of this parameter are 2 time (lowest), 6 times (lower) and 14 times (high) the unit relocation cost.

	Average Results							
Condition	IP	IPTime	Best LB	S.O.	W.T.	Nodes	C.Gap	A.Gap
2x	4357.83	452.09	4357.41	1	0	1006118.10	0.20%	0.010%
6x	4946.71	688.59	4945.75	3	0	1538050.40	0.13%	0.019%
10x (Base Case)	5508.39	369.07	5507.71	1	0	924459.95	0.24%	0.012%
14x	6038.494	277.832	6038.494	0	1	945808.45		

Table 5.3-10 Handling cost - average results

Table 5.3-10 shows the average results of each tested data settings. As the unit handling cost increases, we see an obvious increasing trend in the objective value and the best

bounds. However, the computational time and the number of nodes seem to peak in the 2nd case where the unit handling cost equals 6 times the unit relocation cost. Changing this parameter in either direction within the tested range reduces the of burden of computation.

	Separated Costs				
Condition	Holding	Setup	StockFixed	Handling	Relocation
2x	21.17%	60.17%	11.79%	6.87%	0.00%
6x	17.06%	55.80%	9.65%	17.49%	0.00%
10x (Base Case)	13.45%	54.27%	7.66%	24.62%	0.00%
14x	10.93%	53.07%	6.23%	29.77%	0.00%

	Other Parameters						
Condition	Handled	Relocated	NSetup	NLocation	TotalSpace	UsedSpace	Utilization
2x	2395.90	0.00	26.55	41.10	4110.00	3690.15	89.78%
6x	2307.15	0.00	28.00	38.20	3820.00	3375.70	88.37%
10x (Base Case)	2170.00	0.00	30.40	33.75	3375.00	2964.45	87.84%
14x	2054.55	0.00	32.55	30.10	3010.00	2640.05	87.71%

Table 5.3-11 Handling cost - separated costs and other parameters

As the unit handling cost increases, this portion grows as its absolute value rise as well. Meanwhile, it becomes less and less interesting to keep products in inventory, which directly leads to a drop in the number of handled units and in the total holding cost. A smaller inventory requires less space. Hence, the total number of open locations and the storage fixed cost are reduced as well.

The reduction in inventory is made possible by a more frequent production. It is therefore easy to explain the increase in the number of production setup. However, the percentage of this cost follows an opposite trend, dropping from 60.17% to 53.07%. The cause for this seemingly contradictory result is the total handling cost which rises drastically, out-pacing the increase rate of the total setup cost.

5.3.7 Location combinations

We first compare the result of the three combinations of storage locations. In all three cases, the fixed cost of each type of location is proportional to its individual capacity. Including the base case, all three settings have the same overall storage capacity of 400 units.

	Average Results							
Condition	IP	IPTime	Best LB	S.O.	W.T.	Nodes	C.Gap	A.Gap
Base case 4M	5508.39	369.07	5507.71	1	0	924459.95	0.24%	0.012%
Normal 2S 1M 1L	5480.49	31.65	5480.49	0	0	54871.45		
2L	5555.84	59.88	5555.84	0	0	119212.90		

Table 5.3-12 Location combination - average results

Although the alternative combinations do not have a significant impact on the total cost or on the best bounds, they result in a largely reduced computational time, allowing all instances to reach optimality in the two alternative cases.

A possible explanation for this behavior is the symmetry between storage locations. In the base case, the four storage locations are identical in terms of compatibility, fixed cost, and individual capacity. This means that for a given solution (i.e. an allocation of inventory to a specific location), the locations can be permuted to obtain an equivalent solution. Without additional differentiating constraints, these locations will generate symmetric nodes in the search tree. Due to these duplicate nodes, much more CPU time is required to solve the problem to optimality. Comparing the base case with the first alternative setting (2S 1M 1L), we can see that differentiating the storage locations allows the solver to explore 94.06% fewer nodes before reaching optimality. The comparison between the base case and the second alternative setting (2L) indicates that the computational burden escalates drastically as the system is broken into more symmetric elements.

	Separated Costs				
Condition	Holding	Setup	StockFixed	Handling	Relocation
Base case 4M	13.45%	54.27%	7.66%	24.62%	0.000%
Normal 2S 1M 1L	13.72%	53.90%	7.38%	24.99%	0.010%
2L	12.75%	55.62%	7.69%	23.93%	0.000%

	Other Parameters						
Condition	Handled	Relocated	NSetup	NLocation	TotalSpace	UsedSpace	Utilization
Base case 4M	2170.00	0.00	30.40	33.75	3375.00	2964.45	87.84%
Normal 2S 1M 1L	2191.20	4.20	30.00	28.95	3237.50	3006.90	92.88%
2L	2127.30	0.10	31.40	17.10 (30 max)	3420.00	2833.85	82.86%

Table 5.3-13 Location combinations - separated costs and other parameters

By analyzing these parameters, we first notice that, given the same overall storage capacity, a more flexible combination leads to more handled units. The most flexible combination is the first alternative setting (2S 1M 1L) where different types of locations can be combined as needed, while the least flexible is the second alternative (2L) in which the overall storage capacity can only take three values, i.e. 0, 200, and 400. This is reasonable since, in some periods, the fixed cost of one additional medium or large location might undermine the choice of further increasing inventory. The total variable holding cost follows the same trend. As a result of a higher inventory, more savings can be made on the total setup cost through less frequent production batches.

Another benefit of combining different types of locations is the higher utilization rate of space. Although the solution of the first alternative (2S 1M 1L) suggests the highest inventory among all three cases, it also uses the smallest storage space, therefore generating the least storage fixed cost.

5.3.8 Cost structure for storage fixed costs

We also a run groups of tests on the structure of storage fixed costs using the first alternative combination 2S 1M 1L to observe the impacts of economy of scale (EOS) and diseconomy of scale (DOS).

	Average Results							
Condition	IP	IPTime	Best LB	S.O.	W.T.	Nodes	C.Gap	A.Gap
2S 1M 1L EOS	5415.51	12.41	5415.51	0	0	15793.35		
2S 1M 1L Normal	5480.49	31.65	5480.49	0	0	54871.45		
2S 1M 1L DOS	5492.50	40.23	5492.50	0	0	38723.55		

Table 5.3-14 Storage fixed cost - average results

As shown in the table above, an EOS environment generates the total cost, while requiring the least time for the computation. A DOS environment generates extra cost, while consuming more time on the computation. However, under the normal cost structure where locational fixed costs are proportional with the individual storage capacity, the solver has gone through the largest number of nodes.

	Separated Costs				
Condition	Holding	Setup	StockFixed	Handling	Relocation
2S 1M 1L EOS	14.15%	54.07%	6.31%	25.46%	0.011%
2S 1M 1L Normal	13.72%	53.90%	7.38%	24.99%	0.010%
2S 1M 1L DOS	12.73%	56.19%	6.83%	24.21%	0.050%

	Other Parameters						
Condition	Handled	Relocated	NSetup	NLocation	TotalSpace	UsedSpace	Utilization
2S 1M 1L EOS	2206.30	4.68	29.75	21.80	3347.50	3064.75	91.55%
2S 1M 1L Normal	2191.20	4.20	30.00	28.95	3237.50	3006.90	92.88%
2S 1M 1L DOS	2127.15	22.02	31.35	37.85	3022.50	2796.65	92.53%

Table 5.3-15 Storage fixed cost - separated costs and other parameters

The locational fixed costs influence directly the number of opened locations. The EOS favors larger locations over small ones. Therefore, in this case, the least number of locations are used throughout the 20 periods, yet providing the largest total opened space. As a result, even though more units are stored in the warehouse, the EOS structure has led to a lower space utilization rate. As in the previous test, the higher inventory under the EOS helps reduce the number of production setups.

In the case of DOS, smaller locations are prioritized. Although, in this case, a largest number of locations are opened, the total storage fixed cost appears lower than in the normal case because small locations are used much more frequently.

It is also worth noticing that all three cases discussed here have involved certain relocation. Despite the low quantity of relocated units, their existence indicates that the relocation is not a movement forced by the strict compatibility constraints. Even under perfect compatibility, relocations can be included in the optimal solution.

5.3.9 Symmetry

Based on the results observed in the previous tests, we decide to run an additional group of tests to examine the impact of symmetry on the calculation. Two alternative cases are included in this group: the first case features one integral storage space with a fixed cost of 50 and a capacity of 400 units (i.e. four medium locations combined). In the second case, new constraints are added for the first time period to break the symmetry by imposing a sequence for storage locations $(z_{11} \geq z_{21} \geq z_{31} \geq z_{41})$. However, since

relocating inventory between locations will generate certain costs, we cannot break the temporal linkage of inventory in each location. This symmetry breaking technique can only be applied to the first period in our model.

	Average Results							
Condition	IP	IPTime	Best LB	S.O.	W.T.	Nodes	C.Gap	A.Gap
Base case	5508.39	369.07	5507.71	1	0	924459.95	0.24%	0.012%
Integral storage	5724.79	1.64	5724.79	0	0	5300.65		
Initial symmetry breaking	5508.39	334.46	5508.39	0	0	989695.50		

Table 5.3-16 Symmetry - average results

Table 5.3-16 shows the average results of the problem. Comparing the base case to the case of integral storage, we first notice a reduction of 99.56% in CPU time and 99.43% in nodes explored, while all instances are solved to optimality. These results, coupled with the results of alternative location combinations, provide further evidence that the symmetry among identical locations significantly increases the computational burden.

Moreover, the integration of storage locations has greatly reduced the flexibility of inventory management to optimize costs. Regardless of the inventory size, the whole storage space has to be opened, generating a high fixed cost. This has led to a higher objective value.

The second alternative case shows that the addition of symmetry-breaking constraints for the first time period also help to reduce the CPU time. However, on average, the solver has run through more nodes than in the base case.

	Separated Costs				
Condition	Holding	Setup	StockFixed	Handling	Relocation
Base case	13.45%	54.27%	7.66%	24.62%	0.000%
Integral storage	16.02%	46.23%	12.01%	25.73%	0.000%
Initial SB	13.45%	54.27%	7.66%	24.62%	0.000%

	Other Parameters						
Condition	Handled	Relocated	NSetup	NLocation	TotalSpace	UsedSpace	Utilization
Base case	2170.00	0.00	30.40	33.75	3375.00	2964.45	87.84%
Integral storage	2357.00	0.00	26.85	13.75 (15 max)	5500.00	3669.45	66.72%
Initial SB	2170.00	0.00	30.40	33.75	3375.00	2964.45	87.84%

Table 5.3-17 Symmetry - separated costs and other parameters

Taking a closer look at the separated costs and other parameters, we can see that integrating the storage locations leads to a much higher storage fixed cost. In addition, more units are kept as inventory, therefore generating a higher variable holding cost and a higher handling cost. Since the whole storage space is opened once the storage fixed cost is paid, it is thus intuitively reasonable to make the most of this space, rather than keeping a relatively smaller inventory. Although more units are stored, there is still a significant drop in space utilization rate. The higher inventory level has also led to a higher production frequency and a higher total production setup cost.

5.3.10 Compatibilities

We first compare the results of overall compatibility changes, where the item-item compatibility is at the same percentages level as the item-location compatibility, e.g. 80% overall compatibility means that in this case, the both compatibility levels are at 80%.

	Average Results							
Condition	IP	IPTime	Best LB	S.O.	W.T.	Nodes	C.Gap	A.Gap
Base case	5508.39	369.07	5507.71	1	0	924459.95	0.24%	0.012%
90% Overall	5508.99	327.34	5508.91	1	0	747518.35	0.03%	0.001%
80% Overall	5515.55	720.15	5514.50	4	0	838247.50	0.09%	0.019%
70% Overall	5523.52	816.44	5520.00	5	2	598691.00	0.25%	0.063%
60% Overall	5572.39	821.76	5567.09	6	1	554675.60	0.32%	0.095%

Table 5.3-18 Overall compatibilities - average results

As the overall level of compatibility decreases, the problem becomes more and more restricted, resulting in an increasing objective value. The computational burden increases considerably as well. In the extreme case of 60% compatibility, the CPU time has more than doubled in comparison to the base case, while 6 instances could not be solved to optimality within the time limit. In addition, the lowering overall compatibility level leads to increasing optimality gaps among all instances, as well as among suboptimal instances.

However, the extra computational burden does not arise from more nodes to explore. On the contrary, the average number of nodes explored decreases as we lower the overall compatibility level. Considering the time increase, this means that the solver has to spend much more time on each node.

We also notice that, compared to the base case, the case of 90% overall compatibility is even faster to solve with less explored nodes. This improvement in computational speed could possibly be explained by the slightly lower compatibility level. As we see in the previous analysis, the symmetry between locations has greatly contributed to the computational burden in the base case. With a 10% lower level of compatibility, the four locations can be differentiated. In this case, the effect of this differentiation outweighs the negative impact of tighter constraints, resulting in a shorting computational time. But the effect of symmetry-breaking is relatively limited in others cases of this test.

	Separated Costs				
Condition	Holding	Setup	StockFixed	Handling	Relocation
Base case	13.45%	54.27%	7.66%	24.62%	0.000%
90% Overall	13.41%	54.34%	7.66%	24.60%	0.000%
80% Overall	13.57%	54.00%	7.74%	24.69%	0.001%
70% Overall	13.36%	54.40%	7.68%	24.54%	0.017%
60% Overall	13.02%	54.98%	7.95%	24.02%	0.025%

	Other Parameters						
Condition	Handled	Relocated	NSetup	NLocation	TotalSpace	UsedSpace	Utilization
Base case	2170.00	0.00	30.40	33.75	3375.00	2964.45	87.84%
90% Overall	2167.95	0.00	30.45	33.75	3375.00	2955.20	87.56%
80% Overall	2178.85	0.55	30.30	34.15	3415.00	2993.30	87.65%
70% Overall	2168.50	7.42	30.55	33.95	3395.00	2951.60	86.94%
60% Overall	2141.80	11.05	31.05	35.45	3545.00	2901.45	81.85%

Table 5.3-19 Overall compatibilities - separated costs and other parameters

From Table 5.3-19, we can see that the cost structure remains stable across all these 5 cases. As the compatibility level drops, an increasing number of units are relocated between different storage locations. Although this cost is very low when compared to other types of cost, its increasing trend indicates that, under strict storage conditions, allowing relocation can be beneficial.

In case of a low compatibility level, the storage space is no longer available to all types of item as in the case of perfect compatibility. As a result, the overall inventory level drops while more locations are opened. In addition, the low compatibility level also leads to the deteriorating space utilization rate.

In order to observe the different impacts of the item-item compatibility and item-location compatibility, we ran another group of tests, separating the two types of compatibility conditions.

	Average Results							
Condition	**IP**	**IPTime**	**Best LB**	**S.O.**	**W.T.**	**Nodes**	**C.Gap**	**A.Gap**
Perfect Compatibility (Base case)	5508.39	369.07	5507.71	1	0	924459.95	0.24%	0.012%
90% Overall	5508.99	327.34	5508.91	1	0	747518.35	0.03%	0.001%
90% item-item	5508.99	367.30	5508.80	1	2	864081.15	0.07%	0.003%
90% item-location	5508.39	287.96	5508.16	1	0	718273.40	0.08%	0.004%
80% Overall	5515.55	720.15	5514.50	4	0	838247.50	0.09%	0.019%
80% item-item	5515.48	1030.70	5512.90	7	1	1506278.45	0.13%	0.047%
80% item-location	5508.39	100.57	5508.39	0	0	217649.50		
70% Overall	5523.52	816.44	5520.00	5	2	598691.00	0.25%	0.063%
70% item-item	5519.20	1800.17	5508.79	20	0	1347052.95	0.19%	0.189%
70% item-location	5509.13	79.77	5509.13	0	0	162615.45		
60% Overall	5572.39	821.76	5567.09	6	1	554675.60	0.32%	0.095%
60% item-item	5528.41	1800.40	5498.46	20	0	770209.25	0.54%	0.539%
60% item-location	5540.86	28.70	5540.86	0	0	45988.15		

Table 5.3-20 Separated compatibilities - average results

In all four item-item cases, the storage locations are perfectly compatible with any item. Therefore, lowering the item-item compatibility only adds to the complexity of the problem without breaking the symmetry between locations. Consequently, these four cases all take much longer time to solve. In the 70% and 60% item-item cases, none of the instances can be solved to optimality within the time limit.

A lower item-location compatibility level, however, not only tightens the compatibility constraints, but also breaks the symmetry between the storage locations. The computational time, as well as the number of nodes, is both largely reduced in these cases.

5.3.11 Disabling item relocation

To further demonstrate the impact of item relocation, we choose three cases to rerun without allowing any relocation, including the base case and two other cases with relatively significant relocation activities.

	Average Results							
Condition	IP	IPTime	Best LB	S.O.	W.T.	Nodes	C.Gap	A.Gap
Base case	5508.39	369.07	5507.71	1	0	924459.95	0.24%	0.012%
Base case No relocation	5508.39	145.15	5508.39	0	0	424077.60		
2S 1M 1L DOS	5492.50	40.23	5492.50	0	0	38723.55		
2S 1M 1L DOS No relocation	5496.68	58.54	5496.68	0	0	55003.45		
60% Overall	5572.39	821.76	5567.09	6	1	554675.60	0.32%	0.095%
60% Overall No relocation	5575.69	666.00	5570.75	5	0	521057.20	0.35%	0.088%

Table 5.3-21 Disabling item relocation - average results

When item relocation is forbidden, the number of decision variables is in fact reduced, therefore simplifying the problem. Hence, we can observe a large reduction in computational time both in the base case where the original solution does not include any relocation, and in the case of 60% overall compatibility.

	Separated Costs				
Condition	Holding	Setup	StockFixed	Handling	Relocation
Base case	13.45%	54.27%	7.66%	24.62%	0.000%
Base casae No relocation	13.45%	54.27%	7.66%	24.62%	0.000%
2S 1M 1L DOS	12.73%	56.19%	6.83%	24.21%	0.050%
2S 1M 1L DOS No relocation	12.66%	56.37%	6.82%	24.15%	0.000%
60% Overall	13.02%	54.98%	7.95%	24.02%	0.025%
60% Overall No relocation	12.80%	55.41%	7.90%	23.89%	0.000%

	Other Parameters						
Condition	Handled	Relocated	NSetup	NLocation	TotalSpace	UsedSpace	Utilization
Base case	2170.00	0.00	30.40	33.75	3375.00	2964.45	87.84%
Base casae No relocation	2170.00	0.00	30.40	33.75	3375.00	2964.45	87.84%
2S 1M 1L DOS	2127.15	22.02	31.35	37.85	3022.50	2796.65	92.53%
2S 1M 1L DOS No relocation	2125.52	0.00	31.45	37.90	3025.00	2782.45	91.98%
60% Overall	2141.80	11.05	31.05	35.45	3545.00	2901.45	81.85%
60% Overall No relocation	2132.81	0.00	31.30	35.25	3525.00	2854.20	80.97%

Table 5.3-22 Disabling item relocation - separated costs and other parameters

Since the original solution to the base case does not include any relocation, its separated costs and all other parameters remain the same. In the other two cases, forbidding relocation imposes stricter conditions to manage the inventory. As a result, a reduced number of units are put into storage while the production setups become slightly more frequent. As a direct indicator of storage flexibility, the space utilization rate drops as well. The result of this group shows that allowing item relocation helps reduce the total cost.

5.3.12 Transportation reformulation

In the previous discussion, we proposed a transportation reformulation for the CLSP-MSL problem. Six cases (marked with "TP") are tested to illustrate the performance of this alternative formulation.

	Average Results								Relaxation		
Condition	IP	IPTime	Best LB	S.O.	W.T.	Nodes	C.Gap	A.Gap	RelaxedOBJ	Time	Status
Base case TP	5508.39	284.77	5508.31	1	0	280764.00	0.03%	0.001%	4227.18	0.031	1
Base case	5508.39	369.07	5507.71	1	0	924459.95	0.24%	0.012%	1684.80	0.007	1
2S 1M 1L Normal TP	5480.49	78.02	5480.49	0	0	50590.25			4227.18	0.037	1
2S 1M 1L Normal	5480.49	31.65	5480.49	0	0	54871.45			1684.80	0.009	1
90% Overall TP	5508.99	283.03	5508.99	0	0	239550.15			4227.18	0.033	1
90% Overall	5508.99	327.34	5508.91	1	0	747518.35	0.03%	0.001%	1684.80	0.007	1
80% Overall TP	5515.56	695.93	5513.94	4	0	270321.10	0.15%	0.029%	4227.18	0.034	1
80% Overall	5515.55	720.15	5514.50	4	0	838247.50	0.09%	0.019%	1684.80	0.009	1
70% Overall TP	5523.48	1077.32	5519.49	6	0	255863.65	0.24%	0.072%	4227.18	0.042	1
70% Overall	5523.52	816.44	5520.00	5	2	598691.00	0.25%	0.063%	1684.80	0.011	1
60% Overall TP	5572.43	820.89	5566.76	5	0	183600.40	0.40%	0.101%	4260.99	0.039	1
60% Overall	5572.39	821.76	5567.09	6	1	554675.60	0.32%	0.095%	1685.23	0.010	1

Table 5.3-23 Transportation formulation - average results and relaxation

Thanks to the lower upper bounds in the production setup constraints, solving the linear relaxation of the problem gives an improved objective in all the six cases. However, as previously discussed, there are a larger number of production variables in the TP, which requires more time to compute. It takes the solver 4 times longer to solve the linear relaxation using TP.

For the original problem, less nodes are explored using TP, with different impacts on the computational time. In the base case, using TP saves 22.84% of computational time, with a remarkable improvement in the conditional gap and in the average gap. In the case of 90% overall compatibility, a shorter computational time allows the S.O. instance to reach optimality. In the case of 80% overall compatibility, despite the smaller time consumption, the 4 S.O. instances show a larger conditional gap and average gap. In the rest three cases, solving the problem using TP costs more time than using the general formulation. These inconsistent impacts on CPU time is possibly due to the fact that CPLEX is automatically adding several cuts while solving the problem.

Chapter 6. Conclusions, limitations, and future research

6.1 Conclusions

In order to scientifically manage the manufacturing activities, researchers have extended the lot-sizing models in various directions. However, little effort has been addressed to the optimization of production and inventory at the same time. In this thesis, we focus on modeling the CLSP-MSL using MIP. In the CLSP-MSL, the storage space is divided into several individual locations. Inventory is assigned to the storage locations according to specific rules and conditions. By extending the traditional CLSP model with new constraints and parameters, we are able to formulate models for CLSP-MSL under complex storage conditions.

We first looked into the single-item version of the problem. Since only one item is involved, the assignment task is mainly based on the individual capacity and fixed cost of storage locations. However, in a multi-item production environment, additional factors have to be taken into consideration to optimize the production and storage process at the same time. The individual capacity remains important, but is complicated by the different space requirements of multiple items. We also assume that incompatibilities might exist between certain pairs of items, or between items and storage locations. Moreover, we also include the possibility to relocate inventory from one location to another. Item relocation is necessary in certain cases with strict compatibility constraints, while in other cases it allows a more efficient inventory management at a lower cost. A general model has been proposed for the CLSP-MSL. All situations discussed above including the original CLSP can be derived from this general model. We also reformulated the problem as a transportation problem (TP), which provides a better LP relaxation bound.

Such an integrated optimization model provides managers with a tool to coordinate their production lines with warehouses. Situations can thus be avoided where the additional storage cost of large batches overweighs the economy of scale. Moreover, by dividing the warehouse into several storage locations, the model serves as another step towards the complex reality. Storage conditions and related activities can now be mathematically modeled and optimized, while their impacts become visible and quantified.

In order to observe the impact of the various extensions on the CLSP-MSL model and the performance of the TP reformulation, we have carried out a series of computational tests. Due to the absence of existing data sets that contains the compatibility and relocation parameters discussed in this thesis, we generated our own instances according to results of the preliminary tests.

A base case has first been calculated as the benchmark. Next, a sensitivity analysis is done among all test groups. In each group, only the target parameter changes, generating several alternative scenarios. Solving the CLSP-MSL in these alternative data settings using the general model allows us to analyze the trade-offs between various parameters. Results show that, with the warehouse broken downs to several storage locations, we can organize the space more efficiently at a lower cost rather than running the warehouse as an integral storage. Allowing the inventory to be relocated at a cost also helps reduce the total cost by allowing more flexibility to decide which locations to open. However, in out tests, the cost of relocation remains trivial in comparison to other type of costs. Even in cases where we observe the highest number of relocated units, disabling the possibility of item relocation does not lead to a significant increase in the total cost.

It is also shown that the CPU time can be largely reduced when we include different types of storage locations in our decision, or when the locations are differentiated by the compatibility parameters. This leads us to the hypothesis that a large part of the computational burden is caused by the symmetry between these storage locations.

Solving the CLSP-MSL using the TP formulation yields similar results in terms of solution quality and CPU time for the MIP. But solving the linear relaxation of the problem gives a higher objective value, indicating that this reformulation provides better lower bounds than the original general formulation.

6.2 Limitations and future research

Even though in this research we have attempted to bring the lot-sizing models one step closer to the complex reality of an operating business, simplifications and assumptions have been made in order to control the scope of study and the size of the problem.

We have used a more complex cost structure with respect to the inventory related costs in comparison with the classical lot-sizing problems in which only a variable unit holding cost is taken into account. According to the theory of ABC, we can identify numerous cost-generating activities throughout the process. However, our discussion only involved several costs which occur in the warehouse including handling cost, fixed cost for locations, variable holding cost, and relocation cost. A possible path for future research is to formulate a more general model with a more detailed breakdown of various activities, e.g. fixed inventory cost covering a window of periods instead of only one period.

The proposed formulations are more general than the traditional CLSP which presume one general storage location and no fixed storage costs. However, these new formulations may need further modifications to accommodate certain products and industries with unique characteristics. For example, liquid products are usually stored in tanks. Keeping the same type of item in a tank for several consecutive periods may not generate extra fixed cost. In this case, the fixed cost of stock takes rather the form of product changeovers: when a different type of item needs to be stored in this location, the residual of the previous item must be cleaned up. Another situation is when the new item must be stored under different conditions, such as a different temperature, therefore requiring a cost-generating preparation before the new item can be stored.

Due to the limited computational capability of the machine and the test environment, we had to strictly control the size of the data sets. In certain test groups, the tested parameter varies within a selected range, while only several values are chosen for the experiments. The correlation between different parameters and the impact of changing these parameters will be better illustrated if more values can be tested. Moreover, the values given to all parameters were set to appropriate levels according to our preliminary tests. Altering these values may better reflect the characteristic of a specific industry, which might as well provide new insights of the studied subject, e.g. a scenario with significant inventory relocation.

References

Akbalik, Ayse, Bernard Penz and Christophe Rapine (2014). « Multi-item uncapacitated lot sizing problem with inventory bounds », *Optimization Letters*, vol. 9, no 1, p. 143-154.

Akbalik, Ayse, Bernard Penz and Christophe Rapine (2015). « Capacitated lot sizing problems with inventory bounds », *Annals of Operations Research*, vol. 229, no 1, p. 1-18.

Atamtürk, Alper and Simge Küçükyavuz (2005). « Lot sizing with inventory bounds and fixed costs: Polyhedral study and computation », *Operations Research*, vol. 53, no 4, p. 711-730.

Atamtürk, Alper and Simge Küçükyavuz (2008). « An o(n2) algorithm for lot sizing with inventory bounds and fixed costs », *Operations Research Letters*, vol. 36, no 3, p. 297-299.

Azzi, Anna, Daria Battini, Maurizio Faccio, Alessandro Persona and Fabio Sgarbossa (2014). « Inventory holding costs measurement: A multi-case study », *The International Journal of Logistics Management*, vol. 25, no 1, p. 109-132.

Berling, Peter (2008). « Holding cost determination: An activity-based cost approach », *International Journal of Production Economics*, vol. 112, no 2, p. 829-840.

Chu, Chengbin, Feng Chu, Jinhong Zhong and Shanlin Yang (2013). « A polynomial algorithm for a lot-sizing problem with backlogging, outsourcing and limited inventory », *Computers & Industrial Engineering*, vol. 64, no 1, p. 200-210.

Coppens, Tim (2013). *Capacitated lot sizing in a dynamic make-to-order production environment with practical inventory restrictions on warehouses*, Master's thesis, Eindhoven University of Technology, 92 p.

Cunha, Artur Lovato and Maristela Oliveira Santos (2017). « Mathematical modelling and solution approaches for production planning in a chemical industry », *Pesquisa Operacional*, vol. 37, no 2, p. 311-331.

Cunha, Artur Lovato, Maristela Oliveira Santos, Reinaldo Morabito and Ana Barbosa-Póvoa (2018). « An integrated approach for production lot sizing and raw material purchasing », *European Journal of Operational Research*, vol. 269, no 3, p. 923-938.

Di Summa, Marco and Laurence A. Wolsey (2010). « Lot-sizing with stock upper bounds and fixed charges », *SIAM Journal on Discrete Mathematics*, vol. 24, no 3, p. 853-875.

Erenguc, S Selcuk and Yasemin Aksoy (1990). « A branch and bound algorithm for a single item nonconvex dynamic lot sizing problem with capacity constraints », *Computers & Operations Research*, vol. 17, no 2, p. 199-210.

Guan, Yongpei and Tieming Liu (2010). « Stochastic lot-sizing problem with inventory-bounds and constant order-capacities », *European Journal of Operational Research*, vol. 207, no 3, p. 1398-1409.

Harris, F. W. (1913). « How Many Parts to Make at Once », *Factory, The Magazine of Management* 10, 135-136, 152.

Horngren, C. T., G. Foster, S. M. Datar, M. Rajan, C. Ittner and A. A. Baldwin (2010). *Cost accounting: A managerial emphasis*, 25e éd., Toronto, Pearson Canada, coll. Issues in accounting education.

Hwang, Hark-Chin and Wilco van den Heuvel (2012). « Improved algorithms for a lot-sizing problem with inventory bounds and backlogging », *Naval Research Logistics (NRL)*, vol. 59, no 3-4, p. 244-253.

Jans, Raf and Zeger Degraeve (2007). « Meta-heuristics for dynamic lot sizing: A review and comparison of solution approaches », *European Journal of Operational Research*, vol. 177, no 3, p. 1855-1875.

Jaruphongsa, Wikrom, Sila Çetinkaya and Chung-Yee Lee (2004). « Warehouse space capacity and delivery time window considerations in dynamic lot-sizing for a simple supply chain », *International Journal of Production Economics*, vol. 92, no 2, p. 169-180.

Krarup, Jakob and Ole Bilde (1977). « Plant location, set covering and economic lot size: An o (mn)-algorithm for structured problems », in *Numerische methoden bei optimierungsaufgaben band 3*, Springer, p. 155-180.

Lin, Binshan, James Collins, Robert K Su and Logistics Management (2001). « Supply chain costing: An activity-based perspective », *International Journal of Physical Distribution*, vol. 31, no 10, p. 702-713.

Love, Stephen F (1973). « Bounded production and inventory models with piecewise concave costs », vol. 20, no 3, p. 313-318.

Melo, Rafael A. and Celso C. Ribeiro (2016). « Formulations and heuristics for the multi-item uncapacitated lot-sizing problem with inventory bounds », *International Journal of Production Research*, vol. 55, no 2, p. 576-592.

Minner, Stefan (2009). « A comparison of simple heuristics for multi-product dynamic demand lot-sizing with limited warehouse capacity », *International Journal of Production Economics*, vol. 118, no 1, p. 305-310.

Pochet, Yves and Laurence A Wolsey (2006). *Production planning by mixed integer programming*, Springer Science & Business Media.

Trigeiro, William W, L Joseph Thomas and John O McClain (1989). « Capacitated lot sizing with setup times », *Management science*, vol. 35, no 3, p. 353-366.

Van Vyve, M. and F. Ortega (2004). « Lot-sizing with fixed charges on stocks: The convex hull », *Discrete Optimization*, vol. 1, no 2, p. 189-203.

Wagner, Harvey M and Thomson M Whitin (1958). « Dynamic version of the economic lot size model », *Management Science*, vol. 5, no 1, p. 89-96.

Wolsey, Laurence A (2006). « Lot-sizing with production and delivery time windows », *Mathematical Programming*, vol. 107, no 3, p. 471-489.

Wolsey, Laurence A (2015). *Uncapacitated lot-sizing with stock upper bounds, stock fixed costs, stock overloads and backlogging: A tight formulation*, no 2015041, Université catholique de Louvain, Center for Operations Research and Econometrics (CORE).

Appendix A

Implementation of the general model

The general model for the CLSP-MSL proposed in 4.3.2 is implemented using OPL (Optimization Programming Language) as follows:

```
/*********************************************************
********************** Parameters ***********************
*********************************************************/

/*************** Indexes ***************/
int T=...;
range periods=1..T;
int I=...;
range items=1..I;
int L=...;
range locations=1..L;

/*************** Cost Structure ***************/
float f[items]=...;//production fixed
float g[locations]=...;//storage fixed
float vc[items]=...;//unit production cost
float hc[items]=...;//unit holding cost
float r[locations][locations]=...;//relocation cost

/*************** Other Parameters ***************/
float d[items][periods]=...;
float vt[items]=...;//unit production capacity usage
float P=...;//production capacity
float st[items]=...;//unit volume
float H[locations]=...;//locational storage capacity
int alpha[locations][items]=...;//item-location compatibility
int beta[items][items]=...;//item-item compatibility
float ha=...;

/*********************************************************
********************** Variables ************************
*********************************************************/
dvar float+ x[items][periods];
dvar float+ s[items][locations][periods];
dvar boolean y[items][periods];//production decision binary
dvar boolean z[locations][periods];//storage decision binary
dvar boolean w[items][locations][periods];//allocation item-
loaction binary
dvar float+ v[items][locations][locations][periods];//relocated
```

```
items
dvar float+ inflow[items][locations][periods];//inflow
production-locations
dvar float+ outflow[items][locations][periods];//outflow
locations-demand

/*************** Expresssions ***************/
dexpr float production_cost=sum(i in items, t in
periods)(x[i][t]*vc[i]);

dexpr float holding_cost=sum(i in items, l in locations, t in
periods)(s[i][l][t]*hc[i]);

dexpr float production_fixed=sum(i in items, t in
periods)(f[i]*y[i][t]);

dexpr float stock_fixed=sum(l in locations, t in
periods)(g[l]*z[l][t]);

dexpr float handling_cost=sum(i in items, l in locations, t in
periods)(inflow[i][l][t]*ha);

dexpr float handled_units=sum(i in items, l in locations, t in
periods)inflow[i][l][t];

dexpr float relocation_cost=sum(i in items, k in locations, l in
locations:k!=l, t in periods)(r[k][l]*v[i][k][l][t]);

dexpr float units_reloc=sum(i in items, k in locations, l in
locations:k!=l, t in periods)v[i][k][l][t];

dexpr float setup_count=sum(i in items, t in periods)y[i][t];

dexpr float location_count=sum(l in locations, t in
periods)z[l][t];

dexpr float total_space=sum(t in periods, l in
locations)H[l]*z[l][t];

dexpr float used_space=sum(i in items, l in locations, t in
periods)st[i]*s[i][l][t];

dexpr float total_demand=sum(i in items, t in periods)d[i][t];
```

```
/*********************************************************
********************** Objective ************************
*********************************************************/
minimize   production_cost + holding_cost +
           production_fixed + stock_fixed +
           relocation_cost + handling_cost;

/*********************************************************
********************** Constraints **********************
*********************************************************/
subject to{
ignore_production:
forall (i in items)
  vc[i]==0;

ct_balance:
forall (i in items, l in locations)
  s[i][l][1] == inflow[i][l][1]; //in period 1, inventory=inflow
forall (i in items, l in locations, t in periods:t>=2)
  s[i][l][t-1]+inflow[i][l][t]+sum(k in
locations:k!=l)v[i][k][l][t]
== s[i][l][t]+outflow[i][l][t]+sum(k in
locations:k!=l)v[i][l][k][t];

ct_fulfillment:
forall (i in items, l in locations)
  sum(l in locations)s[i][l][1]==x[i][1]-d[i][1];
forall (i in items, t in periods : t>=2)
  sum(l in locations)s[i][l][t-1] + x[i][t] == d[i][t] + sum(l in
locations)s[i][l][t];

ct_production_setup:
forall (i in items, t in periods)
    x[i][t]<=y[i][t]*minl(P/vt[i], sum(k in
periods:k>=t)d[i][k]);

ct_production_capacity:
forall (t in periods)
    sum(i in items)vt[i]*x[i][t]<=P;

ct_inventory_setup:
forall (i in items, l in locations, t in periods)
  s[i][l][t]<= w[i][l][t]*minl(H[l]/st[i], sum(k in
periods:k>=t+1)d[i][k]);
```

```
ct_inventory_capacity:
forall (l in locations, t in periods)
  sum(i in items)st[i]*s[i][l][t]<=H[l]*z[l][t];

ct_compatibility_location:
forall (i in items, l in locations, t in periods)
  w[i][l][t]<=alpha[l][i];

ct_compatibility_item:
forall (i in items, j in items:i<j, l in locations, t in periods)
  w[i][l][t]+w[j][l][t]<=beta[i][j]+1;
};
```

In our test on the symmetry, the initial symmetry breaking in the first period is realized by imposing a specific sequence among the four storage locations. The additional constraints are implemented by adding the following codes:

```
Symmetry_Breaking:
z[1][1]>=z[2][1];
z[2][1]>=z[3][1];
z[3][1]>=z[4][1];
```

In our test of disabling item relocation, the additional constraints are implemented by the following codes:

```
Disabling_Relocation:
forall (i in items, k in locations, l in locations, t in periods)
  v[i][k][l][t]==0;
```

Appendix B

Implementation of the transportation formulation

The transportation problem formulation proposed in 4.4 is implemented using OPL as follows:

```
/*********************************************************
********************** Parameters ***********************
*********************************************************/

/*************** Indexes ***************/
int T=...;
range periods=1..T;
int I=...;
range items=1..I;
int L=...;
range locations=1..L;

/*************** Cost Structure ***************/
float f[items]=...;//production fixed
float g[locations]=...;//storage fixed
float vc[items]=...;//unit production cost
float hc[items]=...;//unit holding cost
float r[locations][locations]=...;

/*************** Other Parameters ***************/
float d[items][periods]=...;
float vt[items]=...;//unit production capacity usage
float P=...;
float st[items]=...;//unit volume
float H[locations]=...;//locational storage capacity
int alpha[locations][items]=...;//item-location compatibility
int beta[items][items]=...;//item-item compatibility
float ha=...;

/*********************************************************
********************** Variables ************************
*********************************************************/
dvar float+ p[items][periods][periods];
dvar float+ s[items][locations][periods];
dvar boolean y[items][periods];//production decision binary
dvar boolean z[locations][periods];//storage decision binary
dvar boolean w[items][locations][periods];//allocation item-
loaction binary
```

```
dvar float+ v[items][locations][locations][periods];//relocated
items
dvar float+ inflow[items][locations][periods];//inflow
production-locations
dvar float+ outflow[items][locations][periods];//outflow
locations-demand

/*************** Expresssions ***************/
dexpr float production_cost=sum(i in items, t in periods, u in
periods:u<=t)(p[i][u][t]*vc[i]);

dexpr float holding_cost=sum(i in items, l in locations, t in
periods)(s[i][l][t]*hc[i]);

dexpr float production_fixed=sum(i in items, u in
periods)(f[i]*y[i][u]);

dexpr float stock_fixed=sum(l in locations, t in
periods)(g[l]*z[l][t]);

dexpr float handling_cost=sum(i in items, l in locations, u in
periods)(inflow[i][l][u]*ha);

dexpr float handled_units=sum(i in items, l in locations, u in
periods)inflow[i][l][u];

dexpr float relocation_cost=sum(i in items, k in locations, l in
locations:k!=l, t in periods)(r[k][l]*v[i][k][l][t]);

dexpr float units_reloc=sum(i in items, k in locations, l in
locations:k!=l, u in periods)v[i][k][l][u];

dexpr float setup_count=sum(i in items, u in periods)y[i][u];

dexpr float location_count=sum(l in locations, t in
periods)z[l][t];

dexpr float total_space=sum(t in periods, l in
locations)H[l]*z[l][t];

dexpr float used_space=sum(i in items, l in locations, t in
periods)st[i]*s[i][l][t];

dexpr float total_demand=sum(i in items, t in periods)d[i][t];
```

```
/*********************************************************
********************** Objective ************************
*********************************************************/
minimize   production_cost + holding_cost +
           production_fixed + stock_fixed +
           relocation_cost + handling_cost;

/*********************************************************
********************** Constraints **********************
*********************************************************/
subject to{
ignore_production:
forall (i in items)
  vc[i]==0;

forall (i in items, t in periods)
  sum(u in periods:u<=t)p[i][u][t] == d[i][t];

forall (i in items, u in periods)
  sum(t in periods:t>=u)p[i][u][t]-d[i][u] == sum(l in
locations)(inflow[i][l][u]-outflow[i][l][u]);

ct_balance:
forall (i in items, l in locations)
  s[i][l][1] == inflow[i][l][1]; //in period 1, inventory=inflow
forall (i in items, l in locations)
  outflow[i][l][1]==0;
forall (i in items, l in locations, u in periods:u>=2)
  s[i][l][u-1]+inflow[i][l][u]+sum(k in
locations:k!=l)v[i][k][l][u]
== s[i][l][u]+outflow[i][l][u]+sum(k in
locations:k!=l)v[i][l][k][u];

ct_production_setup:
forall (i in items, u in periods, t in periods: t>=u)
    p[i][u][t]<=y[i][u]*d[i][t];

ct_production_capacity:
forall (u in periods)
    sum(i in items, t in periods:t>=u)vt[i]*p[i][u][t]<=P;

ct_inventory_setup:
forall (i in items, l in locations, t in periods)
  s[i][l][t]<= w[i][l][t]*minl(H[l]/st[i], sum(u in
periods:u>=t+1)d[i][u]);
```

```
ct_inventory_capacity:
forall (l in locations, t in periods)
  sum(i in items)st[i]*s[i][l][t]<=H[l]*z[l][t];

ct_compatibility_location:
forall (i in items, l in locations, t in periods)
  w[i][l][t]<=alpha[l][i];

ct_compatibility_item:
forall (i in items, j in items:i<j, l in locations, t in periods)
  w[i][l][t]+w[j][l][t]<=beta[i][j]+1;
};
```