Data Driven Efficiency for E-Warehousing: Descriptive and Prescriptive Analytics

by

Ugur Yildiz

A thesis
presented to the University of Waterloo
in partial fulfillment of the
requirement for the degree of
Doctor of Philosophy
in
Management Sciences

Waterloo, Ontario, Canada, 2018

Examining Committee Membership

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

External Examiner:	Manish Verma Associate Professor, Dept. of Operations Management McMaster University
Supervisor(s):	Fatma Gzara Associate Professor, Dept. of Management Sciences Samir Elhedhli Professor, Dept. of Management Sciences
Internal Member:	Sibel Alumur Alev Assistant Professor, Dept. of Management Sciences
Internal Member:	Bissan Ghaddar Assistant Professor, Dept. of Management Sciences
Internal-External Member:	Adil Al-Mayah Assistant Professor, Dept. of Civil & Environmental Engineering

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Based on data provided by a warehouse logistics management company, we analyze the warehousing operation and its major processes of order picking and order consolidation. Without access to the actual layouts and process flow diagrams, we analyze the data to describe the processes in detail, and prescribe changes to improve the operation.

We investigate the characteristics of the order preparation process and the order consolidation operation. We find that products from different orders are mixed for effective picking. Similar products from different orders are picked together in containers called totes. Full totes are stored in a buffer area, and then routed to a conveyor system where products are sorted. The contents of the totes are then consolidated into orders. This order consolidation process depends on the sequence in which totes are processed and has a huge impact on the order completion time. OCP is a new problem for both the warehouse management system and the parallel machine scheduling literature. We provide mathematical formulations for the problem and devise two solution methods. The first is a simulated annealing metaheuristic, while the second is an exact branch-and-price method.

We test the solutions on both random and industry data. Simulated Annealing is found to achieve near optimal solutions within 0.01 % of optimality. For the branch-and-price approach, we use a set partitioning formulation and a column generation method where the subproblems are single machine scheduling problems that are solved using dynamic programming. We also devise a new branching rule and new dynamic programming algorithm to solve the subproblem after branching. To assess the efficiency of the proposed branch-and-price methodology, we compare against the branch-and-price approach of Chen and Powell (1999) for the parallel machine scheduling problem. We take advantage of the fact that OCP is a generalization of the parallel machine scheduling problem. The proposed, more general, branch-and-price approach achieves the same solution quality, but

takes more time.

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my supervisors, Professor Fatma Gzara and Professor Samir Elhedhli for their guidance and support throughout my graduate studies. I learned a lot from their technical expertise and professional ethics. Their dedicated guidance and generous support, in terms of time, effort and finance, were the critical factors in my ability to reach this point.

I also would like to thank my supervisory committee, Professors Sibel Alumur Alev, Bissan Ghaddar, and Adil Al-Mayah for serving on my PhD committee. I also thank Professor Manish Verma, my external examiner, for his time. My special thanks go to the industrial collaborator of this project. They shared their data and helped me understand the warehouse operations. I am thankful for the time and resources they invested in this project.

I am honoured to be a member of the Waterloo Analytics and optimization (WanOpt) lab and am extremely grateful to be surrounded by this group of bright minds and joyful spirits. I am thankful for the help and encouragement I received from my fellow lab-mates, especially Daniel Ulch for the time he dedicated to help review my thesis.

Last but not the least, I would like to thank my family: my parents and my brother for supporting me spiritually throughout writing this thesis and my life in general.

Dedication

This is dedicated to the one I love.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Warehousing operations have a significant effect on transportation efficiency and the cost of a product. This is particularly true for e-business, where customer expectations are, to a great extent, measured by the efficiency of the packaging and delivery processes. Today, manual warehousing operations are being replaced by intelligent automated systems. As these systems are taking over manual operations, data is being collected on every step an order goes through. Data often carries with it valuable information that can be used to improve operations and increase efficiency. It is with this mindset that a data analytics (DA) collaboration with a global warehousing and logistics automation company was established, leading to the current work.

Data analytics can help analyze process flow, predict anomalies, hint at possible courses of action, and validate any proposed changes. But most importantly, carefully designed DA tools can identify weaknesses in large systems where manual inspection is overshadowed by large production volumes. E-commerce warehouses are an example of such facilities. Online orders with different content and priorities have to be filled in record time to meet customer expectations. Although the demand for individual products may be low, orders

contain multiple products, typically of high variety. Not only does an e-warehouse require a large physical space, but it also requires an efficient order fulfilment process in terms of product picking, consolidating, and packaging.

Order picking has been identified as the most costly and labour-intensive activity in a warehouse; the cost of order picking is estimated to be as much as 55% of the total warehouse operating expenses (De Koster et al., 2007). To reduce the cost of order picking, several policies have been introduced, such as batch or wave picking. Batch/wave picking refers to grouping orders into batches/waves for simultaneous processing. Orders from the same batch are placed into the same tote, whereas orders from the same wave may be placed into multiple different totes. Also, order integrity is maintained in batch picking as all products from the same order have to be in the same tote. In wave picking, however, orders are consolidated at a later stage. The current work uses data from an e-commerce warehouse that uses a wave picking policy. Products from all orders within a wave are picked together regardless of order integrity. The picking process is done manually where an operator is handed a picking list and a tote, and is asked to go through the aisles of the warehouse to pick the corresponding products. After picking, totes are stored temporarily in a buffer area.

To consolidate orders and ensure their integrity, an Accumulation/Sorting (A/S) system is often used. For the warehouse under study, there is no accumulation lane. Totes, with information on the products they carry, are fed to a unit sorter through multiple induction lines. The system routes the totes to the unit sorter through the induction lines. The unit sorter is a conveyor equipped with a scanner that scans and routes products to their final packaging location, called a cubby. A cubby is similar to an accumulation lane, where products of the same order are gathered. This order consolidation process continues until the last product within an order is put. Once an order is complete, it is packed and moved

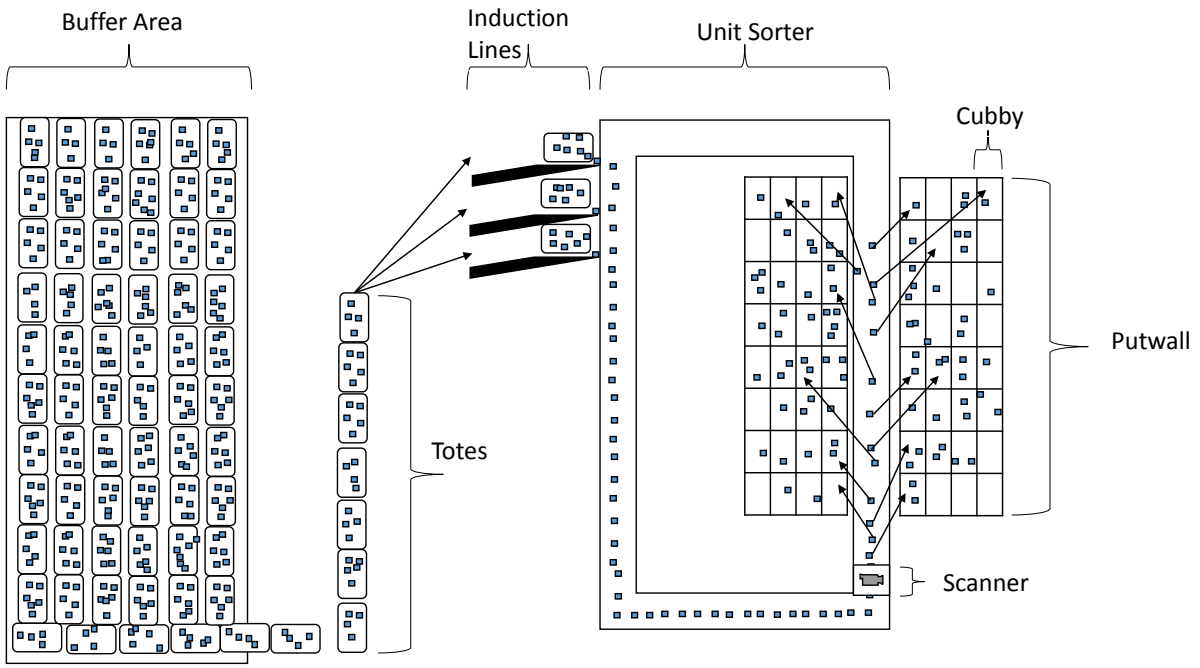


Figure 1.1: Consolidation process flow diagram deduced from the data.

to the delivery area. Figure 1.1 shows a process diagram of the consolidation process in the warehouse under study, entirely deduced from the data provided.

As an e-warehouse caters to online customer demand, the daily demand of an e-warehouse is typically higher than that of a regular warehouse. For example, the warehouse under study release more than 250,000 orders containing more than 1,200,000 products in eight days and may operate up to 24 hours a day as observed in the data. According to the data, the warehouse use a wave picking policy. The characteristics of the wave in terms of the number and variety of orders and products is important. For the warehouse under study, the average number of products and orders per wave for an eight day period (data covers eight days) is given in Figure 1.2. Figure 1.2 shows that the daily average number of

orders per wave varies between 790 and 1,405, while the daily average number of products per wave ranges from 3,955 to 7,100.

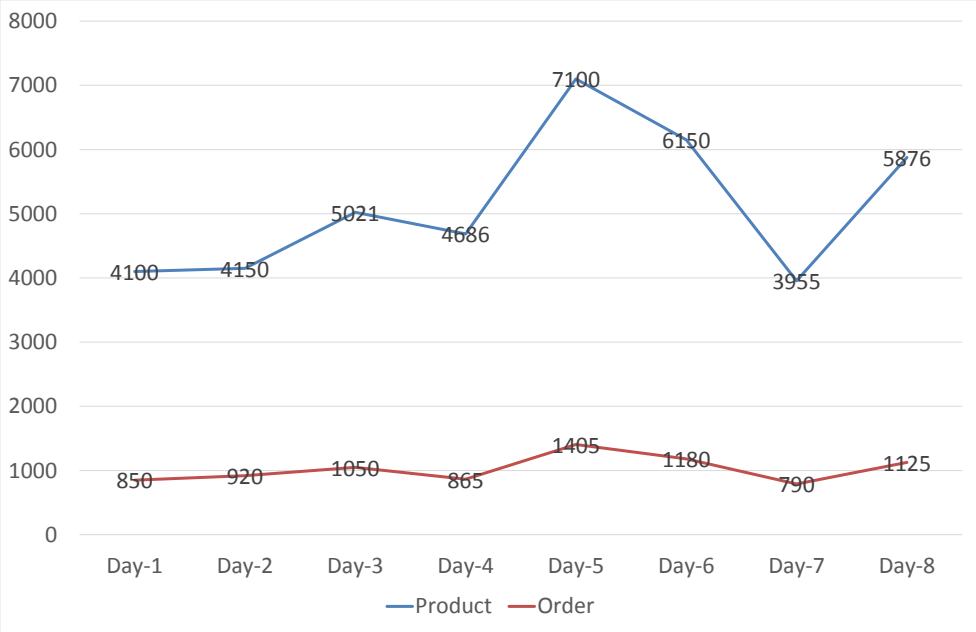


Figure 1.2: Average number of orders and products per wave during eight day.

Order characteristics affect the pick lists, hence, the contents of the totes, and, eventually, the consolidation process. It is expected that the number of different totes that an order is dispersed across is higher when the number of products in an order is higher. The number of products per order is displayed in Figure 1.3. It shows that 75 % of the orders contain at most five products, whereas only 7 % contain more than 10 products.

Finally, Figure 1.4 displays the number of waves, orders, products, and totes for each day. The number of waves ranges between 6 and 92, the number of orders between 9,835 and 74,680, the number of products between 24,600 and 387,450, and the number of totes

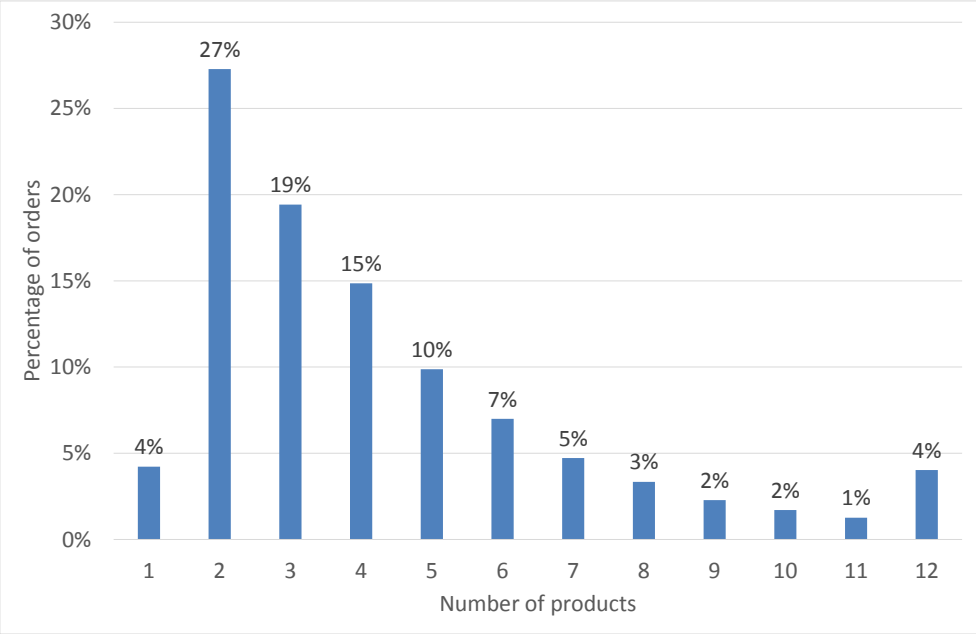


Figure 1.3: Number of products per order.

between 4,252 and 20,630, which implies that, on average, 32,814 orders containing 165,335 products are processed in 34 waves using 10,742 totes per day. The large scale of the daily operation suggests that minor improvement in the order consolidation time may translate in huge savings.

In this work, we analyze the data to understand the warehousing operation and the different steps it involves. We then identify processes that can be optimized to improve the overall order fulfilment operation. We emphasize that no prior information on the warehouse is available, including the nature of the products served, the size and layout of the warehouse, or the equipment used. The data was provided by an industrial partner who obtained the data from the automated warehousing system the warehouse operates.

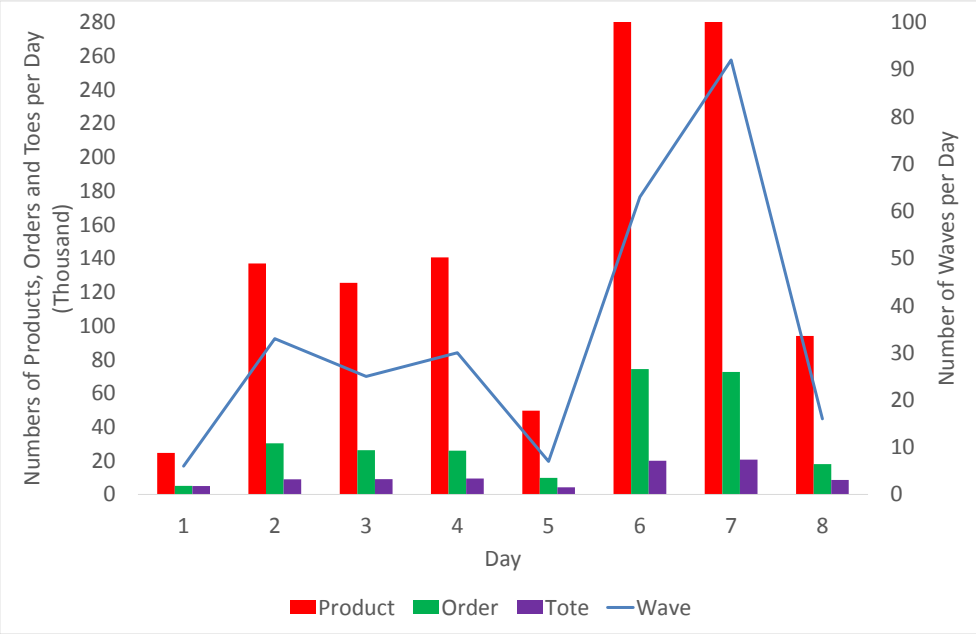


Figure 1.4: Size of daily operation.

We analyze historical data to identify the different operations in the consolidation process using descriptive analytics. Three main parts (picking, consolidation, and waiting before packaging) are identified, and the importance of each, as well as the relationships among them, are examined. We find that consolidation time, which is the completion time of an order at the unit sorter, has a significant impact on the whole process. In investigating the problem, we identify a general class of parallel machine scheduling problems that we call the Order Consolidation Problem (OCP). We define OCP to optimize the consolidation process. We present a nonlinear mixed integer programming formulation of OCP, and develop heuristic and exact solution methodologies based on simulated annealing and branch-and-price, respectively. We test our algorithm on both industry and random in-

stances. The random instances are used to assess the quality of the solutions generated, while the relatively large industry instances are used for validation. The results show that our proposed solutions improve the consolidation time of the orders significantly and, hence, the entire order preparation process.

The rest of the thesis is structured as follows: in Chapter 2, we review the literature for both warehouse management systems and parallel machine scheduling problems. In Chapter 3, we provide descriptive anlaytics based on the data provided and show that improvement of the order consolidation process would improve the whole process. In Chapter 4, we introduce the order consolidation problem (OCP), propose a solution methodology based on simulated annealing, and present improvements on the real system. We propose a branch and price algorithm and conduct an extensive computational study in Chapter 5. Finally, we conclude the thesis in Chapter 6.

Chapter 2

Literature Review

In this chapter, we review the literature on warehouse management systems and the parallel machine scheduling/order consolidation problem.

2.1 Warehouse Management Systems

Warehouse management is a well-studied research area with several research segments such as picking strategies, routing policies, operation strategies, and accumulation/sorting (A/S) systems design. Since the focus of the work is on the order consolidation process, we review the literature on A/S systems and order batching. We recommend the reviews of Gu et al. (2007) and De Koster et al. (2007) for picking policies.

The main goal of order batching is to increase efficiency by grouping orders and processing them together. A batch is a set of orders that is picked by a single picker. The problem was studied by Gademann et al. (2001), Hsieh and Huang (2011), and Grosse et al. (2014), who proposed heuristic and exact methods to solve it. The purpose is to

reduce travel times by picking the same products or the products that are stored in close proximity in the warehouse (Henn et al., 2012).

After order batching is performed, the routing is determined, i.e. the sequences in which pickers pick the required products. Ratliff and Rosenthal (1983) and Roodbergen and Koster (2001), for example, use a special case of the Traveling Salesman Problem to optimize routing in a rectangular warehouse. Most of the research is focused on developing heuristics for the routing of pickers (Hwang et al., 2004; Petersen and Aase, 2004; Theys et al., 2010). Recently, other issues, such as picker blocking and congestion, have been included in routing pickers (Pan and Wu, 2012; Hong et al., 2012; Chen et al., 2013).

Wave picking is a policy that is applied when several pickers handle a larger set of orders simultaneously (Roodbergen and Koster, 2001). The difference between batch picking and wave picking is the definition of the pick lists. An order belongs to a single pick list in batch picking, while an order can be split into multiple pick lists in wave picking. Therefore, an order accumulation/sorting process is needed following wave picking.

Order accumulation/sorting systems are used to consolidate orders when products of order are split into multiple pick lists in the picking process. An automatic A/S is usually composed of a closed-loop conveyor with automatic divert mechanisms, a scanner, and accumulation lanes where the products are diverted (Van den Berg and Zijm, 1999). In the warehouse under study, there are cubbies instead of accumulation lanes. A sensor scans products that are inducted to the unit sorter. Products corresponding to the same order are then automatically diverted into the assigned cubby.

There are few publications on A/S systems. Under the assumption of a single order-lane assignment, Bozer and Sharp (1985) examined the advantages of recirculation to avoid lane blocking in an A/S system when a shipping lane is full. Bozer and Sharp (1985) study a system that processes a relatively small number of large orders, and where each

sorting lane is dedicated to one order. Simulation is used to analyze the dependence of system throughput on factors such as the induction capacity, the number of lanes, and the length of the lanes. Considering A/S systems where multiple orders can be assigned to one lane, Bozer et al. (1988) and Johnson (1997) suggested that assigning lanes to orders at the induction step is better than any predetermined assignment rule. Meller (1997) considered a two-level order A/S system and developed an algorithm, based on decomposing a favorably structured mathematical program, to optimally assign orders to lanes based on the arrival sequence of items to the sortation system. Russell and Meller (2003) presented a prescriptive model, which examines several system parameter combinations to help decide whether or not to automate the sortation system.

2.2 Parallel Machine Scheduling Literature Review

The order consolidation problem is closely related to the parallel machine scheduling (PMS) problem. Given a set of jobs $J = \{1, 2, ..., n\}$ with positive processing times p_j and a set of identical machines $I = \{1, 2, ..., m\}$, the PMS problem seeks to find a non-preemptive schedule such that at most one job is processed on a machine at any given time. The objective is to minimize the total job completion time. The parallel machine scheduling (PMS) problem was introduced by McNaughton (1959).

Although most of the literature focuses on single order jobs, additional types, such as batch jobs and multiple order jobs (moj), have been studied in large-scale integrated circuits and semiconductor manufacturing industries. A batch job is a group of jobs that are processed simultaneously using what is aclled batch processing machin. A multiple order job groups mulitple orders and is then assigned to a batch job. Uzsoy (1995) developed several exact and heuristic solution approaches for both single and parallel machines. Mul-

tiple order jobs were first introduced by Mason et al. (2004). Mason et al. (2004), Qu and Mason (2005), Erramilli and Mason (2006), and Mason and Chen (2010) studied multiple order jobs for single machine problems, while Jampani and Mason (2008) and Liu (2010) considered parallel machines. Studies with both batch and multiple order jobs focus on how to group orders into jobs. Accordingly, researchers focused on grouping orders and scheduling them simultaneously. Split jobs, on the other hand, refers to when an order can be processed on two machines simultaneously if preemption is allowed. The PMS problem with split jobs was studied by Serafini (1996), Xing and Zhang (2000), Logendran and Subur (2004), Shim and Kim (2008), and Wang et al. (2013). Several performance criteria such as minimization of total weighted completion time (TWCT), total weighted tardiness time (TWTT), and completion time of the last job (Cmax), etc, have been used as objective functions. We focus on TWCT due to its relevance to OCP.

The literature considers three types of machines: identical, uniform, and unrelated. Identical machines have the same processing time for each job (Chan et al., 1998; Juesheng, 1998; Min and Cheng, 1999; Chen and Powell, 1999; van Den Akker et al., 1999; Radhakrishnan and Ventura, 2000; Mokotoff, 2004; Pessoa et al., 2010; Kaplan and Rabadi, 2013; Li et al., 2016; Liaw, 2016; Kowalczyk and Leus, 2017). Uniform machines have processing speeds, but the speed of the machine is independent of which job it is processing (Balakrishnan et al., 1999; Chen and Powell, 1999; Yeh et al., 2015; Ramezani et al., 2015; Li et al., 2018). Unrelated machines have different processing speeds, and the speed of the machine is dependent on which job it is processing (Liu et al., 2001; Yin et al., 2001; Weng et al., 2001; Mokotoff and Chrétienne, 2002,?; Kim et al., 2002; Anagnostopoulos and Rabadi, 2002; Liu and Wang, 2006; Logendran et al., 2007; Tavakkoli-Moghaddam et al., 2009; Vallada and Ruiz, 2011; Lin and Ying, 2015; Chen, 2015; Bülbül and Şen, 2017; Villa et al., 2018).

The order consolidation problem includes batching, multiple order jobs, and splitting. Totes, which contain products from multiple orders, are similar to multiple order jobs. For an order to be completed, multiple totes need to be processed, which is similar to the splitting property. Also, the processing time of each tote is given, which is similar to batch jobs. Boysen et al. (2018) recently defined this job type and called it as the batched order bin. Unlike this work, they focus on the single machine scheduling problem.

The OCP minimizes the total completion time (TCT) of orders. Due to the job definition, OCP is shown to be NP-Hard by reducing it to the parallel machine scheduling problem with TWCT. The latter is studied by Chan et al. (1998) who showed that under some assumptions, the linear relaxation of the set partitioning formulation is optimal. van Den Akker et al. (1999) solved the same problem using column generation. Chen and Powell (1999) proposed branch-and-price for the parallel machine scheduling problem with TWCT. Weng et al. (2001) studied unrelated parallel machines with setup times. Lin and Ying (2015), Chen (2015), and Bülbül and Şen (2017) studied the unrelated PMS problem with TWCT. Mason et al. (2004) and Mason and Chen (2010) studied multiple order jobs with TCT and TWCT for the single machine problem, whereas Jampani and Mason (2008) and Jia and Mason (2009) considered the parallel machine version.

Several exact solution approaches were presented for various PMS problems. Belouadah and Potts (1994) formulated the TWCT problem as an integer program. De et al. (1994) proposed a dynamic programming exact solution algorithm for a problem involving earliness, tardiness, and due date penalties. Uzsoy (1995) developed efficient optimal algorithms to minimize Cmax, maximum lateness (Lmax), and TWCT for single batch processing machines with incompatible job families. Uzsoy (1995) also applied some of those results to problems with parallel identical batch processing machines. Schutten and Leussink (1996) presented a branch and bound algorithm to solve the PMS problem with release dates and

family setup times to minimize Lmax. Serafini (1996) studied job splitting on identical parallel machines in a textile industry to minimize the maximum weighted tardiness. It is shown that minimizing maximum weighted tardiness can be done in polynomial time. Both van Den Akker et al. (1999) and Chen and Powell (1999) proposed and analyzed the column generation approach to the set partitioning formulation of the TWCT problem. Chen and Powell (1999) considered a class of problems of scheduling uniform or unrelated parallel machines with an objective of minimizing an additive criterion. They proposed a decomposition approach to solve this problem exactly. Balakrishnan et al. (1999) proposed a mixed integer programming (MIP) formulation for the problem of early/tardy scheduling with sequence-dependent setups on uniform parallel machines. Mokotoff and Chrétienne (2002) and Mokotoff (2004) developed exact cutting plane algorithms for parallel machines with the criteria of Cmax.

For split jobs, Shim and Kim (2008) studied identical parallel machines with the objective of minimizing total tardiness and developed a branch and bound algorithm using several dominance properties and lower bounds for the problem. Liaw (2016) studied the problem of scheduling preemptive jobs on identical parallel machines to minimize total tardiness and developed a branch-and bound algorithm using a lower bound scheme and a heuristic algorithm. Bülbül and Şen (2017) proved that relaxing the problem of TWCT for unrelated parallel machines naturally provides a non-preemptive solution and formulated the problem as a mixed integer linear program. They also developed an exact Benders decomposition-based algorithm for solving this formulation. Kowalczyk and Leus (2017) considered identical machines with an undirected conflict graph which imposes whether or not two jobs can be processed at the same machine. For the objective of Cmax, they presented an exact branch and price algorithm.

Heuristic approaches have also been presented. For identical parallel machines, Juesh-

eng (1998) and Min and Cheng (1999) proposed a genetic algorithm for the minimization of Cmax. Radhakrishnan and Ventura (2000) used simulated annealing for the identical parallel machines to minimize the sum of the absolute deviations of job completion times from their corresponding due dates. Xing and Zhang (2000) developed a heuristic to solve the identical parallel machine problem with job splitting to minimize Cmax. Weng et al. (2001) studied unrelated parallel machines for the TWCT criteria with setup times, proposed seven heuristic algorithms, and tested them using simulation. Kim et al. (2002) devised a simulated annealing algorithm for the unrelated parallel machines problem with sequence dependent setup times. Anagnostopoulos and Rabadi (2002) studied the same problem for the Cmax criteria and also implemented simulated annealing. Logendran and Subur (2004) investigated an unrelated parallel machine problem with job splitting, where jobs are split beforehand, and a tabu search based heuristic algorithm was developed for minimizing the TWT objective function. Logendran et al. (2007) studied unrelated parallel machines with sequence-dependent setup times for the TWT criteria and proposed a tabu search algorithm.

For the parallel machine multiple order job scheduling problem of minimizing the TWCT, Jampani and Mason (2008) developed a column generation based heuristic for the problem with order ready times. Jia and Mason (2009) also developed heuristic algorithms containing three parts: order sequencing, job-to-machine assignment and sequencing, and order-to-job assignment. Tavakkoli-Moghaddam et al. (2009) and Vallada and Ruiz (2011) studied unrelated machines and proposed a genetic algorithm. Fanjul-Peyro and Ruiz (2010) proposed a set of simple iterated greedy local search based metaheuristics for the unrelated parallel machine problem for the objective of Cmax minimization. To minimize total tardiness, Lee et al. (2013) studied the problem of unrelated PMS with machine and sequence-dependent setup times and developed a tabu search algorithm. Kaplan

and Rabadi (2013) defined fighter jets refuelling as an identical parallel machine problem with the objective of TWT and solved it using simulated annealing. Lin and Ying (2015) used simulated annealing to solve the unrelated machines problem with multiple criteria. Chen (2015) developed three heuristics for the unrelated parallel machines problem with sequence-dependent setup time for the objective of TWCT. Li et al. (2016) studied identical machines with Cmax and TWCT criteria under a green environment. The green environment is defined through a constraint that limits the total machine cost by a given threshold. Several heuristics were developed for both problems. Villa et al. (2018) developed several heuristics for the unrelated parallel machine scheduling problem with one scarce additional resource to minimize Cmax.

PMS problems have been studied for several objectives, machine types, job types, and constraints. However, with the technological developments in production environments, new job, machine, or performance criteria may be required to tackle recent problems.

Table 2.1 provides a summary of the papers reviewed in terms of machine type (Identical (P), Uniform (Q), Unrelated (R)), job type (Single, Batch, Split, Multiple Order Jobs (MOJ)), and solution methodology (Exact, Heuristic). According to the table, most of the work focuses on single order jobs, with only two studies on multiple order jobs. Apart from Boysen et al. (2018), which is the closest to this work, there is very limited research on multiple orders-multiple jobs.

2.3 Conclusions

In this chapter, we provide literature review on order picking and, accumulation and sortation systems. We also present an extensive literature review on parallel machine scheduling problems. We review parallel machine scheduling problems under four categories: machine

Table 2.1: Classification of machine schedulling literature

	Machine Type			Job Type				Solution		
Paper	**P**	**Q**	**R**	**Single**	**Batch**	**Split**	**MOJ**	**Exact**	**Heur.**	**Criteria**
Belouadah and Potts (1994)	✓			✓				✓		TWCT
De et al. (1994)	✓			✓				✓		Other
Schutten and Leussink (1996)	✓			✓				✓		Lmax
Serafini (1996)	✓					✓		✓		Maximum WT
van Den Akker et al. (1999)	✓			✓				✓		TWCT
van Den Akker et al. (1999)	✓			✓				✓		TWT
Chen and Powell (1999)	✓	✓	✓	✓				✓		TWCT
Chen and Powell (1999)	✓	✓	✓	✓				✓		TWU
Lin and Jeng (2004)	✓				✓			✓		Lmax
Lin and Jeng (2004)	✓				✓			✓		TU
Balakrishnan et al. (1999)		✓		✓				✓		TT+ TE
Mokotoff and Chrétienne (2002)			✓	✓				✓		Cmax
Mokotoff (2004)	✓			✓				✓		Cmax
Shim and Kim (2008)	✓					✓		✓		TT
Liaw (2016)	✓			✓				✓		TT
Bülbül and Şen (2017)			✓					✓		TWCT
Kowalczyk and Leus (2017)	✓							✓		Cmax
Min and Cheng (1999)	✓			✓					✓	Cmax
Xing and Zhang (2000)	✓	✓	✓			✓			✓	Cmax
Weng et al. (2001)			✓	✓					✓	TWCT
Kim et al. (2002)			✓	✓					✓	TT
Anagnostopoulos and Rabadi (2002)			✓	✓					✓	Cmax
Logendran and Subur (2004)			✓			✓			✓	TWT
Mönch et al. (2005)	✓				✓				✓	TWT
Logendran et al. (2007)			✓	✓					✓	TWT
Jampani and Mason (2008)	✓						✓		✓	TWCT
Jia and Mason (2009)	✓						✓		✓	TWCT
Tavakkoli-Moghaddam et al. (2009)			✓	✓					✓	TT+TC
Fanjul-Peyro and Ruiz (2010)			✓	✓					✓	Cmax
Vallada and Ruiz (2011)			✓	✓					✓	Cmax
Lee et al. (2013)			✓	✓					✓	TT
Kaplan and Rabadi (2013)	✓			✓					✓	TWT
Chen (2015)			✓	✓					✓	TWCT
Villa et al. (2018)			✓	✓					✓	Cmax

type, job type, performance criteria, and solutions methods. The many orders-many jobs characteristic of OCP makes it a unique problem in the literature. It generalizes the PMS problem and to the best of our knowledge, there is only one related paper.

Before we formally define OCP, we report on the descriptive data analysis on which OCP is identified in the next chapter.

Chapter 3

Descriptive Analytics of the Order Preperation Process

In this chapter, we analyze a data set which contains detailed information on the unit sorter operation, where order consolidation is performed. We define the steps of order preparation process, then show the importance of the order consolidation process by analyzing the contribution of each step to the overall order preparation time.

The wave operation is executed as follows. A set of orders are grouped into a wave to be processed simultaneously. Then pick lists are formed so that products from different orders are grouped to be picked by the same picker and placed in a tote. Once all products in a pick list are placed in a tote, the latter is placed in a buffer area awaiting the start of consolidation. Consolidation starts when all totes of a wave reach the buffer. Totes are diverted to induction lines where their content is inducted on the unit sorter. The unit sorter uses a conveyor system to send each product to the cubby assigned to the respective order. Recall that each product belongs to an order. Once all the products of an order are in the cubby, the order may be packed. The wave operation is completed

all orders are packed. The wave operation timeline is illustrated in Figure **??**. In this complex operation, order preparation process is performed. The order preparation process timeline is illustrated in Figure **??**. After an order is assigned to a wave, its products are assigned to pick lists, which are then picked into totes. Totes wait in the buffer area until the consolidation process starts. The consolidation of an order starts when the first product from the wave is inducted to the unit sorter. The duration between the order release time and the beginning of the sortation operation refers to the picking duration. The two main operations in consolidation are putting and packing. The putting duration is defined as the time between induction of the first and the last products from the same order. Consequently, a waiting time for an order occurs between the start time of the wave and the start time of the order. Therefore, consolidation time is the summation of the waiting time in the buffer and putting time. Also, the idle time, which is defined as the time spent waiting for packing, may occur after all products are gathered in the same cubby and the order is ready to be packed. Order completion time is the time between the wave consolidation start time and the packing start time, which excludes picking from the order preparation time.

We investigate the average order completion times to understand the importance of the different processes. First, we clean the data by eliminating waves that are processed over multiple days to remove the samples with huge break times. Second, we eliminate the waves whose processes are stopped due to lunch or dinner breaks. The reason for eliminating these waves instead of removing the break times is the uncertainty of working environments before and after the breaks. Working environments may refer to the number of working operators and induction lines in use. Thus, for the sake of reliability of the analysis, we eliminate the waves with long breaks and only analyze the waves with no significant breaks.

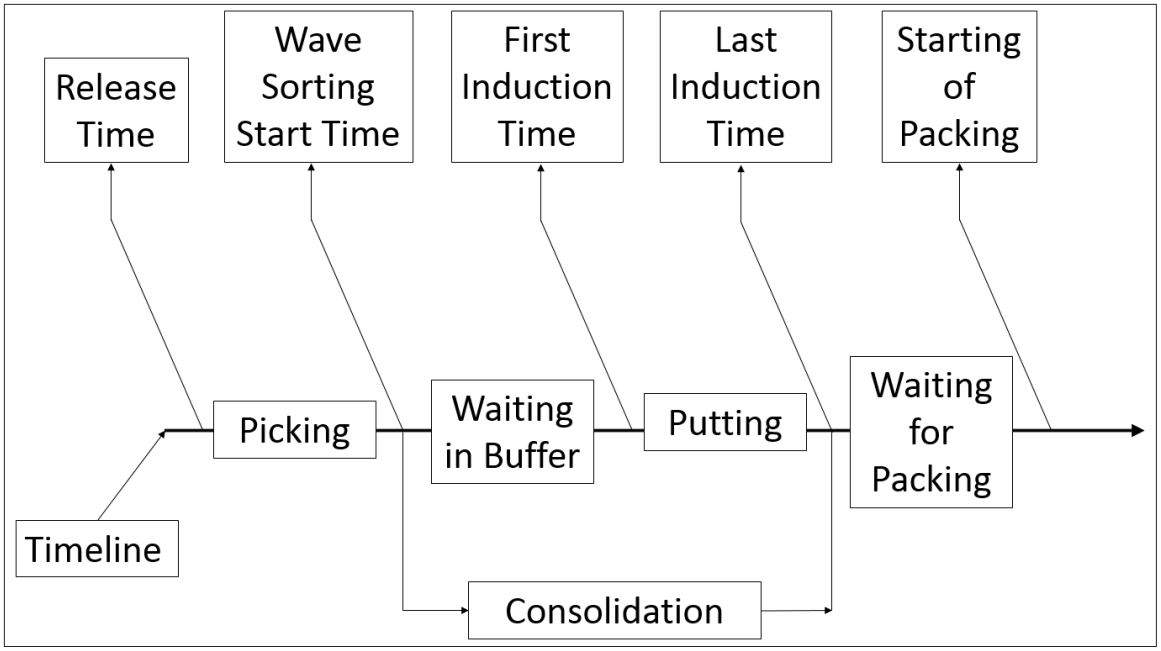


Figure 3.1: Definitions of the durations and the times.

We calculate the process times for 17 waves over eight days. Figure 3.2 shows process times for picking, waiting in buffer, putting, and waiting for packing. On average, an order spends 61 minutes in picking, 26 minutes in the buffer area, 30 minutes in putting, and 19 minutes waiting for packing. Given the lack of data on the picking operation, we investigate the consolidation operation further. Since consolidation covers the buffer waiting and putting time of an order, we calculate consolidation time as the summation of those two durations. For the same waves from Figure 3.2, we present picking, consolidation, and waiting for packing times in Figure 3.3. The consolidation operation is the second largest part of order preparation with an average time of 56 minutes.

In Figures 3.4, the waves are sorted by descending order according to their completion time. We observe that consolidation time contributes the longest part of completion time.

To achieve an improved consolidation process, we propose to determine an optimal sequence

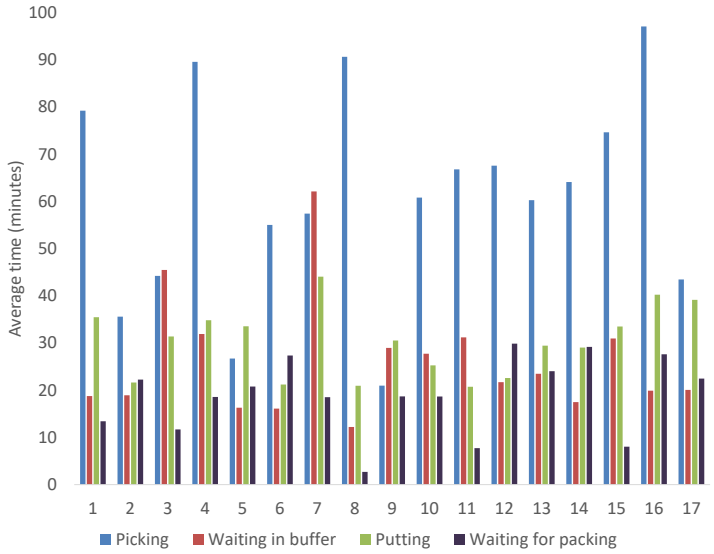


Figure 3.2: Average picking, waiting in buffer, putting times and waiting for packing of the orders for 17 waves.

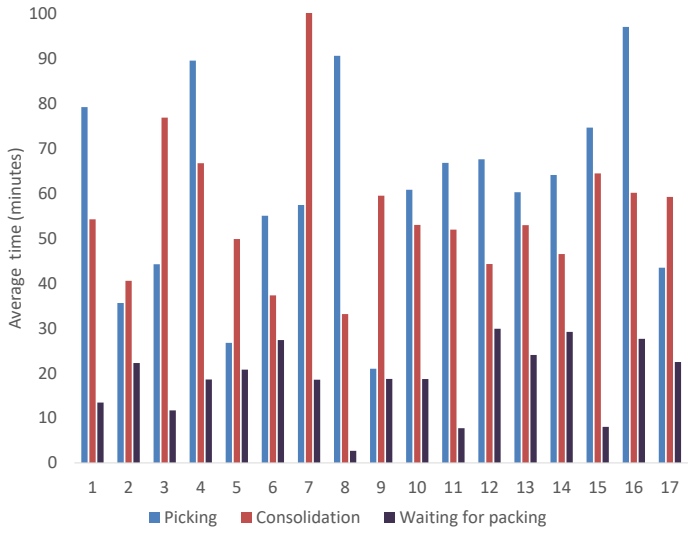


Figure 3.3: Average picking, consolidation and waiting for packing of the orders for 17 waves.

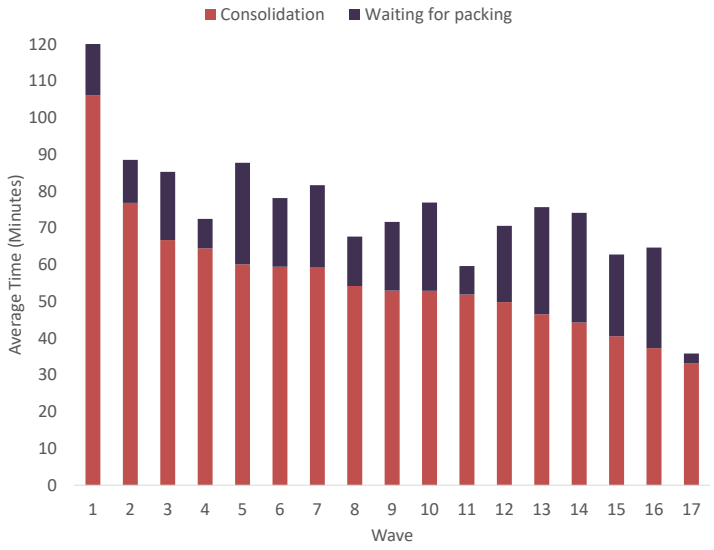
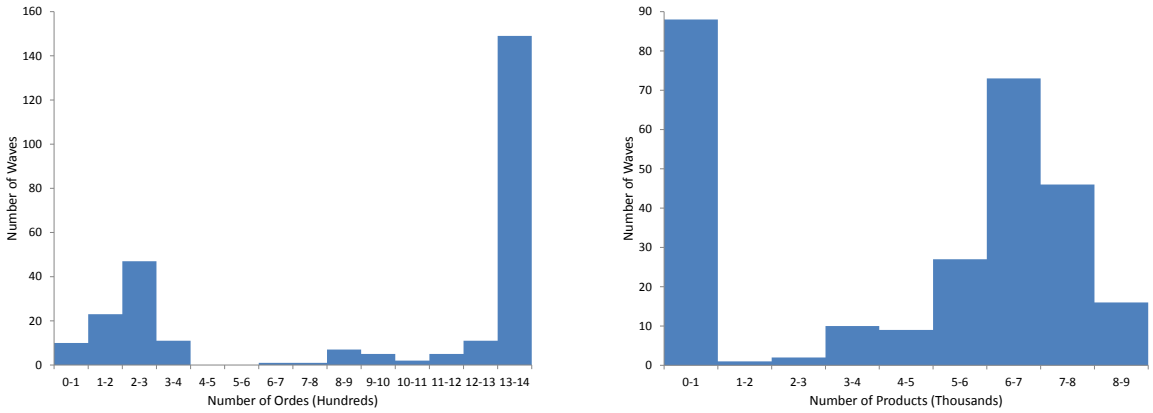


Figure 3.4: Total average completion times of the orders for 17 waves.

of the totes, i.e, how the totes are diverted from the buffer area to the putwall area. The sequence determines the last induction time of the orders and, hence, the corresponding consolidation time.

Next we calculate statistics on the total number of orders, the total number of products, the average consolidation time per order, and the average waiting time for packing per order. For this analysis, we include the 273 waves and eliminate breaks longer than one minute. Figures 3.5(a) and 3.5(b) depict the distribution of the order and product quantities among waves, respectively. Figure 3.5(a) reveals that more than 50 % of the waves are from the group of interval (1300-1400). Also, around 30 % of the waves are from the group of interval (0-400). Figure 3.5(b) provides similar features. Around 30 % of the waves are from the group of interval (0-3000), whereas around 50 % of them are from the group of interval (5000-8000). Both findings suggest that there are two types of waves: small-sized and large-sized. The distributions of the time related variables, average

consolidation time per order, and average waiting for packing time per order, are analyzed and illustrated in Figures 3.6(a) and 3.6(b), respectively. Both Figures 3.6(a) and 3.6(b) show that the average consolidation time of orders is significantly greater than the average waiting for packing time.



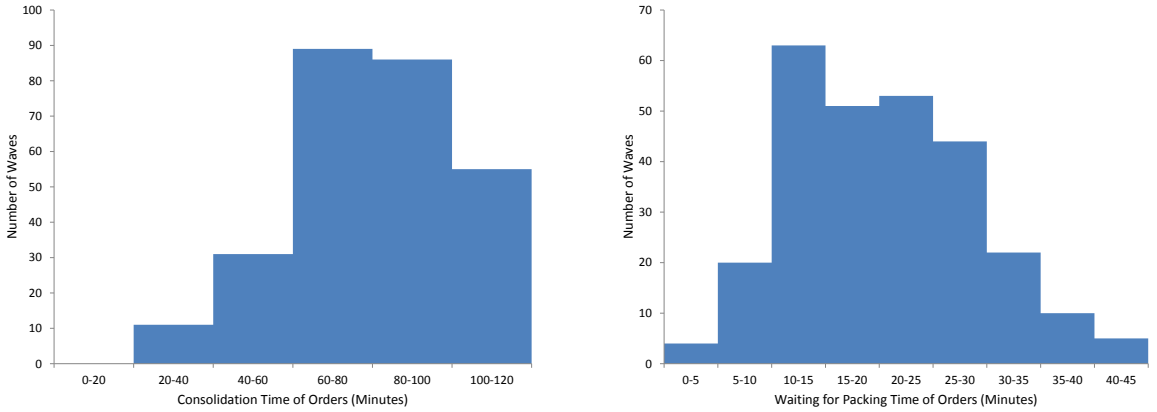
(a) Distribution of number of orders per wave.

(b) Distribution of number of products per wave.

Figure 3.5: Distributions of order size related variables.

3.1 Conclusions

In this chapter, we perform data analysis to gain an understanding of the problem under study. Based on data, we describe the current status wave operation. We define the steps of the wave operation and order preparation process and analyze their importance. On average, an order spends 56 minutes in consolidation step and 19 minutes in waiting for the packing step. Therefore, we conclude this chapter by suggesting to determine the tote sequence in which they are diverted from the buffer area to the unit sorter area to improve the order consolidation process. In the next chapter, we define the order



(a) Distribution of average consolidation time per wave. (b) Distribution of average waiting for packing time per wave.

Figure 3.6: Distributions of time related variables.

consolidation process and develop a mathematical formulation. We propose a simulated annealing metaheuristic and validate it using industry instances.

Chapter 4

Optimizing the Order Consolidation Process

The consolidation time of an order, i.e. the time from the beginning of the wave consolidation operation until the last product in the order is inducted, depends on the number of totes that carry the order's products and their induction sequence. Totes are emptied at the induction line according to the sequence in which they are released from the buffer area. An order seizes a cubby from the time that the first product from the order is inducted to the time that the last product of the order is inducted. Our calculation of the putting time for an order is tote-based rather than product-based. A tote-based calculation considers a tote as a whole and equates the induction time of all products from the tote to the induction time of the last product inducted. The product-based calculation requires the actual induction time of items; therefore, we would need to know the sequence in which items are inducted. However, there is no such sequence as operators pick products and induct them to the unit sorter arbitrarily.

In Table 4.1, the total consolidation time for five waves is compared for product versus

tote based induction times.

Table 4.1: Total consolidation time comparison.

Wave	Product-based calculation	Tote-based calculation	% GAP
1	1136034	1171015	3.08
2	506281	544820	7.61
3	1354862	1390378	2.62
4	660476	689239	4.35
5	420242	446133	6.16

According to Table 4.1, the difference between both calculations is a maximum 7.61%. Given that it is not possible to know or enforce the order of emptying products from totes, it is reasonable to use a tote-based calculation. It is both practical and acceptable.

Reducing the average consolidation time also leads to the efficient usage of resources. One of the critical resources in the warehouse is the putwall. Each wave is assigned to a putwall which has a determined number of cubbies (1309 for the warehouse under study). Figure 4.1 highlights the relation between average cubby usage and average consolidation time. It is clear that there is almost a linear relationship.

As a solution to reduce order consolidation times, we introduce an optimization problem to determine the sequence in which totes are inducted/emptied. We call such a problem, the order consolidation problem (OCP). The order consolidation problem is a generalization of the parallel machine scheduling problem as it introduces a third element besides jobs and machines. In OCP, there are totes (jobs), induction lines (machines), and orders. Orders are similar to split jobs and totes are similar to batches of split jobs. To finish processing an order, all of its parts from different batches have to be processed.

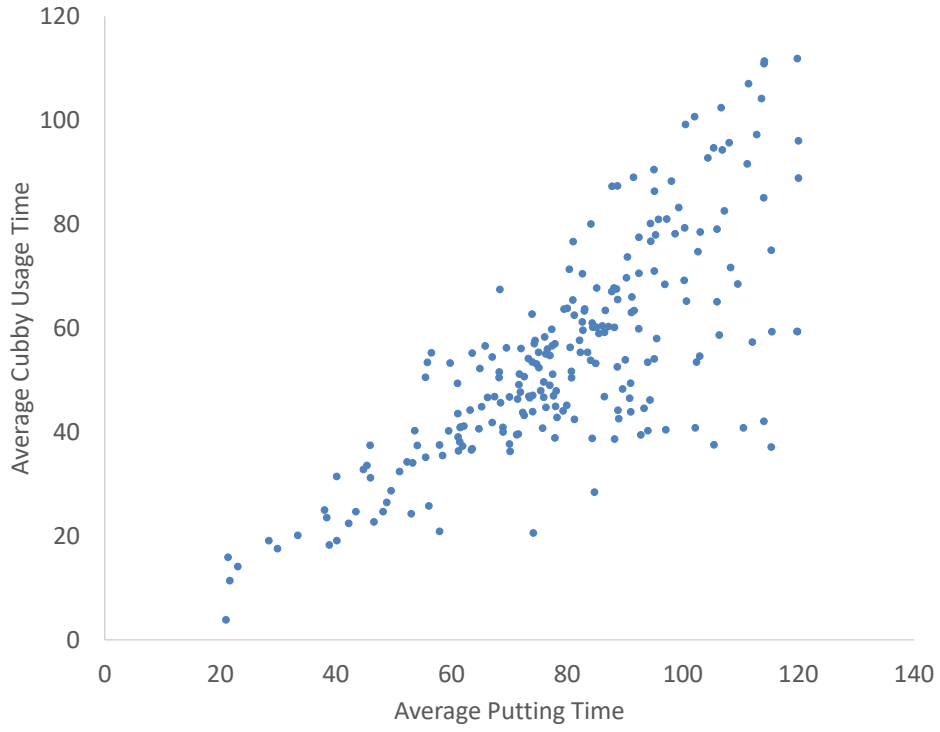


Figure 4.1: Relation between consolidation time and cubby usage.

4.1 Mathematical Formulation

We use indices $k \in K$ for orders and $i, j \in J$ for totes. Once an operator has a tote, he/she starts emptying the products one at a time at one of m induction lines. The processing time of tote j, i.e., the time it takes to empty it, is denoted by p_j. We define two sets of continuous decision variables, the completion time of order k and of tote j, denoted by CO_k for order $k \in K$ and C_j for tote $j \in J$, respectively. We also define assignment variable x_{ij} which takes value 1 if tote $i \in J$ precedes tote $j \in J$. Let us introduce the additional sets J_k, J_0, and J_{n+1}. J_k is a subset of J containing totes with products belonging to order $k \in K$. J_0 and J_{n+1} are sets with dummy node 0 and dummy node $n+1$, respectively.

The problem is formulated as nonlinear mixed integer program (NMIP):

$$\text{[NMIP]: min} \sum_{k \in K} CO_k \tag{4.1}$$

$$\text{s.t.} \sum_{i \in J_{n+1}} x_{ji} = 1 \qquad j \in J \tag{4.2}$$

$$\sum_{j \in J} x_{0j} = m \tag{4.3}$$

$$\sum_{i \in J_0} x_{ij} - \sum_{i \in J_{n+1}} x_{ji} = 0 \qquad j \in J \tag{4.4}$$

$$C_j = p_j x_{0j} + \sum_{i \in J} (C_i + p_j) x_{ij} \qquad j \in J \tag{4.5}$$

$$CO_k \geq C_j \qquad j \in J_k, k \in K \tag{4.6}$$

$$C_j \geq 0 \qquad j \in J \tag{4.7}$$

$$CO_k \geq 0 \qquad k \in K \tag{4.8}$$

$$x_{ij} \in \{0, 1\} \qquad i \in J_0, j \in J_{n+1} \tag{4.9}$$

The objective function (4.1) minimizes the total completion time of orders. Constraints (4.2) ensure that each tote precedes another tote. Since multiple totes cannot be processed simultaneously in the same line, each tote can be followed by exactly one tote, unless it is the last tote in the line. Constraint (4.3) states that there are m induction lines and m sequences of totes are formed. Constraint (4.3) assigns m totes to the dummy tote. Constraints (4.4) are the flow balance constraints to ensure that the precedence assignments form sequences. Constraints (4.5) calculate the completion time of totes.

Altough nonlinear, constraints (4.5) may be linearized as:

$$C_j \geq p_j x_{0j} \qquad j \in J \tag{4.10}$$

$$C_j \geq (C_i + p_j) - M(1 - x_{ij}) \qquad i, j \in J \tag{4.11}$$

where M is a big number. Constraints (4.6) calculate the order completion times. The rest of the constraints are the domain constraints on the decision variables.

Let us consider the special case where each order is allocated to a single tote. In this case, the OCP reduces to the machine scheduling problem with weighted completion time. Since each order corresponds to a single tote, the variable CO_k may be eliminated, and the objective function reduces to:

$$\min \sum_{k \in K} \sum_{j \in J_k} C_j \tag{4.12}$$

Note that the weighted completion time of a tote is the sum of the completion time of the orders in it. Therefore, the weights correspond to the number of of orders in a tote, and the problem reduces to the parallel machine scheduling problem with weighted completion time minimization, which is known to be a NP-hard. Hence, the OCP is also NP-hard.

Since OCP is an NP-hard problem and the size of the industry instances are be too large, we propose a simulated annealing metaheuristic approach to obtain near optimal solutions in reasonable time (in 2 minutes).

4.2 Solving OCP using Simulated Annealing

Simulated Annealing (SA) is a metaheuristic approach that draws an analogy with the physical annealing of solids to solve optimization problems (Metropolis et al., 1953). SA searches for good solutions using a probabilistic search approach. It starts with an initial solution denoted by x, with cost Z_x. A neighbour y of this initial solution is generated with cost Z_y. The change in cost $\triangle = Z_y - Z_x$ is computed. If $\triangle < 0$, then the current solution x is replaced by its neighbour y; otherwise, y replaces x with a non-zero probability that decreases gradually as the search proceeds. The acceptance function is $exp(\triangle/T)$, where T is the control parameter, which corresponds to the temperature in the analogous physical annealing process (Kirkpatrick et al., 1983).

We represent solutions of OCP as a single vector which contains all totes. We obtain machine schedules by assigning totes from this vector to the machines one by one according machine availability. We choose this representation to avoid obtaining solutions that do not satisfy the rule that a job is assigned to the first availoable machine. We refer to such solutions as infeasible. Infeasibility may only occur while assigning a tote to a machine such that the tote waits for a busy machine while there is an available one. Whenever a machine is available, it must be used for the next job. In the case where each machine schedule is defined by a vector, a simple insertion or swap operation may lead to a sollution that does not satisfy this rule and hence is infeasible. When we define the solution as a single vector of totes and obtain the machine schedules based on machine availibilitty, we do not need to worry about feasibility, because every neighbour solution we obtain is feasible. The SA algorithm is initialized using a list-based heuristic that we present next, in which a neighbour formed using either a swap or an insertion operation.

4.2.1 Initial Solution and Neighbourhood Definition

To begin the SA algorithm, an initial feasible solution is required. One approach to generate this initial solution is to randomly generate a list including every tote. This approach is easy to implement and requires no prior knowledge of the problem structure; however, it could yield solutions which are far from optimal. This approach could, in turn, lead to poor performance of SA algorithm (Radhakrishnan and Ventura, 2000). In this work, we propose a list-based heuristic which forms a list of totes based on a given criterion, then uses the list to process the totes whenever an induction line becomes available. Let v_j denote the number of distinct orders in tote $j \in J$. We sort the totes in descending order of $\frac{v_j}{p_j}$, then use it as a tote sequence for the consolidation process.

A neighbour solution is found through either a swap or an insertion. We randomly choose one of the neighbourhoods and generate the neighbour solution. Regardless of the neighbourhood we select, we randomly generate two numbers (indices of two totes from the current solution) from the interval [1,..., n] and randomly select whether a swap or an insertion is performed. If an insertion operation is selected, then we insert the second tote in the location of the first tote. If a swap operation is selected, then we swap the locations of those two totes. In the following section, we explain how the parameters of the SA algorithm are set.

4.2.2 Parameter Calibration and Main Steps

In theory, SA converges to a globally optimal solution with probability 1 if all theoretical conditions are satisfied (Kirkpatrick et al., 1983). However, the conditions for asymptotic convergence, such as the cooling method and stopping criterion, may not be met in practice (Kirkpatrick et al., 1983). Therefore, it is often critical to adjust the values of

parameters, such as initial temperature, cooling schedule, and stopping criteria based on problem characteristics (Kim et al., 2002). In addition to those parameters, we also have another parameter that defines how a neighbourhood is picked. Let ω be the probability of selecting a neighbour using swapping.

We design an experiment to determine the parameters of the SA algorithm. We choose three values for the temperature parameter T: $5n$, $10n$ and $15n$. The temperature cooling formula is $(1-q)T$, where: $q = 0.01$, $q = 0.001$ and $q = 0.0001$. We use the standard setting of a problem with 4 lines, 50 totes, 100 orders and select processing times of totes from the interval $(10, 40)$. For this problem setting, we generate 200 random instances and solve them using the SA algorithm given in Algorithm 1 for each parameter combination presented in Table 4.2.

Table 4.2: Values of SA algorithm parameters.

Parameter	Value-1	Value-2	Value-3	Value-4	Value-5
ω	0	0.25	0.5	0.75	1
T	$5n$	$10n$	$15n$	-	-
q	0.01	0.001	0.0001	-	-

Algorithm 1 first generates an initial solution using the list-based heuristic. Then, one of the neighbours is randomly selected using parameter ω and a candidate solution is generated. If the candidate solution is better than the current solution, the search moves to the neighbour solution and the best solution is updated. When a worse solution is generated, the search moves to the neighbour solution with probability $Exp(\frac{Cost(S_{current})-Cost(S_{neighbour})}{T})$. Once this is done, T is updated as $(1-q)T$ and a new iteration is performed. The search continues until T drops below a predetermined threshold of 0.01.

The averages of the best solutions found for each parameter setting are given in Table 4.3. The best parameter values according to Table 4.3 are $15n$, 0.0001, and 1 for T, q and ω,

Algorithm 1 Simulated Annealing Algorithm

Input: $Instance$, T, q, w
Output: S_{best}
$S_{current} \leftarrow ListHeuristic(Instance)$
$S_{best} \leftarrow S_{current}$
While ($T \geq 0.01$)
  $S_{neighbour} \leftarrow GetNeighbour(S_{current}, w)$
  $T \leftarrow T(1-q)$
  If $Cost(S_{neighbour}) \leq Cost(S_{current})$
    $S_{current} \leftarrow S_{neighbour}$
    If $Cost(S_{neighbour}) \leq Cost(S_{best})$
      $S_{best} \leftarrow S_{neighbour}$
    End
  Else If $Exp(\frac{Cost(S_{current})-Cost(S_{neighbour})}{T}) \geq Rand(0,1)$
    $S_{current} \leftarrow S_{neighbour}$
  End
end While
Return (S_{best})

respectively. However, the difference between the best and the worst parameter settings is only 0.3 %. Therefore, we continue analyzing the results. We first find the optimal solution for each instance under all parameter settings. We then average the optimal objectives for each value of ω for each instance. The best ω value for a given instance is the one which resulted in the best average of the optimal objectives; these ω values are shown in Figure 4.2. As shown, the best ω values are $\omega = 1$ for 149, $\omega = 0.75$ for 43 and $\omega = 0.5$ for 8 instances. The same analysis is applied to the other parameters. Figure 4.3 displays the results for T. The best solutions are obtained at $T = 5n$, $T = 10n$, and $T = 15n$ for 74, 54 and 72 instances, respectively.

Similarly, Figure 4.4 displays the same information for parameter q. The best solutions are obtained in 81, 66 and 53 instances for $q = 0.001$, $q = 0.01$ and $q = 0.0001$, respectively. This shows that no value dominates the others.

Table 4.3: Average results for each parameter setting.

T	q	$\omega = 0$	$\omega = 0.25$	$\omega = 0.5$	$\omega = 0.75$	$\omega = 1$
$5n$	0.0001	14532.19	14509.93	14502.13	14496.37	14490.64
$5n$	0.001	14531.43	14510.53	14501.34	14495.12	14490.68
$5n$	0.01	14530.74	14509.6	14501.57	14495.92	14490.7
$10n$	0.0001	14535.29	14512.07	14502.4	14497.33	14490.59
$10n$	0.001	14530.31	14511	14503.46	14496.11	14491.53
$10n$	0.01	14531.89	14513.28	14501.47	14496.91	14491.51
$15n$	0.0001	14531.56	14509.43	14502.88	14496.42	14490.54
$15n$	0.001	14528.72	14509.6	14503.12	14495.04	14491.03
$15n$	0.01	14530.01	14510.31	14502.59	14494.05	14491.45

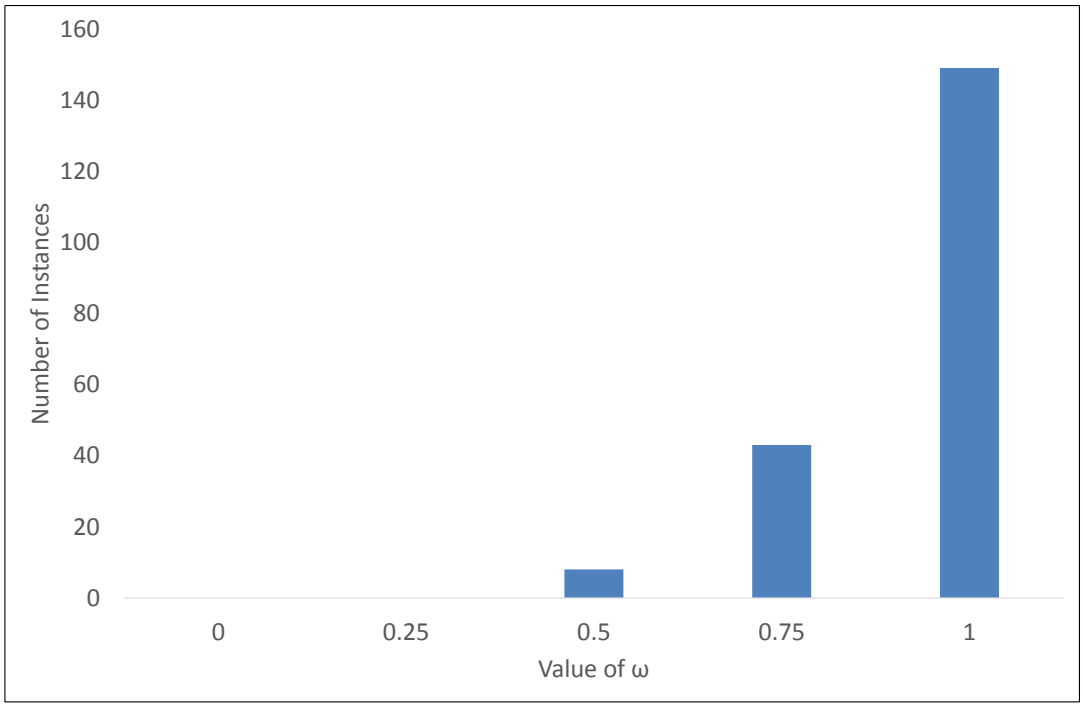


Figure 4.2: Best average performance for ω.

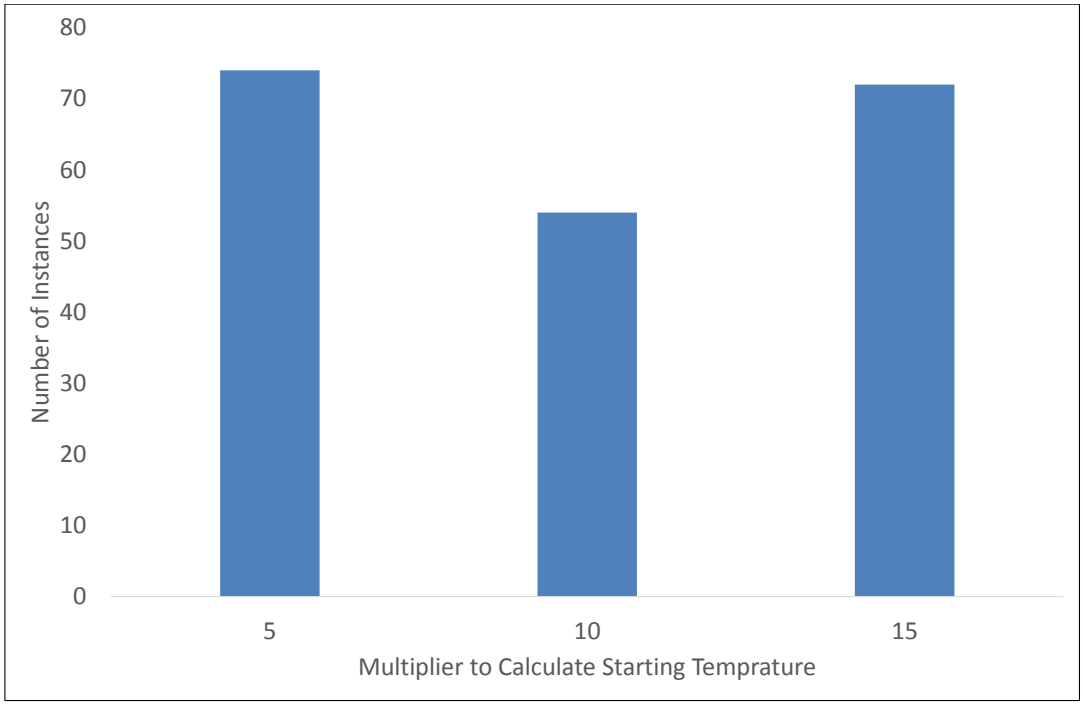


Figure 4.3: Best average performance for T.

Based on these findings, we set the values of q and T to 0.001 and $5n$, respectively. For the neighbour selection, although the swap neighbourhood dominates insertion most of the times, we set ω to 0.85, in between the best and the second best values.

4.2.3 Computational Study

We begin this section by explaining how the industry instances are extracted from the database provided by the industry partner. We then validate the results on the industry instances and compare to the actual consolidation times observed in the warehouse. Lastly,

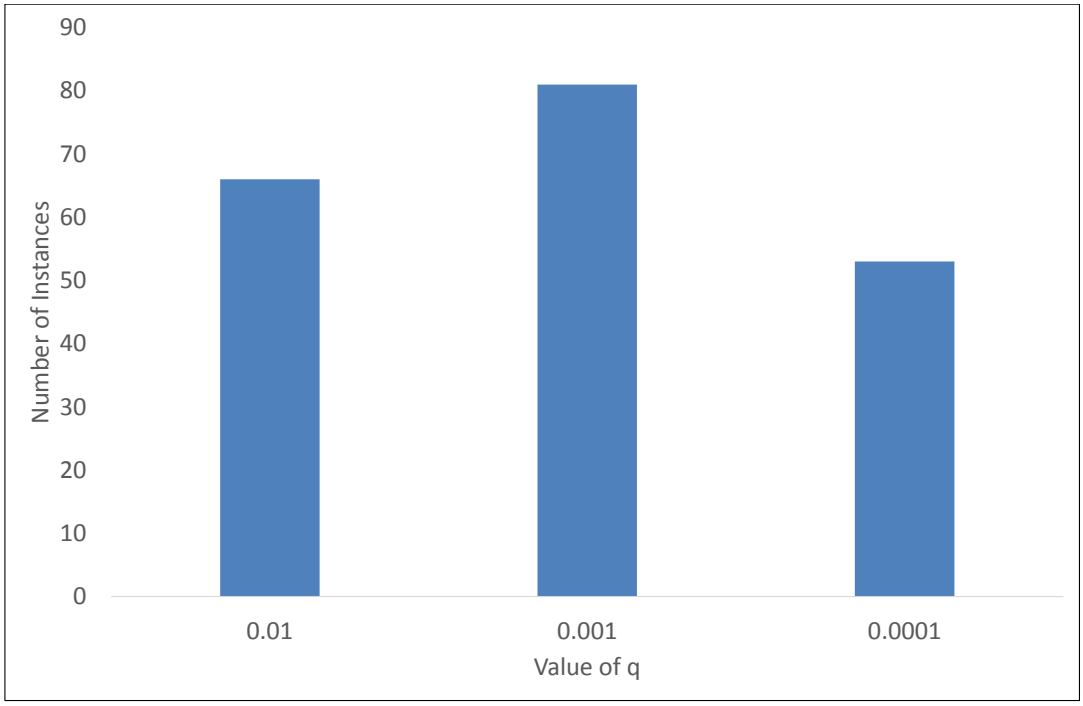


Figure 4.4: Best average performance for q.

we test on random instances for a special case of OCP to evaluate the quality of the heuristic solutions. The SA algorithm, along with the list heuristic, are coded in C#, and the mathematical models are solved using CPLEX 12.6. The tests are performed on a PC with I7-4790 CPU and 8 GB of RAM.

Extracting Industry Instances

The input parameters for the OCP are the tote contents, the processing time of totes, and the number of induction lines. Unfortunately, not all information is explicitly available

in the database provided by the industrial partner. The data provides the order-wave allocation and order content, but not the content of the totes. Since the content of each tote is not known, the processing time of the totes is also unavailable. We use system data characteristics to estimate tote content, and, hence, the processing time of the totes.

Based on the data, it appears that, after inducting its content, the induction operator also inducts the empty tote. Based on this, we extract the content information. We consider all items inducted between two empty totes as belonging to the second empty tote. However, there are anomalies in the data when the scanner cannot read the bar-codes properly. Figure 4.5 shows two excerpts from the database for 15 consecutive induction times. Figure 4.5(a) and Figure 4.5(b) show the induction operations without and with anomalies, respectively.

The first column of the tables in Figure 4.5 is the induction clock time of missions. A mission may refer to a product or an empty tote induction. The mission codes are provided in the second column of the tables. Each mission has its own unique code. Missions starting with "6" are products; whereas those starting with "T" are empty totes. The question marks are missed scans. There is no other data to determine whether a missed scan refers to a product or a tote. The duration between two consecutive inductions may help identify the missing records. Figure 4.5(a) shows that the time difference between inducting the last item in the tote and the empty tote is significantly larger than that between two consecutive items. We use this observation to identify missing records. The reason for the greater time difference after inducting a tote may be due to tote switching. Figure 4.6 shows two excerpts, each with an anomaly. The anomalies are most likely caused by different types of missions (product or empty tote induction). Figure 4.6(a) illustrates an anomaly that is probably due to a missed tote induction while Figure 4.6(b) illustrates an anomaly that is due probably due to a missed product induction. To further strengthen

Row	Induction Time	Mission
1	8:03:30 PM	615728022
2	8:03:31 PM	615728022
3	8:03:33 PM	619369362
4	8:03:34 PM	616647831
5	8:01:35 PM	T00010397
6	8:03:41 PM	613887960
7	8:03:45 PM	613887960
8	8:03:48 PM	616591398
9	8:03:51 PM	618419428
10	8:04:00 PM	613955627
11	8:04:05 PM	613960506
12	8:02:03 PM	T00004984
13	8:04:13 PM	613960506
14	8:04:16 PM	613960506
15	8:04:28 PM	617282538

(a) Induction table without any anomaly.

Row	FirstTimeInducted	Mission
1	8:09:37 PM	618277554
2	8:09:38 PM	612226481
3	8:09:39 PM	615590801
4	8:09:40 PM	614813111
5	8:09:41 PM	614713081
6	8:09:43 PM	614713030
7	8:09:49 PM	617275717
8	8:09:53 PM	T00016968
9	8:10:02 PM	615773659
10	8:10:06 PM	619076594
11	8:10:08 PM	616171086
12	8:10:10 PM	616171086
13	8:10:12 PM	????????????????
14	8:10:17 PM	618983547
15	8:10:19 PM	616519821

(b) Induction table with an anomaly.

Figure 4.5: Screenshots of the induction table.

this analysis, we investigate the break times between consecutive inductions. Two samples are compared. The first consists of break times before and after the induction of a product, while the second sample is for the break times before and after an empty tote.

The warehouse has three putwalls, each fed by four semiautomatic and two manual induction lines for a total of twelve semiautomatic and six manual induction lines. The data distinguishes the semiautomatic lines from the manual ones by their ID numbers. As it is hard to make predictions for the manual induction lines, we focus only on the semiautomatic lines.

Given a sequence of products/totes A, B, C, the break time for B is defined as the time from when A is inducted until the time when C is inducted. Figure 4.7 shows the

Row	FirstTimeInducted	Mission
1	8:09:37 PM	618277554
2	8:09:38 PM	612226481
3	8:09:39 PM	615590801
4	8:09:40 PM	614813111
5	8:09:41 PM	614713081
6	8:09:43 PM	614713030
7	8:09:49 PM	617275717
8	8:09:53 PM	T00016968
9	8:10:02 PM	615773659
10	8:10:06 PM	619076594
11	8:10:08 PM	616171086
12	8:10:10 PM	616171086
13	8:10:12 PM	????????????????
14	8:10:17 PM	618983547
15	8:10:19 PM	616519821

(a) Induction table with an anomaly that is assumed to be caused by an empty tote.

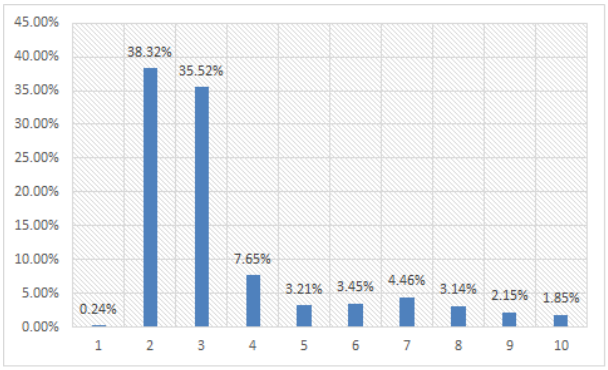
Row	FirstTimeInducted	Mission
1	8:14:39 PM	617122391
2	8:14:40 PM	619117860
3	8:14:41 PM	617399904
4	8:14:43 PM	619101921
5	8:14:45 PM	618498234
6	8:14:48 PM	617398549
7	8:14:54 PM	617398549
8	8:14:57 PM	T00011072
9	8:15:01 PM	614712942
10	8:15:03 PM	614572038
11	8:15:06 PM	620799909
12	8:15:07 PM	618863019
13	8:15:08 PM	????????????????
14	8:15:09 PM	614572038
15	8:15:11 PM	620766437

(b) Induction table with an anomaly that is assumed to be caused by an item.

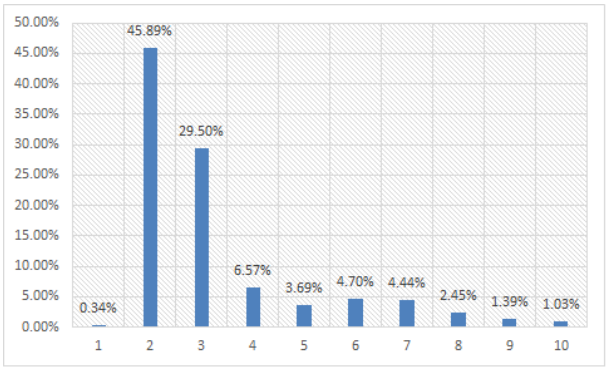
Figure 4.6: Anomaly comparison.

distribution of break times for products at all induction lines that feed putwall-1. Similarly, Figure 4.8 provides the distribution of break times for empty totes at all induction lines that feed putwall-1. Both Figure 4.7 and Figure 4.8 show that the break times for an empty tote are significantly greater than the break times for a product. It is more than five seconds for more than 90% of the totes. To be on the safe side, we use a threshold of eight seconds to identify totes.

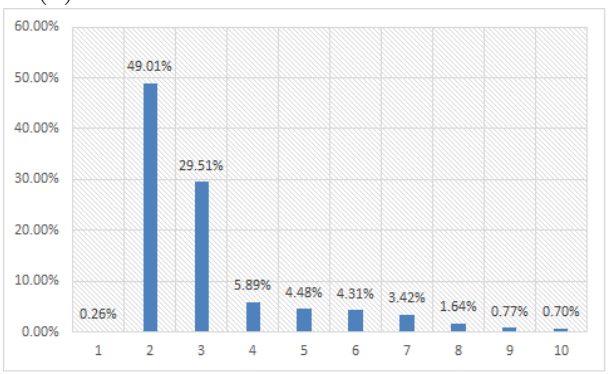
Another important and practical assumption is to set an upper bound on the capacity of a tote. Based on partial but accurate data on tote content, the distribution of the number of products per tote is illustrated in Figure 4.9. The figure shows that the maximum number of items in a tote is 40 with 97.5 % of totes having less than 30 items. Based on this, we set the tote capacity to 30 items. Using this and the threshold of 8 seconds for



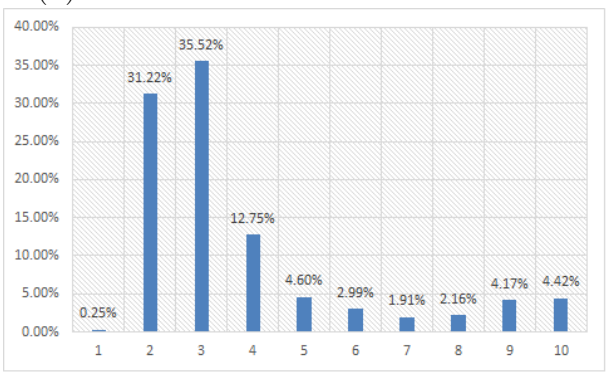
(a) Item induction at line-1.



(b) Item induction at line-2.



(c) Item induction at line-3.



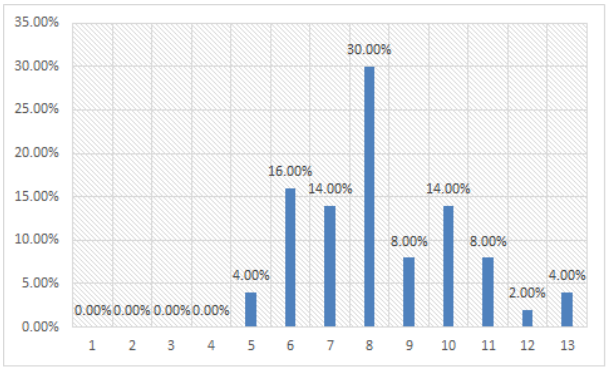
(d) Item induction at line-4.

Figure 4.7: Item induction times (in seconds) at putwall-1.

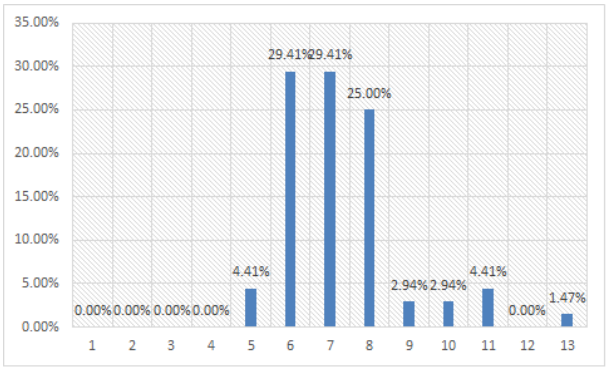
empty totes, the content of totes is identified.

When investigating the number of lines, we find that not all lines are used to process every wave. We provide the number of waves that use a specific number of lines in Table 4.4. Columns 3, 4 and 5 show the average number of totes, orders and products per wave, respectively.

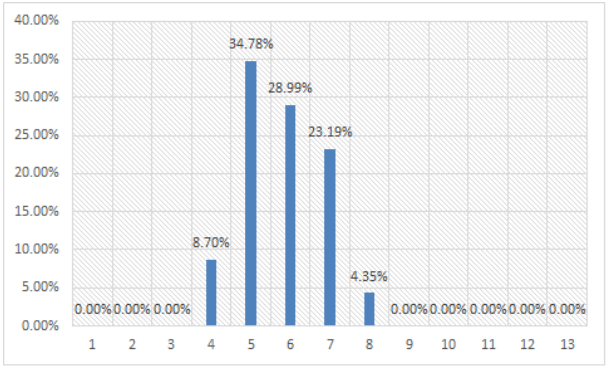
Most of the waves are processed with both manual and semiautomatic induction lines. Few are processed with one, two, or three semiautomatic lines. As mentioned earlier, the data on the waves that use manual lines is not reliable since we cannot derive tote content. Therefore, we continue our study with the waves that use only semiautomatic induction



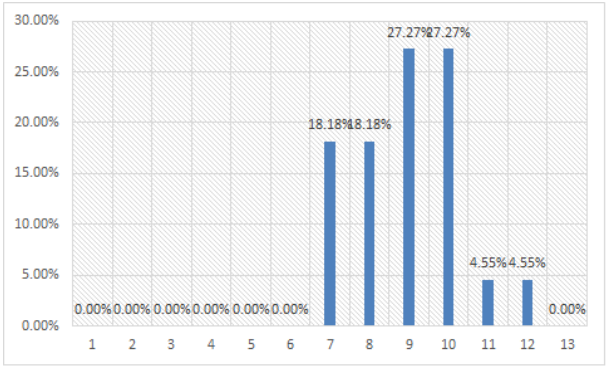
(a) Empty tote induction at line-1.



(b) Empty tote induction at line-2.



(c) Empty tote induction at line-3.



(d) Empty tote induction at line-4.

Figure 4.8: Tote induction times (in seconds) at putwall-1.

lines. Figure 4.10 depicts the usage of each line for 10 waves. A detailed version of Figure 4.10 with the normalized values of processing time (tote emptying), idle time, and blocked time are illustrated in Figure 4.11. Processing time is defined as the time that operators process totes. Idle time is the time that occurs between switching emptied and full totes. Blocked time is the time when there is no tote processing.

Both Figures 4.10 and 4.11 indicate that not all waves are processed when all lines are available. Therefore, we need to take the availability of the lines into consideration while comparing our results to actual system performance. Our purpose is to determine a sequence to improve the order consolidation process; hence, we need to compare sequences

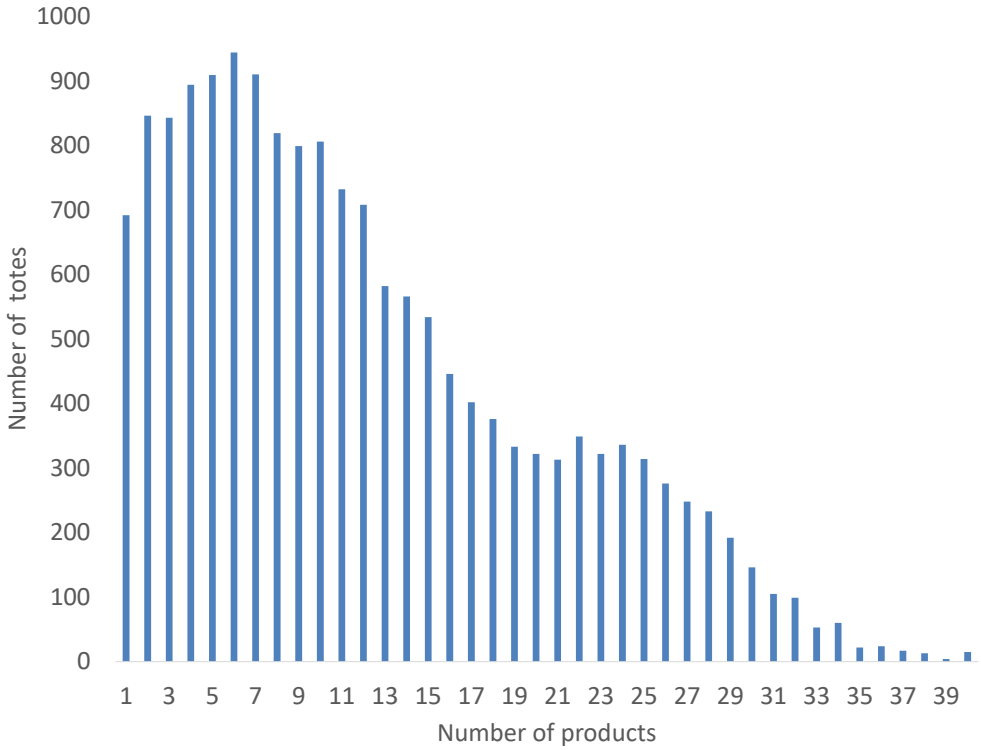


Figure 4.9: Characteristics of the totes.

instead of schedules. We define 2 scenarios to compare the sequences. The first scenario considers all lines are available all the time. To be able to make a comparison for this scenario, we derive the industry sequence from the data and reschedule the totes according to this industry sequence while ignoring block times. The second scenario considers lines with blocked times. We use the optimized sequence to schedule the totes to the lines while enforcing block times.

Table 4.4: The average values of the generated instances.

Induction line	Number of waves	Avg. # totes/wave	Avg. # orders/wave	Avg. # items/wave
1	3	422	1273	5269
2	6	358	1389	5073
3	4	304	1333	5997
4	50	424	1394	6458
6	238	407	1346	5975

Validation of Solution on Industry Instances

In this section, we present the results of SA on industry instances. We test on ten instances. The number of totes, orders, products are given in Table 4.5. Note that the complexity of an instance depends on the order-tote relationship. Detailed information about the instances is provided in Appendix A.

Table 4.5: Industry instances.

		Number of	
Instance	Totes	Orders	Products
W-353-1243	353	1243	4202
W-374-1384	374	1384	6702
W-420-1374	420	1374	6275
W-368-1373	368	1373	5646
W-336-1367	336	1367	5601
W-257-851	257	851	2978
W-253-932	253	932	3925
W-377-989	377	989	4667
W-169-949	169	949	3932
W-121-267	121	267	1234

As explained in Section 4.2.3, we define 2 scenarios based on the availabilities of the lines. Scenario-1 corresponds to all lines being available. Scenario-2 takes line availability into account, which is derived from the data. We compare sequences generated using SA to

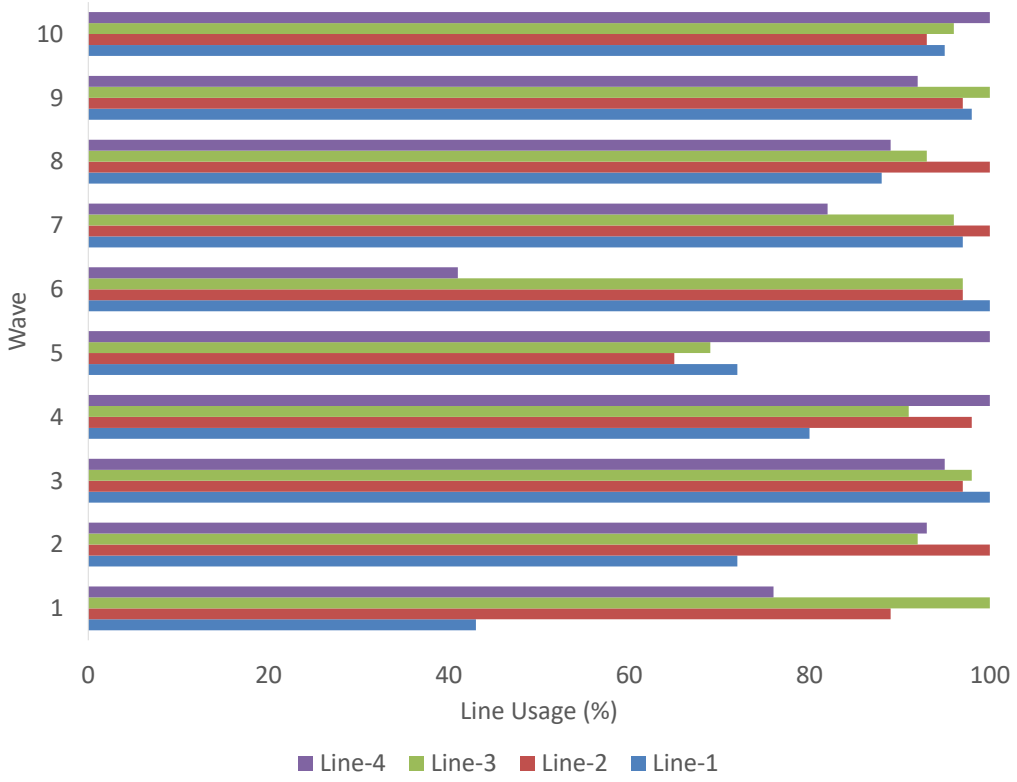


Figure 4.10: Induction line usage for 10 sort waves.

the ones from the data. Once the sequences are known, schedules can be easily generated by assigning the next job in the sequence to the first available line.

We present the average consolidation times (in minutes) per order under both scenarios in Table 4.6. The improvement is calculated as $\frac{100(\text{Current system}-SA)}{\text{Current system}}$. As seen, the improvements over the current system are significant. For Scenario-1, at least 6.74% improvement is achieved with an average of 28.77%. For Scenario-2, these values are 5.96% and 19.9%, respectively. Although consolidation time is the main objective, we compare the number of orders whose consolidation time improves (decreases), increases, or remains the same. This comparison is given in Table 4.7. About 75.66% and 66.96% of orders were

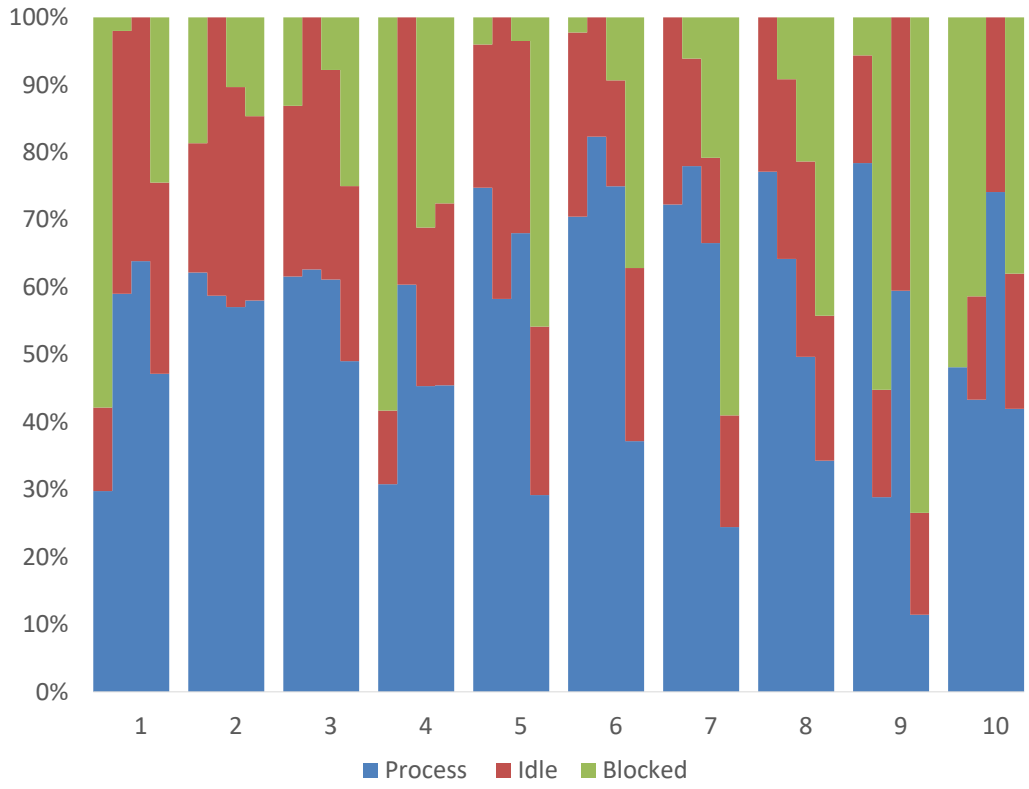


Figure 4.11: Normalized induction line usage for 10 waves.

improved under Scenarios 1 and 2, respectively.

Finally, we investigate the effect of optimizing the consolidation sequence on cubby usage. Once an order's first product is inducted, a cubby is assigned to the order, and that order seizes the cubby until all products are inducted. Table 4.8 compares the cubby usage times in minutes between SA and the current system under Scenarios 1 and 2. On average, 21.92% and 20.71% improvements are achieved Scenarios 1 and 2, respectively. In Figure 4.12, the number of cubbies in use over time are given for the instance W-353-1242. It is clear that a significant improvement is achieved in terms of cubby usage. The current system uses 747 cubbies at any given time, whereas the optimized sequence uses at most

Table 4.6: Comparison of SA and industry sequences.

	Scenario-1			Scenario-2		
Instance	SA	Current System	Improvement (%)	SA	Current System	Improvement (%)
W-353-1243	14.66	25.13	41.66	20.91	31.51	33.64
W-374-1384	27.71	38.94	28.83	70.17	79.25	11.45
W-420-1374	26.01	37.60	30.84	63.61	80.96	21.42
W-368-1373	21.44	30.55	29.82	123.23	139.92	11.93
W-336-1367	21.56	32.22	33.10	23.81	32.19	26.03
W-257-851	11.46	18.65	38.56	23.16	29.84	22.39
W-253-932	17.77	26.85	33.83	21.26	30.27	29.76
W-377-989	30.75	32.97	6.74	45.97	48.88	5.96
W-169-949	18.93	21.52	12.04	23.12	26.65	13.24
W-121-267	5.30	7.83	32.30	5.86	7.64	23.19
Average	19.56	27.22	28.77	42.11	50.71	19.9

Table 4.7: Total number of orders with improved, increased, and the unchanged completion times.

	Scenario-1			Scenario-2		
Wave	Improved	Increased	Unchanged	Improved	Increased	Unchanged
W-353-1243	1020	209	14	918	309	16
W-374-1384	1098	261	25	920	423	41
W-420-1374	1099	258	17	959	384	31
W-368-1373	1075	285	13	891	464	18
W-336-1367	1102	248	17	979	373	15
W-257-851	689	157	5	577	261	13
W-253-932	771	151	10	722	195	15
W-377-989	479	457	53	426	499	64
W-169-949	579	332	38	605	315	29
W-121-267	206	59	2	188	73	6
Total	8118	2417	194	7185	3296	248
%	75.66	22.53	1.81	66.97	30.72	2.31

658 which is about 12% improvement over the current system.

Table 4.8: Cubby usage times.

	Scenario-1			Scenario-2		
Instance	SA	Current System	Improvement (%)	SA	Current System	Improvement (%)
W-353-1243	10.22	15.04	32.06	14.26	21.56	33.86
W-374-1384	22.22	28.28	21.44	34.56	40.35	14.36
W-420-1374	20.53	26.64	22.92	43.42	57.13	24.00
W-368-1373	16.65	50.29	20.57	77.33	88.05	12.07
W-336-1367	16.96	22.47	24.51	18.49	24.99	26.01
W-257-851	7.26	10.48	30.70	12.97	16.74	22.54
W-253-932	14.60	19.71	25.93	17.44	24.10	27.62
W-377-989	22.05	23.57	6.46	32.35	35.83	9.73
W-169-949	13.34	15.52	14.07	16.66	19.91	16.32
W-121-267	3.82	4.80	20.49	4.15	5.22	20.47
Average	14.76	18.75	21.92	27.16	33.39	20.71

Quality of SA solution on Random Instances

To test the quality of the solutions generated by SA, we test on six small problems shown in Table 4.9. Several different parameter settings are generated randomly for each problem. The maximum number of totes an order can have its products in is denoted by γ, and is randomly assigned values 3, 4, or 5. The number of induction lines is set to 4, 5 or 6. After the orders are generated, we distribute the products of an order among the totes randomly. The tote processing time is assumed to be constant and is set to 5. 20 instances were generated for each of the six small problems.

To solve OCP to optimality, we take advantage of the fact that processing times are

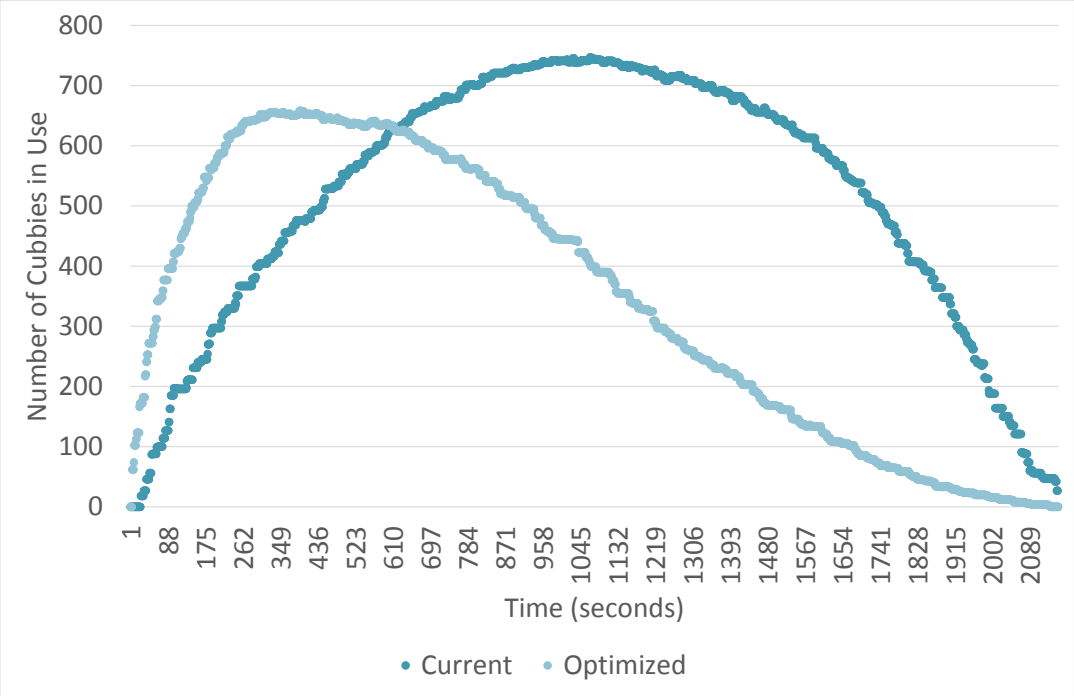


Figure 4.12: Cubby usage in time.

identical and solve the following formulation:

$$\text{[OCPP]: min} \sum_{k \in K} CO_k \tag{4.13}$$

$$\text{s.t. } CO_k \geq p \sum_{t \in T} t z_{jt} \qquad j \in J_k, k \in K \tag{4.14}$$

$$\sum_{j \in J} z_{jt} \leq m \qquad t \in T \tag{4.15}$$

$$\sum_{t \in T} z_{jt} = 1 \qquad j \in J \tag{4.16}$$

Table 4.9: Small random problems.

Problem	Number of Totes	Number of Orders
1	10	20
2	20	30
3	20	40
4	30	40
5	30	50
6	30	60

$$z_{jt} = 0, 1 \qquad j \in J, t \in T \tag{4.17}$$

where $z_{jt} = 1$ if tote j is processed at time slot t and p is the tote processing time. The objective function (4.13) minimizes the total completion time of orders. Constraints (4.14) calculate the completion time of orders with respect to time slot assignments. Constraints (4.15) ensure that at most m totes are assigned to a time slot to prevent assigning totes to more than the number of induction lines. Constraints (4.16) guarantee that all totes are processed.

Table 4.10 displays the number of induction lines, instance number, lower bound, and best objective from solving [OCPP] in Cplex 12.6 for 600 seconds (LB and OCPP) and the objectives achieved by the list heuristic and SA. The rest of the columns display gaps and CPU times. Table 4.10 proves that SA is quite efficient in terms of both CPU time and objective function values. SA achieves better values than OCPP for two instances because of the time limit on OCPP. It is important to emphasize that for the instances that OCPP solved within the time limit, the average relative gap of SA is less than 0.01%. Also large gaps for SA correspond to large gaps for [OCPP]. This is most probably due to the quality of the lower bound LB.

Table 4.10: Results of the random problems.

Instance		Result				GAP (%)			CPU (seconds)		
Induction Lines	Problem	LB	OCPP	List	SA	GAP1	GAP2	GAP3	OCPP	List	SA
4	1	192.25	192.25	201.00	192.25	0.00	0.00	0.00	0.00	0.00	0.00
4	2	301.75	301.75	313.00	301.75	0.00	0.00	0.00	0.00	0.00	0.00
4	3	629.75	629.75	674.75	629.75	0.00	0.00	0.00	2.40	0.00	0.00
4	4	777.24	848.75	950.00	848.50	9.17	9.20	-0.03	695.20	0.00	0.00
4	5	954.07	1095.75	1209.00	1095.25	14.80	14.85	-0.05	823.50	0.00	1.00
4	6	1142.48	1176.75	1264.25	1176.75	3.00	3.00	0.00	1504.15	0.00	1.00
5	1	158.00	158.00	163.50	158.00	0.00	0.00	0.00	0.00	0.00	0.00
5	2	244.75	244.75	253.50	244.75	0.00	0.00	0.00	0.00	0.00	0.00
5	3	524.50	524.50	558.25	524.50	0.00	0.00	0.00	0.05	0.00	0.00
5	4	694.03	697.50	777.75	698.00	0.57	0.50	0.07	128.85	0.00	0.05
5	5	883.16	899.50	990.00	899.75	1.88	1.85	0.03	335.95	0.00	1.00
5	6	969.50	969.50	1037.75	969.75	0.03	0.00	0.03	2.10	0.00	1.00
6	1	147.00	147.00	151.50	147.00	0.00	0.00	0.00	0.00	0.00	0.00
6	2	229.25	229.25	234.75	229.25	0.00	0.00	0.00	0.00	0.00	0.00
6	3	459.00	459.00	488.50	459.00	0.00	0.00	0.00	0.20	0.00	0.00
6	4	599.00	599.00	667.25	599.25	0.04	0.00	0.04	12.95	0.00	0.20
6	5	771.00	771.00	844.25	771.25	0.03	0.00	0.03	10.05	0.00	1.80
6	6	835.00	835.00	889.00	835.00	0.00	0.00	0.00	0.70	0.00	1.15

GAP1 $= 100\frac{(SA-LB)}{LB}$
GAP2 $= 100\frac{(OCPP-LB)}{LB}$
GAP3 $= 100\frac{(SA-OCPP)}{OCPP}$

4.3 Conclusions

In this chapter, we define the order consolidation problem. We present a mathematical formulation of OCP and propose a simulated annealing (SA) algorithm. We calibrate the parameters of the SA algorithm and conduct two computational tests. In the first one, we evaluate the proposed solutions using industry instances. For the first test, to extract tote contents from the data, we analyze induction operations further. We show that in the data there are anomalies which prevent determining tote content accurately. Using information extracted from the data, we devise an algorithm to estimate tote content. To provide reliable analysis, we also consider the usage of the induction lines. We show that while processing a wave, not all induction lines are always available. To compare the SA

sequences to industry instances, we define two scenarios, one of which considers that all lines are available at all time whereas the other takes blocked times into consideration. For the first scenario, at least 6.74% improvement is achieved with the suggested schedules (sequences) whereas the average improvement is 28.77%. For the second scenario, these values are 5.96% and 19.9%, respectively.

In the second computational test, to asses the quality of the SA solutions on random instances, we define a special case of the OCP where processing time of all totes is the same. The quality of the SA sequences are near optimal with an average relative gap of less than 0.01%.

In the next chapter, we develop an exact solution framework based on a branch-and-price approach.

Chapter 5

A Branch-and-Price Approach for the Order Consolidation Problem

In this Chapter, we develop a branch-and-price algorithm for the order consolidation problem. We use a set-partitioning formulation that is equivalent to formulation [NMIP] of Section 4.1. Let S denote the set of all partial schedules on a single induction line. For each tote $j \in J$, let parameter $b_j^s = 1$ if schedule $s \in S$ includes tote j, and 0 otherwise. Defining variable y_s to take value 1 if schedule $s \in S$ is selected and 0 otherwise, the set partitioning formulation is:

$$\text{[ISP]: min} \sum_{k \in K} CO_k \tag{5.1}$$

$$\text{s.t. } CO_k \geq \sum_{s \in S} C_j^s y_s \qquad j \in J_k, k \in K \tag{5.2}$$

$$\sum_{s \in S} b_j^s y_s = 1 \qquad j \in J \tag{5.3}$$

$$\sum_{s \in S} y_s = m \tag{5.4}$$

$$y_s = 0, 1 \qquad s \in S \qquad (5.5)$$

Constraints (5.3) and (5.4) correspond to the original constraints (4.2) and (4.3), which are the assignment and induction line capacity constraints. Constraints (5.3) ensure that each tote is covered by exactly one partial schedule. Constraint (5.4) ensures that m schedules are selected. Constraints (5.2) calculate the completion time of the orders according to the selected schedules. The set-partitioning formulation is solved by column generation, as described in the following section.

5.1 Solution by Column Generation

To solve [ISP], we need to generate all feasible partial schedules in set S. Since the number of feasible partial schedules can be too large, it is not practical to generate all of them. Instead, schedules are generated iteratively. Let [LSP] denote the linear relaxation of [ISP]. Its dual problem is:

$$\text{[DSP]: max } m\lambda_0 + \sum_{j \in J} \lambda_j \qquad (5.6)$$

$$\text{s.t. } \sum_{j \in J_k} \beta_{jk} \leq 1 \qquad k \in K \qquad (5.7)$$

$$\sum_{k \in K} \sum_{j \in J_k} C_j^s \beta_{jk} - \sum_{j \in J} b_j^s \lambda_j - \lambda_0 \geq 0 \qquad s \in S \qquad (5.8)$$

$$\beta_{jk} \geq 0 \qquad j \in J_k, k \in K \qquad (5.9)$$

In [DSP], β_{jk} denotes the dual variable corresponding to tote j and order k, for each $j \in J_k$, $k \in K$, and corresponds to constraints (5.2). The dual variable λ_j corresponds to tote j, for each $j \in J$, and corresponds to constraints (5.3). λ_0 is the dual variable corresponding

to the total available lines and corresponds to constraint (5.4).

Column generation starts by solving [LSP] on a subset of schedules $\bar{S} \subseteq S$ and then verifies whether the solution of the restricted problem (LSP[$\bar{S}$]) is optimal to the original problem [LSP]. If it is not optimal, then a new schedule is generated by solving a pricing problem [PP]:

$$\text{[PP] min} \sum_{j \in J} \sum_{k: j \in J_k} \bar{\beta}_{jk} C_j - \sum_{i \in J_0} \sum_{j \in J} \bar{\lambda}_j x_{ij} \tag{5.10}$$

$$\text{s.t. (4.4), (4.5), (4.7), (4.9)} \tag{5.11}$$

The pricing problem [PP] is a single machine scheduling problem.

If the objective value of [PP] minus $\bar{\lambda}_0$ is negative, the corresponding solution defines a new partial schedule for $\bar{S}$, and the process is restarted. Columns are added to [LSP($\bar{S}$)] until no column with negative reduced cost is identified. In practice, an alternative stopping criterion is when the relative gap between the lower bound and the upper bound is less than a predetermined value (e.g. 0.001). The lower bound of [LSP] at iteration h is calculated with the equation $LB_h = \max\{LB_{h-1}, UB + u\}$ where u is the objective function of the partial schedule with the most negative value, i.e., the objective function value of [PP]. The value of u is at most 0, since the objective of [PP] is a minimization and (0,0) is feasible. UB is the objective value of [LSP($\bar{S}$)] at iteration $h-1$. Since ($\bar{S}$) is updated at every iteration, UB either decreases or stays the same. Therefore, we do not need to check whether or not UB is improved at every iteration. The update of LB is valid because when u is 0, i.e., there are no columns with negative reduced cost, $LB = UB$, and the algorithm stops.

The solution of the pricing problem [PP] is crucial to the efficiency of the branch-and-price approach. In the following section, we describe a dynamic programming algorithm

to solve it.

5.2 A Dynamic Programming Algorithm for the Pricing Problem

According to Smith's rule (Smith, 1956), in any optimal schedule for the parallel machine scheduling problem with the objective of minimizing weighted completion time, jobs on each machine must follow the shortest weighted processing time (SWPT) order. Subproblem [PP] determines a subset of jobs that minimize the total weighted completion time minus the sum of the dual variables. For a given subset of jobs, the problem reduces to a minimum total weighted completion time problem on a single machine, whose optimal solution satisfies the SWPT order. Therefore, any optimal solution must satisfy the SWPT order. Based on this important observation, Chen and Powell (1999) propose a dynamic programming algorithm with $O(nP)$ worst case complexity where P is the total processing time of the jobs and n is the number of jobs. The algorithm uses the recursion function $F(j,t)$ defined as the minimum objective value of a partial schedule that contains a subset of jobs of $\{1,2,...,j\}$, that satisfies the SWPT rule, and is completed at time t. The algorithm is initialized as follows: $F(j,t) = \infty$ for $t < 0, j = 0,...,n$; $F(0,t) = 0$ for $t \geq 0$. And the recursion function is given by $F(j,t) = \min\{F(j-1, t-p_j) + tw_j - \lambda_j, F(j-1,t)\}$ where w and λ are the weight and the dual variable value of the corresponding job. Then the solution of the problem is $\min_{0 \leq t \leq P} F(n,t)$. In PP, the weight of a tote (same as job in the machine scheduling problem) is calculated as $\sum_{k:j \in J_k} \bar{\beta}_{jk}$ for $j \in J$. The dynamic programming algorithm is given in Algorithm 2.

In the following section, we devise a heuristic to warm start the column generation procedure.

Algorithm 2 Dynamic Programming-1

FOR each $j \in N$,t WHERE t< 0 set $F(j,t) = \infty$
FOR each t set $F(0,t) = 0$
FOR each $j \in N$
    FOR each $t \in P$
        $F(j,t) = \min\{F(j-1, t-p_j) + tw_j - \lambda_j, F(j-1,t)\}$
    end FOR
end FOR
$z^* = \infty$
FOR each $j \in N, t \in P$
    IF ($z^* \leq F(j,t)$) THEN
        $z^* \leftarrow F(j,t)$
    END IF
end FOR

5.3 Heuristic Algorithm to Generate Initial Columns

Since the convergence of column generation depends on the quality of the starting columns, we provide a construction heuristic to generate the initial columns. The goal is to generate complete solutions to OCP for use as initial columns.

The heuristic determines the schedule based on the relation between totes. We consider that two totes are neighbours if there is at least one order that has items in both totes. Starting from an arbitrary tote, the heuristic sorts its neighbours in a descending order of processing time. The neighbours are assigned one by one to the first available line. Once all the neighbours of the tote under consideration are assigned, the last tote is picked as the new tote whose neighbours are identified and the same assignment procedure is applied. The heuristic can generate up to nm columns by varying the starting tote.

5.4 Branch and Price Algorithm

The column generation procedure of Section 5.1 is embedded within a branch-and-bound tree, leading to a branch-and-price ($b\&p$) framework. If a predetermined solution quality is not achieved at the root node, branching is performed. The branching rule is crucial to the success of the $b\&p$ algorithm. Branching on the original variable y_s is difficult to enforce for $y_s = 0$. The column corresponding to y_s represents a feasble partial schedule on one of the induction lines. Setting y_s to 0 implies that the schedule has to be removed. It is not easy to exclude such a schedule when solving the single machine scheduling subproblem.

Fortunately, there is a solution to this difficulty. Instead of branching on y_s, we branch on totes being at the same column or not. The [NMIP] formulation uses variable x_{ij} to define whether or not tote j immediately follows tote i. If we branch on the x variables, the precedence relationships between totes has to be considered in the subproblem. Both van Den Akker et al. (1999) and Chen and Powell (1999) easily include the precedence relations to the subproblem. Since there is at least one optimal solution satisfying Smith's rule, only columns that satisfy Smith's rule are generated. When determining the branching variable, the relationship between jobs is enforced while satisfying Smith's rule. For OCP, however, such a relationship does not exist as the objective does not minimize TWCT. The subproblem reduces to a single machine scheduling problem with TWCT objective at the root node, but not after branching.

After branching, van Den Akker et al. (1999) and Chen and Powell (1999) can still generate columns according to Smith's rule with a revised version of Algorithm 2 that is given in Algorithm 3. They define a set, denoted by B_j, for job j, to determine the jobs that can be processed immediately before job j. The set B_j for job j is updated according to the branching rule, and the value of the new recursion is calculated as $F(j,t) =$

$\min_{i \in B_j \cup \{0\}} \{F(i, t - p_j) + tw_j - \lambda_j\}$.

Algorithm 3 Dynamic Programming Algorithm for the Subproblems after Branching When Smith's Rule is Applicable

FOR each $j \in N$,t WHERE t< 0 set $F(j,t) = \infty$
FOR each t set $F(0,t) = 0$
FOR each $j \in N$
    FOR each $t \in P$
        $F(j,t) = \min_{i \in B_j \cup \{0\}} \{F(i, t - p_j) + tw_j - \lambda_j\}$
    end FOR
end FOR
$z^* = \infty$
FOR each $j \in N, t \in P$
    IF ($z^* \leq F(j,t)$) THEN
        $z^* \leftarrow F(j,t)$
    END IF
end FOR

5.5 Solving the Subproblems after Branching

To solve the subproblems after branching, a commercial solver can be used. The computational time, however, may be excessive and could hinder the performance of the entire branch-and-price algorithm. To overcome this, we propose an alternative mathematical formulation for OCP with two sets of binary variables instead of the x variables. We use the same indices, parameters, and continuous variables as [NMIP]. In addition, we define a set of induction lines, denoted by L. We define z_{ij} which takes value 1 if tote $i, j \in J$ are processed on the same induction line and tote i is processed earlier than tote j. We also define binary variable t_{jl} which takes value 1 if tote $j \in J$ is processed in line $l \in L$. The alternative formulation is:

$$\text{[NMIP2]: } \min \sum_{k \in K} CO_k \tag{5.12}$$

$$\text{s.t. } C_i + p_j \leq C_j + M(1 - z_{ij}) \qquad i, j \in J \tag{5.13}$$

$$z_{ij} + z_{ji} = \sum_{l \in L} t_{il} t_{jl} \qquad i, j \in J \tag{5.14}$$

$$\sum_{l \in L} t_{jl} = 1 \qquad j \in J \tag{5.15}$$

$$CO_k \geq C_j \qquad j \in J_k, k \in K \tag{5.16}$$

$$z_{ij} \in \{0, 1\} \qquad i, j \in J \tag{5.17}$$

$$t_{jl} \in \{0, 1\} \qquad j \in J, l \in L \tag{5.18}$$

The objective function (5.12) minimizes the total completion time of all orders. Constraints (5.13) calculate the completion times of totes. Constraints (5.14) are nonlinear and they ensure that $z_{ij} + z_{ji}$ is 1 when both totes i and j are processed in the same induction line; otherwise, the value of $z_{ij} + z_{ji}$ becomes 0. Constraints (5.15) are assignment constraints that force each tote to be processed on only one induction line. Constraints (5.16) calculate the completion times of the orders.

In the subproblem, we solve a single machine scheduling problem by selecting and scheduling a subset totes. Let us present a formulation of subproblem [PP] that uses z and t as variables:

$$\text{[SPP2]: } \min \sum_{j \in J} \sum_{k : j \in J_k} \bar{\beta_{jk}} C_j - \sum_{j \in J} \bar{\lambda_j} t_j \tag{5.19}$$

$$\text{s.t. } C_i + p_j t_j \leq C_j + M(1 - z_{ij}) \qquad i, j \in J \tag{5.20}$$

$$z_{ij} + z_{ji} \leq 0.5(t_i + t_j) \qquad i, j \in J \qquad (5.21)$$

$$t_i + t_j - 1 \leq z_{ij} + z_{ji} \qquad j \in J \qquad (5.22)$$

$$z_{ij} \in \{0, 1\} \qquad i, j \in J \qquad (5.23)$$

$$t_j \in \{0, 1\} \qquad j \in J \qquad (5.24)$$

Note that $t_j = 1$ implies that tote j is selected. The objective function (5.19) minimizes the total weighted completion time of the selected totes plus the cost of selecting a tote to schedule. Constraints (5.20) calculate the completion time of tote $j \in J$. Constraints (5.21) and (5.22) enforce the relationship between totes if they are both selected.

Observing the relation between [ISP] and [SPP2], we can see that for any feasible solution y_s; $s \in S$, there is a corresponding feasible solution z_{ij}, $i, j \in J$ and T_j, $j \in J$ where

$$z_{ij} = \sum_{s \in S} e_{ij}^s y_s \qquad (5.25)$$

where e_{ij}^s is 1 if totes i and j are both contained in schedule s and tote i is processed before tote j. Once the root node is solved, if the solution $y_s; s \in S$ is fractional, then the corresponding z variables are computed using (5.25).

5.6 An Alternative Branching Rule

Instead of branching on the z variables, we propose branching on having two totes in the same column. Accordingly, a pair of totes (i, j) is selected such that $z_{ij} + z_{ji}$ is closest to 0.5, i.e. with the maximum integer infeasibility. Five branches are then created, two of the branches with $z_{ij} + z_{ji}$ fixed to 1 and the other branches with $z_{ij} + z_{ji}$ fixed to 0. If

$z_{ij} + z_{ji}$ is fixed to 0, then the initial restricted master problem of the corresponding child node consists of all the columns of its parent node except the ones in which tote i and tote j are scheduled at the same column. If $z_{ij} + z_{ji}$ is fixed to 1, then the initial restricted master problem of the corresponding child node consists of all the columns of its parent node except the ones in which only one of tote i or tote j is scheduled in a column. When $z_{ij} + z_{ji} = 0$, there are three cases for the subproblem. We can select only tote i, only tote j or neither. As a result, the first child node imposes that tote i and j cannot be scheduled at the same column, and while generating columns with negative reduced cost values, columns must contain only tote i from tote pair (i, j). The second child node is the same as the first child node except the column generation constraint. For the second one, columns that contain only tote j from tote pair (i, j) have to be generated. The third child node is similar to the other child nodes, but no column containing neither tote from tote pair (i, j) can be scheduled in the subproblem. For the child nodes where $z_{ij} + z_{ji}$ is fixed to 1, there are two cases: both totes of pair (i, j) are scheduled in the columns generated by the subproblem, or neither of the columns from pair (i, j) are scheduled in the same column. As a summary, we use z variables to select columns when branching, whereas we use t variables to enforce the branching when solving the subproblems within column generation. Figure 5.1 illustrates the branching rule. The third level in Figure 5.1 corresponds to the case where $z_{ij} + z_{ji} = 1$ but y_s is still fractional. In that case, we further branch on whether i proceeds j or vice versa.

As mentioned earlier, the dynamic programming algorithm of Chen and Powell (1999) fails to accumulate precedence based branching constraints. To overcome this, we propose a new dynamic programming algorithm. Let $\mathrm{F}(j, t)$ denote the minimum objective value (total weighted completion time minus the sum of the dual variable values minus the value of selecting a tote) in a partial schedule consisting of a subset of totes of 1, 2,..., j,

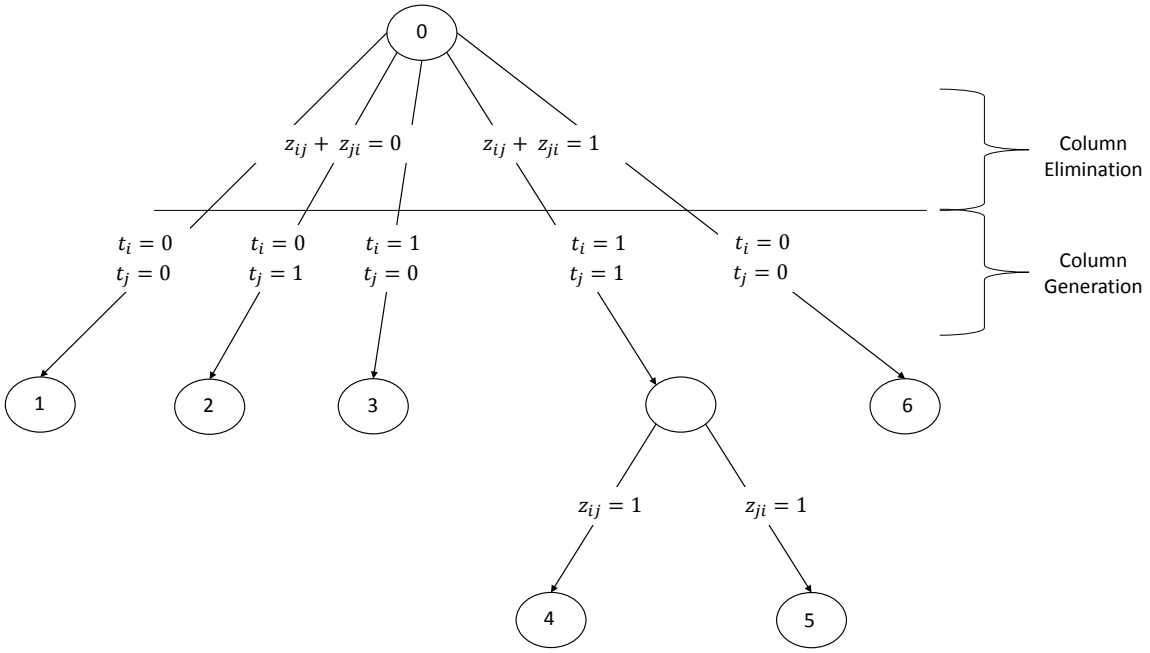


Figure 5.1: An illustration of the branching rule.

provided that the partial schedule follows SWPT order and that tote j is the last tote that is completed at time t in the partial schedule.

To fix or eliminate a tote, due to the branching constraints, we define a cost π_j and set it large enough to eliminate tote j, or sufficiently negative to select tote j. The dynamic programming algorithm is provided in Algorithm 4. The worst-case complexity of Algorithm 4 is bounded by $O(n^2P)$ as there are a total of nP states, and it takes no more than $O(n)$ time to evaluate a state.

5.7 Computational Analysis

In this section, we test the column generation algorithm. The performance of the algorithm can be affected by the number of starting columns and the number of columns generated

Algorithm 4 Dynamic Programming Algorithm for the Subproblems after Branching

UPDATE this FOR each $j \in N$,t WHERE t< 0 set $F(j,t) = \infty$
FOR each t set $F(0,t) = 0$
FOR each $j \in N$
    FOR each $t \in P$
        $F(j,t) = \min_{i \in B_j \cup \{0\}} \{F(i, t - p_j) + tw_j - \lambda_j - \pi_j\}$
    end FOR
end FOR
$z^* = \infty$
FOR each $j \in N, t \in P$
    IF ($z^* \leq F(j,t)$) THEN
        $z^* \leftarrow F(j,t)$
    END IF
end FOR

for each subproblem. We determine four strategies and compare them to determine the best setting. Strategies are presented in Table 5.1. We tested the strategies on five test problems whose configurations are summarized in Table 5.2.

Generating a feasible solution for OCP is easy; therefore, we generate as many feasible solutions as the number of totes, and then generate random columns according to those solutions. Random columns are generated regardless of the strategy tested. The construction heuristic can generate as many solutions as the number of totes. Another important decision is the number of columns generated from the dynamic programming subproblem. For a valid lower bound, the column with the most negative reduced cost should be generated. Since we use a dynamic programming algorithm, we can also easily obtain other columns with negative reduced costs. Adding all negative reduced cost columns may increase the solution time of the master problem, whereas adding just one may lead to poor convergence.

For testing, the number of totes that an order is split over (γ) is set to 1, 2, 3 and the number of induction lines is set to 2, 3, 4, 5 and 6. Items from an order are assigned

Table 5.1: The strategies used for the branch-and-price algorithm.

	Initial Columns		Column Generation	
Strategy	Random Columns	Construction Heur.	Optimal	Best 10 Columns
1	✓	✓	✓	
2	✓	✓	✓	✓
3	✓		✓	
4	✓		✓	✓

Table 5.2: Test problems.

Problem No	n	$\|K\|$
1	10	20
2	20	40
3	30	60
4	40	80
5	50	100

randomly to its totes. The process time of tote $j \in J$ is randomly selected from the interval $[20, 80]$. Five instances are generated for each problem setting. The algorithms are tested on a 64-bit Windows 7 operating system with 8 GB RAM and a 3.6 GHz Intel i7-4790 processor, coded using C#, and solved using Cplex 12.6. Each random instance is run for a maximum of 3600s.

The average results for the four strategies are presented in Tables 5.3, 5.4, 5.5 and 5.6, respectively. The tables provide the number of tree nodes explored, the number of columns generated, and the relative gap under different values of γ.

The results show that, regardless of the strategy chosen, higher values of γ lead to higher gaps. Also, as expected, there is a direct relation between the number of columns generated and the number of nodes explored. When we compare the strategies, strategy 2 seems to be the best. Strategy 4 is the second best, and strategy 3 is the worst.

Table 5.3: Results of strategy 1.

Problem			Nodes Explored			Columns Generated			Relative Gap		
$\|J\|$	$\|K\|$	$\|L\|$	$\gamma = 1$	$\gamma = 2$	$\gamma = 3$	$\gamma = 1$	$\gamma = 2$	$\gamma = 3$	$\gamma = 1$	$\gamma = 2$	$\gamma = 3$
10	20	2	1.00	73.20	176.40	227.40	1413.00	5403.00	0.05	0.89	0.97
10	20	3	1.00	274.80	776.40	171.20	2002.80	6323.20	0.00	0.75	0.98
10	20	4	1.00	580.20	1281.00	149.40	3017.40	8146.00	0.00	0.84	0.97
10	20	5	1.00	1570.60	1803.40	145.20	6729.40	10861.40	0.00	0.88	0.83
10	20	6	1.00	155.20	1885.40	157.40	1035.80	8600.60	0.00	0.60	0.87
20	40	2	1.00	440.80	1758.80	274.20	23318.00	91006.40	0.04	0.80	0.98
20	40	3	1.00	1841.60	6039.80	221.40	42990.40	164033.20	0.18	0.74	3.03
20	40	4	1.00	4814.00	12013.00	199.80	50907.80	155855.80	0.07	0.80	3.52
20	40	5	1.00	1608.20	14996.60	184.20	11367.20	107637.80	0.22	0.70	3.10
20	40	6	1.00	6110.00	13345.40	189.60	32940.00	72429.00	0.02	0.87	2.54
30	60	2	1.00	118.20	383.40	544.20	16712.60	75161.60	0.01	0.93	2.50
30	60	3	1.00	496.00	1533.80	368.40	17631.40	86154.40	0.05	0.94	2.22
30	60	4	1.00	838.20	2925.00	350.20	15398.40	74107.40	0.06	0.87	2.39
30	60	5	1.00	4.20	2616.60	340.60	1726.80	38146.00	0.11	0.85	2.05
30	60	6	1.00	15.00	2708.60	363.80	1626.40	22433.80	0.21	0.67	1.93
40	80	2	1.00	1.00	54.80	1080.20	1310.60	24361.60	0.01	0.64	2.53
40	80	3	1.00	41.60	265.20	511.80	3091.00	32229.40	0.02	0.83	4.73
40	80	4	1.00	364.40	419.00	387.20	11485.00	27569.20	0.03	0.78	3.72
40	80	5	1.00	208.60	437.00	391.80	5735.00	18057.20	0.05	0.84	2.96
40	80	6	1.00	121.00	128.40	384.40	2789.20	6713.40	0.03	0.76	2.38
50	100	2	1.00	2.20	7.40	492.80	2094.00	8804.00	0.02	0.56	5.22
50	100	3	1.00	35.80	47.20	447.60	5265.00	9971.60	0.02	2.32	8.45
50	100	4	1.00	66.80	69.20	350.40	4336.00	9185.00	0.03	2.34	6.99
50	100	5	1.00	66.80	36.80	372.80	3128.40	5234.40	0.07	4.71	8.21
50	100	6	1.00	30.80	13.00	333.80	1866.80	3269.20	0.10	3.98	8.02

Table 5.4: Results of strategy 2.

Problem			Nodes Explored			Columns Generated			Relative Gap		
$\|J\|$	$\|K\|$	$\|L\|$	$\gamma = 1$	$\gamma = 2$	$\gamma = 3$	$\gamma = 1$	$\gamma = 2$	$\gamma = 3$	$\gamma = 1$	$\gamma = 2$	$\gamma = 3$
10	20	2	1	113.2	272.8	211.8	1142.6	2130.4	0.00	0.83	0.95
10	20	3	1	433.6	1298.6	182.6	2177.4	6852	0.00	0.75	0.99
10	20	4	1	1106.2	3003	143.4	4511.8	12812.6	0.00	0.84	0.99
10	20	5	1	1478.6	2627	135.2	6814.4	12923.4	0.00	0.88	0.83
10	20	6	1	299.8	2420.4	149.6	1241	9284.4	0.00	0.61	0.88
20	40	2	1	529.2	4338.8	808.2	19685.4	146448.6	0.04	0.70	1.13
20	40	3	1	2878	7100.4	636.8	44416.2	131676	0.20	0.72	3.21
20	40	4	1	3610	13329.2	645.4	28701.6	101511.4	0.04	0.81	3.38
20	40	5	1	3497.2	14268.2	561	17569.6	73408.8	0.29	0.64	2.80
20	40	6	1	4800.2	6931	509.6	22339.8	27956.2	0.11	0.82	2.63
30	60	2	1	146.8	472	2029	14531.6	65495.4	0.01	0.79	2.36
30	60	3	1	522.2	1724.4	1372.4	12244.6	63421.6	0.05	0.67	2.05
30	60	4	1	841.8	2457.6	1265.8	9387.6	41721.4	0.06	0.71	2.00
30	60	5	1	1	922.2	1295.4	1337.2	13040.8	0.11	0.63	1.56
30	60	6	1	1.8	245.6	1173.4	1160.6	3890.6	0.21	0.55	1.68
40	80	2	1	1	63.4	3553.8	9609.2	23451.8	0.01	0.58	1.73
40	80	3	1	97	218	2594.2	9431.2	25859.6	0.02	0.67	1.90
40	80	4	1	1	190.4	1983.2	2829.6	11999.4	0.03	0.61	1.91
40	80	5	1	200.6	47.6	2015.4	5668.2	4421.6	0.05	0.65	1.86
40	80	6	1	1.2	41	2151.6	2299.2	3489.4	0.03	0.74	1.62
50	100	2	1	1	6	5513.4	9197.8	10081.2	0.02	0.42	1.55
50	100	3	1	1	18.2	3418.8	4406.6	6766.6	0.02	0.47	1.47
50	100	4	1	1	29	3074.4	4115.8	6717.6	0.03	0.44	1.66
50	100	5	1	1	2.6	3020.6	3665.2	3407	0.07	0.42	1.46
50	100	6	1	1	3.4	2925.4	3351	3516.4	0.10	0.56	1.47

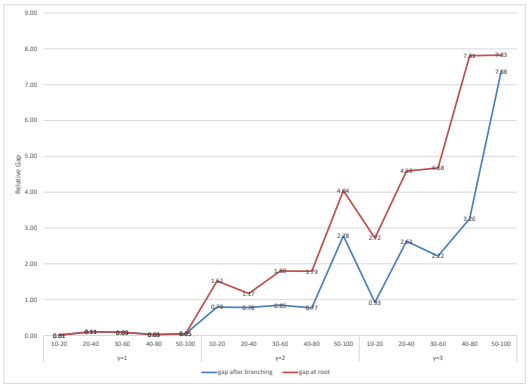
Table 5.5: Results of strategy 3.

Problem			Nodes Explored			Columns Generated			Relative Gap		
$\|J\|$	$\|K\|$	$\|L\|$	$\gamma = 1$	$\gamma = 2$	$\gamma = 3$	$\gamma = 1$	$\gamma = 2$	$\gamma = 3$	$\gamma = 1$	$\gamma = 2$	$\gamma = 3$
10	20	2	1	76.2	159.2	103.2	957.8	1698.2	0.05	0.84	0.93
10	20	3	1	188	479.4	92.4	1228.8	2949.6	0.03	0.75	0.98
10	20	4	1	376.2	816.6	82.4	1768.8	4372.6	0.00	0.92	0.98
10	20	5	1	385	1231.8	72	1857.2	6547.2	0.13	0.88	0.82
10	20	6	1	137	789.6	75.8	668.6	3418.4	0.00	0.63	0.86
20	40	2	1.8	480.4	2063.8	386	24070.6	97064.4	0.04	0.75	0.98
20	40	3	3.8	1331.6	5970.4	360.4	29703	156226.6	0.10	0.83	2.66
20	40	4	2.8	3886.8	12767.8	293.8	44361.2	164487.8	0.00	0.85	3.07
20	40	5	1.2	6805	17463	238.8	43648.8	144214.8	0.15	0.72	3.19
20	40	6	1.6	6970.4	17998.2	221.6	37670.8	107290.2	0.01	0.87	2.64
30	60	2	1.8	112.2	379	902.4	14691.2	66884.6	0.05	0.83	2.28
30	60	3	12.2	453.6	1237.4	1450.4	17526.4	69912	0.00	0.88	2.43
30	60	4	24.8	66.2	2800.6	1493	2185.8	68806	0.02	0.82	2.22
30	60	5	12.6	59.2	4360	777.6	1298.4	67385.6	0.03	0.81	2.09
30	60	6	14.4	6.2	4446.2	683	847.4	31300.8	0.27	0.65	2.02
40	80	2	7	1.4	67.8	3158.4	5736	23541	0.01	0.73	2.45
40	80	3	23	53	252.6	4068.6	5455.8	32681.6	0.00	0.83	12.24
40	80	4	722.8	258	479	31281.8	8845	31174	0.00	0.74	10.96
40	80	5	59.6	422	681.6	4176.2	8324	22956	0.00	0.80	2.52
40	80	6	40.4	277	304.6	2338.4	5186.6	10913.6	0.01	0.88	2.23
50	100	2	8.6	2.2	6.6	6402.4	6067.2	7618	0.00	0.56	13.43
50	100	3	68.6	25.6	49	13846.8	5715	9046.4	13.24	6.14	32.93
50	100	4	274	61.8	80.8	22332.2	6284.8	9278.6	11.67	9.39	24.23
50	100	5	308	99	81.4	19419.6	5910.4	7551.4	11.17	20.08	30.32
50	100	6	392.8	87.6	52.2	19113.4	4506.2	4238.6	11.33	15.33	29.08

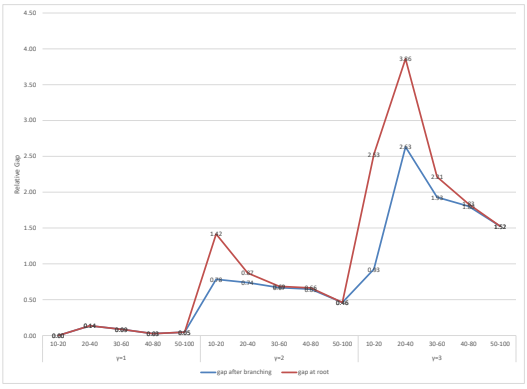
Table 5.6: Results of strategy 4.

Problem			Nodes Explored			Columns Generated			Relative Gap		
$\|J\|$	$\|K\|$	$\|L\|$	$\gamma = 1$	$\gamma = 2$	$\gamma = 3$	$\gamma = 1$	$\gamma = 2$	$\gamma = 3$	$\gamma = 1$	$\gamma = 2$	$\gamma = 3$
10	20	2	1	89.2	215	197.6	946.4	1832.6	0.00	0.80	0.98
10	20	3	1	443	1328.4	174	2364.6	7000	0.00	0.75	0.99
10	20	4	1	1093	2362.4	102.2	4290.8	10372.6	0.00	0.84	0.99
10	20	5	1	1112.8	1582.6	91.2	4408	8457.4	0.00	0.88	0.85
10	20	6	1	151.2	1220.6	87.4	732.8	4765	0.00	0.61	0.88
20	40	2	1	452.8	4401.4	793.6	19538.8	144090.4	0.04	0.71	1.03
20	40	3	1	3759	7232.8	604	57244.4	126976.2	0.20	0.80	3.64
20	40	4	1	3626.4	14135.8	591	27672	109308.6	0.07	0.81	3.50
20	40	5	1	9210	12979.4	479	41839.2	64312.4	0.29	0.73	2.77
20	40	6	1	4345.8	8426	385.8	22808.6	35306	0.12	0.76	2.56
30	60	2	1	126.4	474	1938	14590.4	65748.8	0.01	0.80	2.26
30	60	3	1	538	1769.4	1278	12537.6	66950	0.06	0.69	2.18
30	60	4	1	918	2001.8	1145.4	10757.8	37160.2	0.06	0.74	2.07
30	60	5	1	1	1417.8	1168.8	1192.6	19429	0.11	0.71	1.77
30	60	6	1	1	289.4	1026	947.8	3739.2	0.21	0.57	1.71
40	80	2	1	1	63.4	4126.8	9313	23882.6	0.01	0.54	1.74
40	80	3	1	1	216	2548.4	3381.6	23458	0.02	0.67	1.80
40	80	4	1	1	263.4	1814.6	2638	16401.6	0.04	0.69	2.05
40	80	5	1	13.2	87.4	1971.8	2524.2	5627.6	0.05	0.68	1.86
40	80	6	1	1	25.2	1906	2079	3044.6	0.04	0.72	1.59
50	100	2	1	1	6.2	5661.4	9157.4	10090.4	0.02	0.44	1.65
50	100	3	1	1	18.2	3472	4099.6	7043	0.02	0.42	1.67
50	100	4	1	1	43	2968.2	3990.8	7486	0.03	0.41	1.59
50	100	5	1	1	5.6	2843	3504	3376.6	0.08	0.41	1.59
50	100	6	1	1	4.4	2667.2	2949.6	3318.6	0.11	0.55	1.64

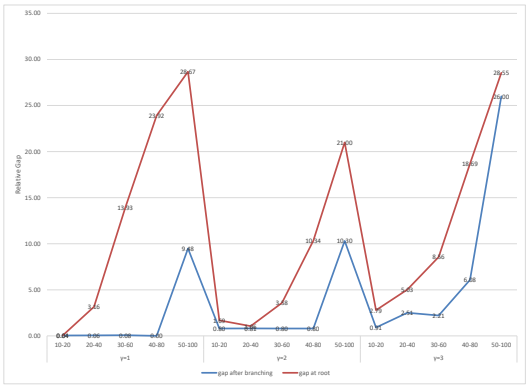
We also analyze the effects of branching on the relative gaps. Figure 5.2 illustrates the average gaps before and after branching for each strategy for different γ's.



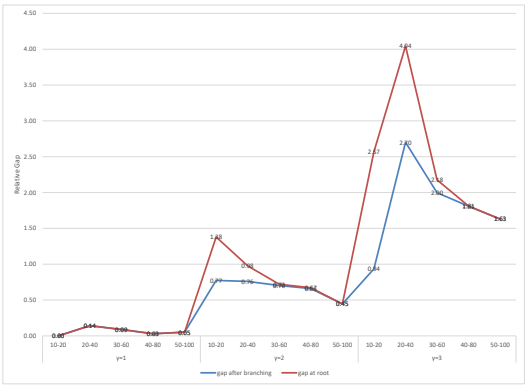
(a) Relative gaps for strategy 1.



(b) Relative gaps for strategy 2.



(c) Relative gaps for strategy 3.



(d) Relative gaps for strategy 4.

Figure 5.2: Average relative gaps before and after branching.

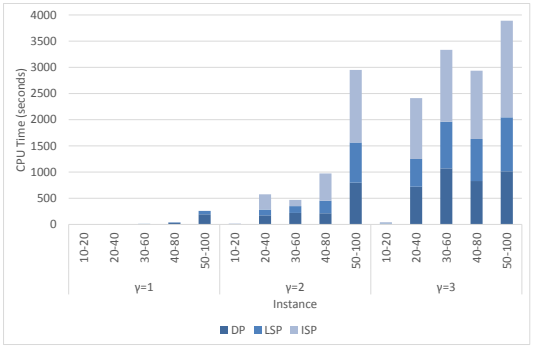
It is clear that strategy 3 is the worst in terms of relative gap obtained. Although strategies 1 and 3 provide the most improvement after branching, strategies 2 and 4 obtain better results. Therefore, we conclude that generating multiple columns (10 columns) results in finding better incumbent solutions. In other words, the reason for the high

improvement after branching for strategies 1 and 3 is the incumbent solutions, not the lower bound. Another important finding is the distribution of CPU time spent. There are three main parts in the column generation algorithm. The first one is the solution of the subproblem which is solved using dynamic programming denoted by DP. The second is the solution of the master problem denoted by LSP. The third is the solution of the binary set partitioning formulation, denoted by ISP. Figure 5.3 displays the CPU time spent on each part of the algorithm for all strategies. Strategies 2 and 4 also outperform other strategies in terms of CPU time. As Figure 5.3 reveals, the solution of [ISP] takes more time than expected. This observation is true regardless of the strategy applied.

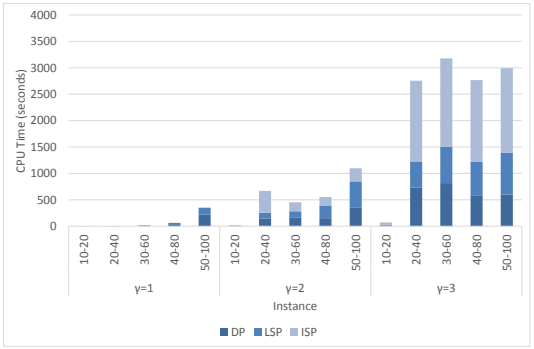
5.8 Comparison Between Order Consolidation Problem and Parallel Machine Scheduling

Being a special case of the order consolidation problem, the proposed solution methodology is applicable to the parallel machine scheduling problem. We set γ to 1 and compare to the branch and price approach of Chen and Powell (1999). Due to the findings in Section 5.7, we apply strategy 2 for both solution methods. We also set the same stopping criteria of 1% relative gap and maximum 3600 seconds for CPU time limit. Average results for both methods are presented in Table 5.7. The average is taken over 35 instances for 2, 3, 4, 5, and 6 induction lines. Detailed results are provided in Appendix B. The first two columns give the number of totes and orders. The remaining columns display the lower bound (LB), upper bound (UB), relative gap, calculated as $100\frac{UB-LB}{LB}$, and solution time in seconds for the method of Chen and Powell (1999) for PMS, and for the proposed branch-and-price for OCP, respectively.

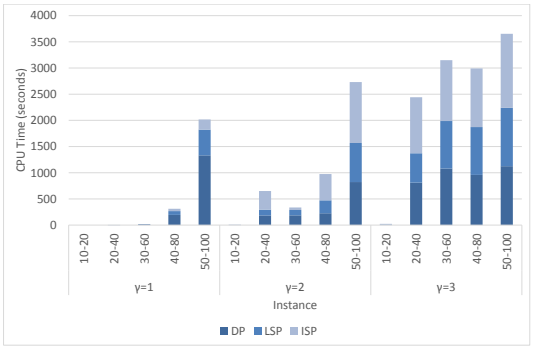
For all instances, both algorithms find an optimal solution within 1%. In 35 instances,



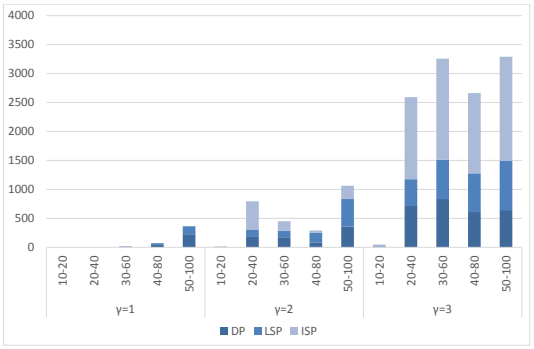
(a) CPU times for strategy 1.



(b) CPU times for strategy 2.



(c) CPU times for strategy 3.



(d) CPU times for strategy 4.

Figure 5.3: Comparison of CPU times.

both methods found the optimal solution. In 112 instances, both methods find the same solution, whereas in 13 instances, only one of the methods obtained the best solution. For those 13 instances, OCP obtained better results in six, Chen and Powell (1999) found better results in seven. Figure 5.4 displays the relative gaps of OCP and PMS for instances where different gaps are found. The instance names are displayed as "$n-|K|-m-$problem". In terms of CPU time, Chen and Powell (1999) is faster than the proposed algorithm. This is expected due to the differences between the respective restricted master problems. Chen

Table 5.7: Comparison between OCP and PMS.

Problem Parameters		PMS				OCP			
n	$\|K\|$	LB	UB	GAP	CPU	LB	UB	GAP	CPU
10	20	1398.28	1398.48	0.0096	0	1398.28	1398.28	0	0.04
20	40	4873.6	4879.16	0.1348	1.56	4873.6	4879.64	0.136	2.08
30	60	10393.24	10400.36	0.0872	11.52	10393.24	10400.28	0.0864	15.76
40	80	18500.84	18505	0.0272	51.4	18500.84	18505	0.0272	61.96
50	100	26285	26295	0.0476	140.44	26285	26295	0.0476	351.8

and Powell (1999) take advantage of the SWPT rule. Figure 5.5 compares the CPU times for instances that required at least 100s for either method.

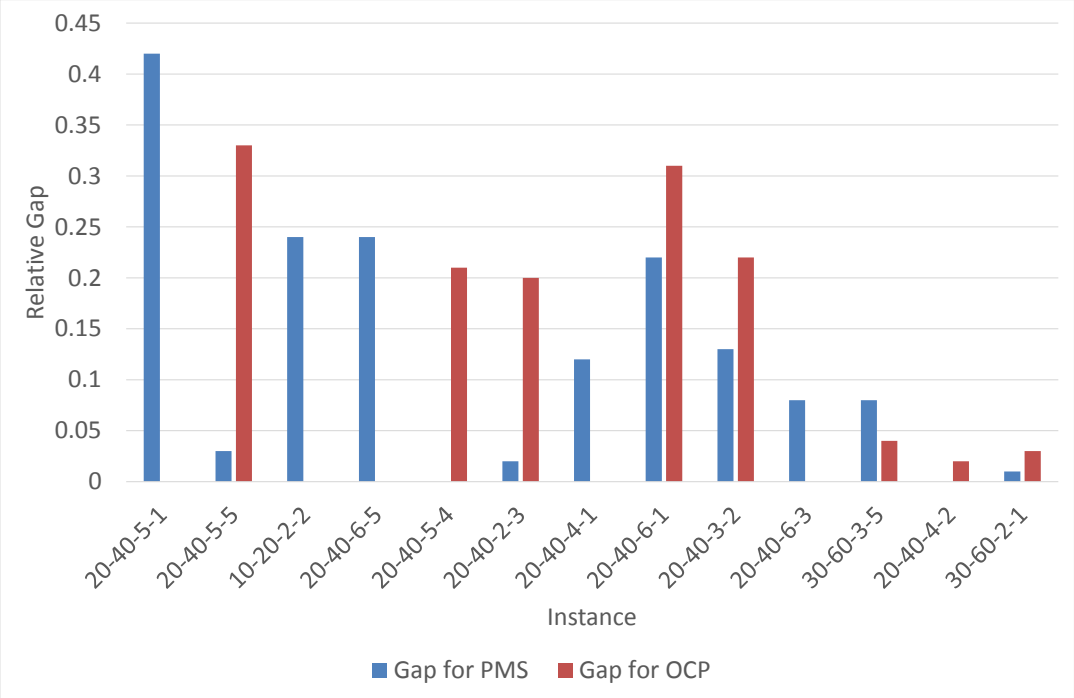


Figure 5.4: Gap comparison for OCP and PMS when different gaps are achieved.

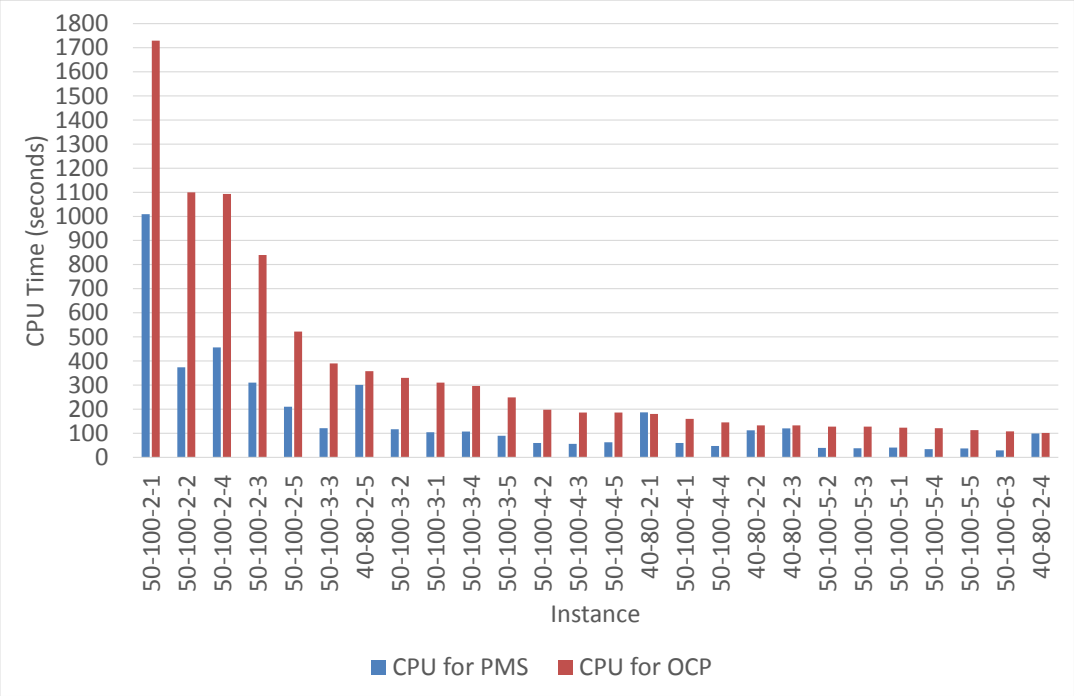


Figure 5.5: CPU time comparison for OCP and PMS on instances which required at least 100s.

5.9 Conclusions

In this chapter, we develop a branch-and-price algorithm for the set partitioning formulation of OCP. Then, we present column generation algorithm that we use to solve each node of branch-and-price. We explain how we use the dynamic programming algorithm introduced in Chen and Powell (1999) in the root node to solve the pricing problem. We introduce a new branching strategy based on a new mathematical model for OCP to determine branching variables and to solve the pricing problem using dynamic programming.

Since the dynamic programming algorithm of Chen and Powell (1999) does not solve the pricing problem after branching, we develop a new dynamic programming algorithm whose complexity is bounded by $O(n^2P)$.

We test four strategies to determine the parameters of branch-and-price algorithm. These strategies determine how the initial columns are obtained and how many columns are generated by the pricing problem. For strategies 1 and 2, we start with columns generated using a construction algorithm in addition to random columns, whereas strategies 3 and 4 start with only random columns. We generate the best 10 columns with the pricing problem for strategies 2 and 4, whereas we generate only the best column for strategies 1 and 3. Strategy 2 is found to be the best strategy based on an extensive computational test. Solutions within 1% gap (0.06% on average) are found in the root node when γ is 1 for strategy 2. When γ is 2, on average, the results are found to be within 0.66%, while the average gap is 1.76% when γ is 3. We also compared a special case where γ is 1. In this case, OCP reduces to a parallel machine scheduling problem with the total weighted completion time objective function. We implemented and compared to the methodology presented in Chen and Powell (1999). We showed that even though the proposed algorithm was designed for a more complex problem, we obtain solutions of similar quality as the algorithm of Chen and Powell (1999). This comparison proves that the proposed algorithm is successful in finding high quality solutions.

Chapter 6

Conclusion

This chapter provides a brief revisit of the analyses and methods proposed in this thesis, followed by a synopsis of our findings and their relevance. We also offer some suggestions for future work directions and their potential benefits.

In this thesis, we use a data analytics approach for an e-warehouse to understand the system in detail and identify areas for improvement. We analyze the system to derive a process diagram for the warehouse, which is entirely deduced from the data provided. We show that the characteristics of waves and orders are related, and that 73 % of the orders contain at most five products whereas 9 % contain more than 10 products. We also present daily operations in terms of numbers of products sold, orders processed, totes used, and waves formed.

In Chapter 2, we review the related literature under two sections. Since this study is conducted for an e-warehouse, in Section 2.1, we explore warehouse management systems. We provide a literature review for order picking and accumulation and sortation systems. We also present an extensive literature review on parallel machine scheduling problems in

Section 2.2, since our analysis suggests a way to improve the order consolidation process, which we define as a parallel machine scheduling problem. We first categorize the problems according to machine type, job type, performance criteria, and solutions methods. We show that there are different job definitions in the literature consequent to differing requirements of various manufacturing environments. We review studies for different job definitions and explain their application areas. From this, we ascertain that the job we define for our study has been considered in only one other study. We also cover problems with the objective of total weighted completion time minimization criterion, because it is the most related criterion to the objective function of the order consolidation problem (OCP). We end our review with an investigation of parallel machine scheduling problems according to the solution method, covering both exact and heuristic approaches. From this literature examination, we find out that there are not many exact solution methods for parallel machine scheduling problems with general job types, other than the classical single job parallel machine scheduling problem.

In Chapter 3, we apply descriptive analytics to draw a picture of the current status of the e-warehousing system. We define the steps (as durations) of the order preparation process and analyze their contribution to whole order preparation time. We first select the waves, in total 17, which are not interrupted with any significant breaks. We calculate processing times of each defined duration and visualize the average times for each wave. On average, an order spends 61 minutes in picking, 26 minutes in the buffer area, 30 minutes in putting, and 19 minutes in waiting for the packing. Since the order consolidation time is defined as the summation of the waiting time in the buffer area and the putting time in the unit sorter, it requires 56 minutes to consolidate an order on average. We defined system variables for the wave processing as follows: number of orders, number of products, average consolidation time, waiting for packing and packing operators. We showed that

the order consolidation process is not significantly correlated to other variables. Therefore, we conclude this chapter by suggesting to determine the tote sequence in which they are diverted from the buffer area to the unit sorter area in order to improve the order consolidation process.

In Chapter 4, we define the order consolidation problem and show that OCP is also expected to improve average cubby usage time per order. We present the mathematical formulation of OCP and propose a simulated annealing (SA) algorithm to solve the problem. To evaluate the performance of the SA algorithm, we consider a special case of OCP where all the totes have the same processing time. We design an experiment to calibrate the parameters of the SA algorithm. We conduct two computational tests. The first one uses industry instances to evaluate the proposed algorithm. Due to the absence of tote contents in the data, we analyze induction operations further. We show that in the data there are anomalies which do not allow determining tote contents. By analyzing induction operations, we develop an algorithm to estimate tote contents. For the sake of reliability, we also analyze the usage of the induction lines and show that during a wave process, not all induction lines are always available. Since we compare the sequences obtained by the SA algorithm to industry instances, we define two scenarios, one of which considers that all lines are available at all times whereas the other takes the breaks into consideration. We show that the improvements over the system are significant. For the first scenario, at least 6.74% improvement is achieved with the suggested schedules (sequences) whereas the average improvement is 28.77%. For the second scenario, these values are 5.96% and 19.9%, respectively. We also test the performance of the SA algorithm on random instances for the special case of OCP whose mathematical model is relatively easier. It is important to emphasize that for the instances wherein the mathematical model obtained optimal solution within the time limit, the average relative gap of the SA is less than 0.01%.

In Chapter 5, we propose a branch-and-price algorithm for the OCP. We first present a set partitioning formulation of the problem. Then, since generating all possible columns is not practical, we explain how we generate columns when they are needed using a column generation algorithm. We use the dynamic programming algorithm introduced in Chen and Powell (1999) in the root node to generate promising and necessary columns. We apply a new branch strategy and demonstrate why, in this problem, it is superior to the two-branch strategy which is common in the literature. We propose another mathematical model to determine branching variables and explain why it is needed for the new branch strategy. We also develop a new dynamic programming algorithm whose complexity is bounded by $O(n^2P)$ for the child nodes because the former algorithm does not solve the pricing problem to optimality after branching. To determine the parameters of branch-and-price algorithm, we define four strategies which indicate how the initial columns are obtained and how many columns are generated by the pricing problem. Strategies 1 and 2 start with columns generated by a construction algorithm in addition to random columns, whereas strategies 3 and 4 start with only random columns. The best 10 columns are generated with the pricing problem for strategies 2 and 4, whereas strategies 1 and 3 generate only the best column. Strategy 2 was found to be the best strategy based on an extensive computational test. The results for strategy 2 show that the proposed algorithm found solutions within 1% gap (0.06% in average) in the root node when γ is 1. When γ is 2, on average, the results are found to be within 0.66%, while the average gap is 1.76% when γ is 3. We also compared a special case of OCP where γ is 1 and which can be defined as a parallel machine scheduling problem with the total weighted completion time objective function. We implemented and compared to the branch-and-price algorithm presented in Chen and Powell (1999). We showed that even though the proposed algorithm was designed for a more complex problem, we obtained solutions of similar quality as the algorithm of Chen and Powell (1999); however, we spent on average 80.32 seconds, while Chen and Powell

(1999) spent 40.98 seconds. Higher average CPU time is expected due to the complexity of the master problem of the OCP compared to the PMS problem. This comparison proves that the proposed algorithm is successful in finding high quality solutions.

This thesis presents the application of data anlaytics to data from an e-warehouse. Employing descriptive analytics to understand the working process of the warehouse and to identify areas for improvement, we found that the whole process can be improved by changing the sequence in which totes are diverted from the buffer area to the unit sorter. Specifically, we introduced a problem to determine the tote sequence and propose several solution methodologies to obtain good quality solutions. Evaluation of our results shows that the proposed methods are not only useful but also practical. For the warehouse operations, the routing sorter is also important for warehouse efficiency. We did not focus on the routing sorter operations due to a lack of data. Nevertheless, investigating the working process of such a sorter remains a promising research area. In the routing sorter, scheduling of waves or the tote diverting mechanism are also promising directions. Lastly, processing times of totes in the OCP are not deterministic; however, in this study we considered the deterministic processing times due to low variance on item induction times. For a different warehouse where variation for product induction times is high, the OCP can be studied under stochastic processing times.

References

Anagnostopoulos, G. C., Rabadi, G., 2002. A simulated annealing algorithm for the unrelated parallel machine scheduling problem. In: Automation Congress, 2002 Proceedings of the 5th Biannual World. Vol. 14. IEEE, pp. 115–120.

Balakrishnan, N., Kanet, J. J., Sridharan, V., 1999. Early/tardy scheduling with sequence dependent setups on uniform parallel machines. Computers and Operations Research 26 (2), 127–141.

Belouadah, H., Potts, C. N., 1994. Scheduling identical parallel machines to minimize total weighted completion time. Discrete Applied Mathematics 48 (3), 201–218.

Boysen, N., Stephan, K., Weidinger, F., 2018. Manual order consolidation with put walls: the batched order bin sequencing problem. EURO Journal on Transportation and Logistics, 1–25.

Bozer, Y. A., Quiroz, M. A., Sharp, G. P., 1988. An evaluation of alternative control strategies and design issues for automated order accumulation and sortation systems. Material flow 4 (4), 265–282.

Bozer, Y. A., Sharp, G. P., 1985. An empirical evaluation of a general purpose automated

order accumulation and sortation system used in batch picking. Material Flow 2 (2-3), 111–131.

Bülbül, K., Şen, H., 2017. An exact extended formulation for the unrelated parallel machine total weighted completion time problem. Journal of Scheduling 20 (4), 373–389.

Chan, L. M. A., Muriel, A., Simchi-Levi, D., 1998. Parallel machine scheduling, linear programming, and parameter list scheduling heuristics. Operations Research 46 (5), 729–741.

Chen, F., Wang, H., Qi, C., Xie, Y., 2013. An ant colony optimization routing algorithm for two order pickers with congestion consideration. Computers & Industrial Engineering 66 (1), 77–85.

Chen, J.-F., 2015. Unrelated parallel-machine scheduling to minimize total weighted completion time. Journal of Intelligent Manufacturing 26 (6), 1099–1112.

Chen, Z.-L., Powell, W. B., 1999. Solving parallel machine scheduling problems by column generation. INFORMS Journal on Computing 11 (1), 78–94.

De, P., Ghosh, J. B., Wells, C. E., 1994. Due-date assignment and early/tardy scheduling on identical parallel machines. Naval Research Logistics (NRL) 41 (1), 17–32.

De Koster, R., Le-Duc, T., Roodbergen, K. J., 2007. Design and control of warehouse order picking: A literature review. European Journal of Operational Research 182 (2), 481–501.

Erramilli, V., Mason, S. J., 2006. Multiple orders per job compatible batch scheduling. IEEE Transactions on Electronics Packaging Manufacturing 29 (4), 285–296.

Fanjul-Peyro, L., Ruiz, R., 2010. Iterated greedy local search methods for unrelated parallel machine scheduling. European Journal of Operational Research 207 (1), 55–69.

Gademann, A., Van Den Berg, J. P., Van Der Hoff, H. H., 2001. An order batching algorithm for wave picking in a parallel-aisle warehouse. IIE transactions 33 (5), 385–398.

Grosse, E., Glock, C., Ballester-Ripoll, R., 2014. A simulated annealing approach for the joint order batching and order picker routing problem with weight restrictions. Tech. rep., Darmstadt Technical University, Department of Business Administration, Economics and Law, Institute for Business Studies (BWL).

Gu, J., Goetschalckx, M., McGinnis, L. F., 2007. Research on warehouse operation: A comprehensive review. European journal of operational research 177 (1), 1–21.

Henn, S., Koch, S., Wäscher, G., 2012. Order batching in order picking warehouses: a survey of solution approaches. In: Warehousing in the Global Supply Chain. Springer, pp. 105–137.

Hong, S., Johnson, A. L., Peters, B. A., 2012. Batch picking in narrow-aisle order picking systems with consideration for picker blocking. European Journal of Operational Research 221 (3), 557–570.

Hsieh, L.-F., Huang, Y.-C., 2011. New batch construction heuristics to optimise the performance of order picking systems. International Journal of Production Economics 131 (2), 618–630.

Hwang, H., Oh, Y., Lee, Y., 2004. An evaluation of routing policies for order-picking operations in low-level picker-to-part system. International Journal of Production Research 42 (18), 3873–3889.

Jampani, J., Mason, S. J., 2008. Column generation heuristics for multiple machine, multiple orders per job scheduling problems. Annals of Operations Research 159 (1), 261–273.

Jia, J., Mason, S. J., 2009. Semiconductor manufacturing scheduling of jobs containing multiple orders on identical parallel machines. International Journal of Production Research 47 (10), 2565–2585.

Johnson, M. E., 1997. The impact of sorting strategies on automated sortation system performance. IIE transactions 30 (1), 67–77.

Juesheng, F., 1998. A genetic algorithm method for identical parallel machine scheduling problem. Application of Statistics and Management 6.

Kaplan, S., Rabadi, G., 2013. Simulated annealing and metaheuristic for randomized priority search algorithms for the aerial refuelling parallel machine scheduling problem with due date-to-deadline windows and release times. Engineering Optimization 45 (1), 67–87.

Kim, D.-W., Kim, K.-H., Jang, W., Chen, F. F., 2002. Unrelated parallel machine scheduling with setup times using simulated annealing. Robotics and Computer-Integrated Manufacturing 18 (3-4), 223–231.

Kirkpatrick, S., Gelatt, C. D., Vecchi, M. P., et al., 1983. Optimization by simulated annealing. science 220 (4598), 671–680.

Kowalczyk, D., Leus, R., 2017. An exact algorithm for parallel machine scheduling with conflicts. Journal of Scheduling 20 (4), 355–372.

Lee, J.-H., Yu, J.-M., Lee, D.-H., 2013. A tabu search algorithm for unrelated parallel machine scheduling with sequence-and machine-dependent setups: minimizing total tardiness. The International Journal of Advanced Manufacturing Technology 69 (9-12), 2081–2089.

Li, K., Zhang, H.-J., Cheng, B.-Y., Pardalos, P. M., 2018. Uniform parallel machine scheduling problems with fixed machine cost. Optimization Letters 12 (1), 73–86.

Li, K., Zhang, X., Leung, J. Y.-T., Yang, S.-L., 2016. Parallel machine scheduling problems in green manufacturing industry. Journal of Manufacturing Systems 38, 98–106.

Liaw, C.-F., 2016. A branch-and-bound algorithm for identical parallel machine total tardiness scheduling problem with preemption. Journal of Industrial and Production Engineering 33 (6), 426–434.

Lin, B., Jeng, A., 2004. Parallel-machine batch scheduling to minimize the maximum lateness and the number of tardy jobs. International Journal of Production Economics 91 (2), 121–134.

Lin, S.-W., Ying, K.-C., 2015. A multi-point simulated annealing heuristic for solving multiple objective unrelated parallel machine scheduling problems. International Journal of Production Research 53 (4), 1065–1076.

Liu, C.-H., 2010. A genetic algorithm based approach for scheduling of jobs containing multiple orders in a three-machine flowshop. International Journal of Production Research 48 (15), 4379–4396.

Liu, M., Wu, C., Yin, W.-J., 2001. Solving identical parallel machine production line scheduling problem with special procedure constraint by genetic algorithm. Acta Automatica Sinica 27 (3), 381–386.

Liu, Z.-x., Wang, S.-m., 2006. Research on parallel machines scheduling problem based on particle swarm optimization algorithm. Computer Integrated Manufacturing Systems-Beijing- 12 (2), 183.

Logendran, R., McDonell, B., Smucker, B., 2007. Scheduling unrelated parallel machines with sequence-dependent setups. Computers & Operations Research 34 (11), 3420–3438.

Logendran, R., Subur, F., 2004. Unrelated parallel machine scheduling with job splitting. IIE Transactions 36 (4), 359–372.

Mason, S. J., Chen, J.-S., 2010. Scheduling multiple orders per job in a single machine to minimize total completion time. European Journal of Operational Research 207 (1), 70–77.

Mason, S. J., Qu, P., Kutanoglu, E., Fowler, J., 2004. The single machine multiple orders per job scheduling problem. IIE Transactions.

Mathirajan, M., Sivakumar, A. I., 2006. A literature review, classification and simple meta-analysis on scheduling of batch processors in semiconductor. The International Journal of Advanced Manufacturing Technology 29 (9-10), 990–1001.

McNaughton, R., 1959. Scheduling with deadlines and loss functions. Management Science 6 (1), 1–12.

Meller, R. D., 1997. Optimal order-to-lane assignments in an order accumulation/sortation system. IIE transactions 29 (4), 293–301.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., Teller, E., 1953. Equation of state calculations by fast computing machines. The journal of chemical physics 21 (6), 1087–1092.

Min, L., Cheng, W., 1999. A genetic algorithm for minimizing the makespan in the case of scheduling identical parallel machines. Artificial Intelligence in Engineering 13 (4), 399–403.

Mokotoff, E., 2004. An exact algorithm for the identical parallel machine scheduling problem. European Journal of Operational Research 152 (3), 758–769.

Mokotoff, E., Chrétienne, P., 2002. A cutting plane algorithm for the unrelated parallel machine scheduling problem. European Journal of Operational Research 141 (3), 515–525.

Mönch, L., Balasubramanian, H., Fowler, J. W., Pfund, M. E., 2005. Heuristic scheduling of jobs on parallel batch machines with incompatible job families and unequal ready times. Computers & Operations Research 32 (11), 2731–2750.

Pan, J. C.-H., Wu, M.-H., 2012. Throughput analysis for order picking system with multiple pickers and aisle congestion considerations. Computers & Operations Research 39 (7), 1661–1672.

Pessoa, A., Uchoa, E., de Aragão, M. P., Rodrigues, R., 2010. Exact algorithm over an arc-time-indexed formulation for parallel machine scheduling problems. Mathematical Programming Computation 2 (3-4), 259–290.

Petersen, C. G., Aase, G., 2004. A comparison of picking, storage, and routing policies in manual order picking. International Journal of Production Economics 92 (1), 11–19.

Qu, P., Mason, S. J., 2005. Metaheuristic scheduling of 300-mm lots containing multiple orders. IEEE Transactions on Semiconductor Manufacturing 18 (4), 633–643.

Radhakrishnan, S., Ventura, J. A., 2000. Simulated annealing for parallel machine scheduling with earliness-tardiness penalties and sequence-dependent set-up times. International Journal of Production Research 38 (10), 2233–2252.

Ramezani, P., Rabiee, M., Jolai, F., 2015. No-wait flexible flowshop with uniform parallel machines and sequence-dependent setup time: a hybrid meta-heuristic approach. Journal of Intelligent Manufacturing 26 (4), 731–744.

Ratliff, H. D., Rosenthal, A. S., 1983. Order-picking in a rectangular warehouse: a solvable case of the traveling salesman problem. Operations Research 31 (3), 507–521.

Roodbergen, K. J., Koster, R., 2001. Routing methods for warehouses with multiple cross aisles. International Journal of Production Research 39 (9), 1865–1883.

Russell, M. L., Meller, R. D., 2003. Cost and throughput modeling of manual and automated order fulfillment systems. IIE Transactions 35 (7), 589–603.

Schutten, J., Leussink, R., 1996. Parallel machine scheduling with release dates, due dates and family setup times. International journal of production economics 46, 119–125.

Serafini, P., 1996. Scheduling jobs on several machines with the job splitting property. Operations Research 44 (4), 617–628.

Shim, S.-O., Kim, Y.-D., 2008. A branch and bound algorithm for an identical parallel machine scheduling problem with a job splitting property. Computers & Operations Research 35 (3), 863–875.

Smith, W. E., 1956. Various optimizers for single-stage production. Naval Research Logistics Quarterly 3 (1-2), 59–66.

Tavakkoli-Moghaddam, R., Taheri, F., Bazzazi, M., Izadi, M., Sassani, F., 2009. Design of a genetic algorithm for bi-objective unrelated parallel machines scheduling with sequence-dependent setup times and precedence constraints. Computers & Operations Research 36 (12), 3224–3230.

Theys, C., Bräysy, O., Dullaert, W., Raa, B., 2010. Using a tsp heuristic for routing order pickers in warehouses. European Journal of Operational Research 200 (3), 755–763.

Uzsoy, R., 1995. Scheduling batch processing machines with incompatible job families. International Journal of Production Research 33 (10), 2685–2708.

Vallada, E., Ruiz, R., 2011. A genetic algorithm for the unrelated parallel machine scheduling problem with sequence dependent setup times. European Journal of Operational Research 211 (3), 612–622.

van Den Akker, J. M., Hoogeveen, J. A., van de Velde, S. L., 1999. Parallel machine scheduling by column generation. Operations Research 47 (6), 862–872.

Van den Berg, J. P., Zijm, W., 1999. Models for warehouse management: Classification and examples. International Journal of Production Economics 59 (1), 519–528.

Villa, F., Vallada, E., Fanjul-Peyro, L., 2018. Heuristic algorithms for the unrelated parallel machine scheduling problem with one scarce additional resource. Expert Systems with Applications 93, 28–38.

Wang, W.-L., Wang, H.-Y., Zhao, Y.-W., Zhang, L.-P., Xu, X.-L., 2013. Parallel machine scheduling with splitting jobs by a hybrid differential evolution algorithm. Computers & Operations Research 40 (5), 1196–1206.

Weng, M. X., Lu, J., Ren, H., 2001. Unrelated parallel machine scheduling with setup consideration and a total weighted completion time objective. International journal of production economics 70 (3), 215–226.

Xing, W., Zhang, J., 2000. Parallel machine scheduling with splitting jobs. Discrete Applied Mathematics 103 (1), 259–269.

Yeh, W.-C., Chuang, M.-C., Lee, W.-C., 2015. Uniform parallel machine scheduling with resource consumption constraint. Applied Mathematical Modelling 39 (8), 2131–2138.

Yin, W.-j., Liu, M., Wu, C., 2001. A new genetic algorithm for parallel machine scheduling with process constraint. Acta Electronica Sinica 29 (11), 1482–1485.

APPENDICES

Appendix A

Characteristics and relationships between totes, orders and items

Table A.1: The Order-Tote relationship.

	Orders per Tote			Totes per Order		
Wave ID	**max**	**mean**	**std**	**max**	**mean**	**std**
W-353-1243	30	11.54	8.39	19	3.27	2.45
W-374-1384	30	17.27	11.1	27	4.66	3.36
W-420-1374	30	14.51	9.42	24	4.43	2.98
W-368-1373	30	14.71	10.47	23	3.94	2.73
W-336-1367	30	16.07	9.66	29	3.95	2.81
W-257-851	30	11.12	8.76	26	3.36	3.18
W-253-932	30	15.13	10.76	14	4.1	1.7
W-377-989	30	11.81	9.66	25	4.5	3.25
W-169-949	30	22.54	10.09	19	4.01	1.9
W-121-267	30	9.37	7.22	20	4.24	4.21

Table A.2: The product-tote relationship and the tote processing times.

	Items per Tote			Tote Processing Times		
Wave ID	**max**	**mean**	**std**	**max**	**mean**	**std**
W-353-1243	30	11.9	8.76	60	24	14
W-374-1384	30	17.9	11.76	60	33	19
W-420-1374	30	14.94	9.75	60	27	16
W-368-1373	30	18.34	10.97	60	28	16
W-336-1367	30	16.66	10.1	60	30	16
W-257-851	30	11.58	9.26	60	25	16
W-253-932	30	15.51	11.1	60	29	18
W-377-989	30	12.3	10.2	60	27	18
W-169-949	30	23.2	10.4	60	38	15
W-121-267	30	10.2	8.3	60	22	14

Appendix B

Detailed Comparison between OCP and PMS

Table B.1: Comparison between PMs and OCP for 10 totes and 20 orders

Problem		PMS				OCP			
Line	**Instance**	**LB**	**UB**	**Gap**	**CPU**	**LB**	**UB**	**Gap**	**CPU**
2	1	1922	1922	0	0	1922	1922	0	0
3	1	1451	1451	0	0	1451	1451	0	0
4	1	1226	1226	0	0	1226	1226	0	0
5	1	1091	1091	0	0	1091	1091	0	0
6	1	1019	1019	0	0	1019	1019	0	0
2	2	2075	2080	0.24	0	2075	2075	0	0
3	2	1569	1569	0	0	1569	1569	0	0
4	2	1326	1326	0	0	1326	1326	0	0
5	2	1187	1187	0	0	1187	1187	0	0
6	2	1093	1093	0	0	1093	1093	0	0
2	3	1775	1775	0	0	1775	1775	0	0
3	3	1351	1351	0	0	1351	1351	0	0
4	3	1154	1154	0	0	1154	1154	0	0
5	3	1036	1036	0	0	1036	1036	0	0
6	3	964	964	0	0	964	964	0	0
2	4	2333	2333	0	0	2333	2333	0	1
3	4	1771	1771	0	0	1771	1771	0	0
4	4	1485	1485	0	0	1485	1485	0	0
5	4	1320	1320	0	0	1320	1320	0	0
6	4	1237	1237	0	0	1237	1237	0	0
2	5	1889	1889	0	0	1889	1889	0	0
3	5	1422	1422	0	0	1422	1422	0	0
4	5	1202	1202	0	0	1202	1202	0	0
5	5	1068	1068	0	0	1068	1068	0	0
6	5	991	991	0	0	991	991	0	0

Table B.2: Comparison between PMs and OCP for 20 totes and 40 orders

Problem		PMS				OCP			
Line	**Instance**	**LB**	**UB**	**Gap**	**CPU**	**LB**	**UB**	**Gap**	**CPU**
2	1	7318	7319	0.01	5	7318	7319	0.01	6
3	1	5239	5251	0.23	1	5239	5251	0.23	3
4	1	4217	4222	0.12	1	4217	4217	0	2
5	1	3599	3614	0.42	0	3599	3599	0	0
6	1	3197	3204	0.22	0	3197	3207	0.31	0
2	2	7474	7474	0	3	7474	7474	0	5
3	2	5351	5358	0.13	1	5351	5363	0.22	2
4	2	4304	4304	0	1	4304	4305	0.02	1
5	2	3676	3699	0.63	0	3676	3699	0.63	0
6	2	3271	3279	0.24	0	3271	3279	0.24	0
2	3	8614	8616	0.02	6	8614	8631	0.2	6
3	3	6136	6141	0.08	2	6136	6141	0.08	2
4	3	4895	4898	0.06	1	4895	4898	0.06	2
5	3	4168	4180	0.29	0	4168	4180	0.29	1
6	3	3676	3679	0.08	0	3676	3676	0	0
2	4	7673	7673	0	7	7673	7673	0	7
3	4	5494	5506	0.22	2	5494	5506	0.22	3
4	4	4409	4409	0	2	4409	4409	0	2
5	4	3774	3774	0	0	3774	3782	0.21	1
6	4	3358	3358	0	0	3358	3358	0	0
2	5	6855	6855	0	5	6855	6855	0	6
3	5	4891	4902	0.22	1	4891	4902	0.22	2
4	5	3928	3933	0.13	1	3928	3933	0.13	1
5	5	3352	3353	0.03	0	3352	3363	0.33	0
6	5	2971	2978	0.24	0	2971	2971	0	0

Table B.3: Comparison between PMs and OCP for 30 totes and 60 orders

Problem		PMS				OCP			
Line	**Instance**	**LB**	**UB**	**Gap**	**CPU**	**LB**	**UB**	**Gap**	**CPU**
2	1	17484	17486	0.01	23	17484	17489	0.03	32
3	1	12222	12232	0.08	12	12222	12232	0.08	13
4	1	9599	9602	0.03	7	9599	9602	0.03	11
5	1	8033	8042	0.11	4	8033	8042	0.11	6
6	1	6992	7014	0.31	3	6992	7014	0.31	4
2	2	14996	14997	0.01	45	14996	14997	0.01	52
3	2	10492	10493	0.01	13	10492	10493	0.01	20
4	2	8252	8256	0.05	5	8252	8256	0.05	7
5	2	6904	6910	0.09	4	6904	6910	0.09	6
6	2	6024	6033	0.15	3	6024	6033	0.15	3
2	3	17622	17623	0.01	33	17622	17623	0.01	38
3	3	12290	12294	0.03	9	12290	12294	0.03	12
4	3	9621	9624	0.03	6	9621	9624	0.03	11
5	3	8025	8037	0.15	4	8025	8037	0.15	6
6	3	6970	6988	0.26	3	6970	6988	0.26	3
2	4	16314	16314	0	34	16314	16314	0	51
3	4	11390	11397	0.06	9	11390	11397	0.06	13
4	4	8947	8958	0.12	5	8947	8958	0.12	7
5	4	7492	7492	0	4	7492	7492	0	7
6	4	6514	6531	0.26	3	6514	6531	0.26	4
2	5	17298	17301	0.02	34	17298	17301	0.02	53
3	5	12075	12085	0.08	13	12075	12080	0.04	17
4	5	9478	9482	0.04	5	9478	9482	0.04	7
5	5	7916	7933	0.21	4	7916	7933	0.21	6
6	5	6881	6885	0.06	3	6881	6885	0.06	5

Table B.4: Comparison between PMs and OCP for 40 totes and 80 orders

Problem		PMS				OCP			
Line	**Instance**	**LB**	**UB**	**Gap**	**CPU**	**LB**	**UB**	**Gap**	**CPU**
2	1	27829	27830	0	187	27829	27830	0	180
3	1	19262	19268	0.03	37	19262	19268	0.03	53
4	1	14993	14994	0.01	21	14993	14994	0.01	25
5	1	12431	12449	0.14	14	12431	12449	0.14	20
6	1	10738	10741	0.03	11	10738	10741	0.03	17
2	2	28171	28173	0.01	112	28171	28173	0.01	133
3	2	19480	19481	0.01	38	19480	19481	0.01	53
4	2	15145	15150	0.03	23	15145	15150	0.03	33
5	2	12549	12554	0.04	14	12549	12554	0.04	20
6	2	10817	10820	0.03	10	10817	10820	0.03	17
2	3	31567	31575	0.03	120	31567	31575	0.03	133
3	3	21779	21783	0.02	66	21779	21783	0.02	76
4	3	16886	16896	0.06	22	16886	16896	0.06	29
5	3	13962	13965	0.02	15	13962	13965	0.02	21
6	3	12015	12023	0.07	9	12015	12023	0.07	17
2	4	30466	30471	0.02	99	30466	30471	0.02	101
3	4	21031	21035	0.02	48	21031	21035	0.02	66
4	4	16315	16319	0.02	21	16315	16319	0.02	32
5	4	13504	13508	0.03	12	13504	13508	0.03	20
6	4	11632	11636	0.03	9	11632	11636	0.03	14
2	5	33505	33505	0	301	33505	33505	0	358
3	5	23084	23086	0.01	50	23084	23086	0.01	64
4	5	17891	17894	0.02	19	17891	17894	0.02	26
5	5	14773	14773	0	15	14773	14773	0	20
6	5	12696	12696	0	12	12696	12696	0	21

Table B.5: Comparison between PMs and OCP for 50 totes and 100 orders

Problem		PMS				OCP			
Line	**Instance**	**LB**	**UB**	**Gap**	**CPU**	**LB**	**UB**	**Gap**	**CPU**
2	1	47256	47268	0.03	1009	47256	47268	0.03	1729
3	1	32435	32451	0.05	104	32435	32451	0.05	310
4	1	25031	25035	0.02	60	25031	25035	0.02	160
5	1	20594	20621	0.13	41	20594	20621	0.13	123
6	1	17641	17673	0.18	29	17641	17673	0.18	90
2	2	45855	45862	0.02	374	45855	45862	0.02	1100
3	2	31450	31453	0.01	117	31450	31453	0.01	330
4	2	24257	24265	0.03	60	24257	24265	0.03	198
5	2	19951	19956	0.03	39	19951	19956	0.03	128
6	2	17090	17097	0.04	29	17090	17097	0.04	88
2	3	42591	42598	0.02	310	42591	42598	0.02	840
3	3	29244	29246	0.01	121	29244	29246	0.01	390
4	3	22583	22585	0.01	56	22583	22585	0.01	186
5	3	18583	18597	0.08	38	18583	18597	0.08	128
6	3	15926	15933	0.04	29	15926	15933	0.04	108
2	4	39466	39474	0.02	456	39466	39474	0.02	1093
3	4	27091	27093	0.01	107	27091	27093	0.01	296
4	4	20911	20917	0.03	47	20911	20917	0.03	145
5	4	17211	17225	0.08	34	17211	17225	0.08	121
6	4	14750	14766	0.11	27	14750	14766	0.11	85
2	5	41993	42003	0.02	210	41993	42003	0.02	522
3	5	28848	28853	0.02	90	28848	28853	0.02	249
4	5	22275	22284	0.04	63	22275	22284	0.04	186
5	5	18350	18361	0.06	37	18350	18361	0.06	113
6	5	15743	15759	0.1	24	15743	15759	0.1	77