Optimizing Order Consolidation with Simulation Optimization

by

Daren Zhou

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Management Sciences

Waterloo, Ontario, Canada, 2017

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

In this work we study the order consolidation process of an e-retailing warehouse using real data. Order consolidation is a process that follows the picking operation in a warehouse. After stock keeping units (SKUs) of an order are picked into possibly multiple totes, they are sorted and the SKUs are consolidated back into orders to be packed and shipped to customers. There are two main operations in order consolidation: wave sortation and putting. Wave sortation involves tote sequencing which is a major driver of how smooth the system runs. First, totes should be sequenced so that the work load is balanced. Second, totes that carry SKUs of the same order should be close in sequence in order to reduce order processing times, and to reduce resource utilization. At the same time, orders should be assigned to operators for putting in order to balance operator workload and to avoid recirculation. The assignment and sequencing of totes, when assuming known and constant tote induction times and when the goal is total completion time, is a problem similar to parallel machine scheduling. However, the goal from order consolidation is minimizing order, as opposed to tote, consolidation times. Order consolidation also depends on the putting operation. Moreover, processing times are uncertain and may vary because of different sizes and weights of SKUs and because of variations in manual and automatic processes. In order to reflect a realistic system performance accurately, we need to take randomness into account when evaluating measures like order completion time, resource utilization, and congestion. In this thesis, we first build a simulation model for the consolidation process using a given tote assignment and sequencing. We use the empirical distributions derived from the data to run the simulation. Second, we develop a simulated annealing heuristic

and apply it to solve the deterministic tote sequencing problem with the goal of minimizing order completion time or order processing time. Third, we develop a stochastic simulated annealing algorithm to optimize the whole consolidation process. The algorithm decides on tote assignment and sequencing and evaluates the sequences using the simulation model. We experimented with optimizing order completion times and SKU wait times for putting. Our approach is then a simulation optimization to optimize the consolidation process using different metrics. We report on numerical tests using real instances.

Acknowledgments

I would like to thank all the people who made this possible.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

E-commerce is booming and is predicted to grow significantly in the coming decades. Retail e-commerce sales worldwide is predicted to grow to 4,058 billion US dollars in 2020 ([Statista, 2017]). E-retailers, like Amazon, Walmart and JingDong, are operating on very large scales. For example, the total operation expense of Amazon during fourth quarter in 2016 was 42.48 billion ([Amazon, 2017]). An e-retailer receives thousands of online orders daily. Such orders have the following features: low volume, high variety and short lead time. Low volume means that there are only a few items required by an order. High variety means that different orders request very different products, consequently e-retailers carry many categories of products. Short lead time means that an e-retailer has to fulfill an order in a short time, once placed. For example, JingDong uses next day delivery and three-hour delivery across cities in China ([Yu et al., 2016]). In order to achieve fast order fulfillment, e-retailers operate large warehouses strategically located and equipped with sophisticated warehouse control systems and logistics software. Such warehouses are

referred to as e-warehouses.

An e-warehouse is a major component of the logistic system of an e-retailer. It performs the following operations to fulfill online orders: receiving, picking, consolidation, packing and shipping. Receiving refers to receiving online orders, grouping many orders together into waves to be processed simultaneously, and preparing pick lists. A pick list is a list of SKUs (stock keeping units), generally stored in close proximity, to be picked together in one or more containers called totes. Picking refers to picking SKUs in pick lists from short term storage area. Generally, pick lists, and consequently totes, contain SKUs from multiple orders. SKUs are then put back into orders using a process called order consolidation. Once all of the SKUs of an order are consolidated, the order is packed and shipped to the customer. E-warehouses use a wide range of technologies from manual to fully automated systems to perform these operations. In this work we study the operation of an e-warehouse. It is part of the industrial project "Adding Analytics to Warehouse Logistics". This project is a collaboration with Dematic, a global leader in warehouse control systems and supply chain solutions. The project is funded by Dematic and Ontario Centres of Excellence (OCE)'s Voucher for Innovation and Productivity (VIP).

In this work, we study the order consolidation process of an e-warehousing system using data provided by Dematic. Order consolidation is a process that follows the picking operation. After items of an order are picked into possibly multiple totes, they are consolidated again to be packed and shipped. An order consists a list of items, and the items are picked in different totes. Totes are then selected and items are consolidated into orders. There are two main operations in the consolidation process under study, wave sortation and putting. Recall that a wave is a set of orders processed together through the warehouse. First,

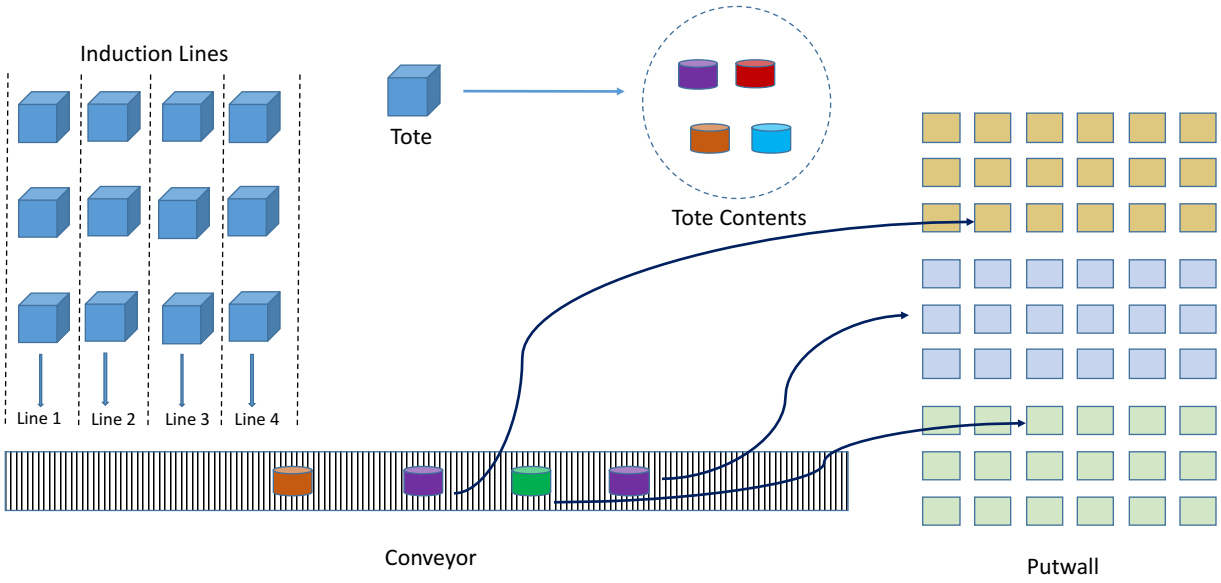

Figure 1.1: Order Consolidation System

SKUs are picked into totes based on pick lists. Totes are then collected into a buffer area before being diverted to the sortation system. In sortation, totes are directed to induction stations where the SKUs are emptied (inducted) onto a conveyor leading to a putting area. The putting area is composed of several putwalls, and a putwall is a wall of cubbies. A cubby is used to consolidate the SKUs or an order. Once all SKUs of an order are placed in the assigned cubby, the order is ready for packing and shipping. In the system under study, there are four induction lines, two of which are automatic and the other two are manual. At the putwall, putting is done manually. An operator is guided by a put-to-light system, picks up SKUs on the conveyor and places them in the pre-assigned cubby. Figure 1.1 shows the main components of an order consolidation system.

The assignment of totes to induction lines and the sequence in which they are inducted is a major driver of how smooth the system runs. First, totes should be assigned to induction lines so that the work load is balanced. Second, totes that carry SKUs of the same order should be inducted as close as possible in order to reduce order processing times, and to reduce cubby utilization. At the same time, orders should be assigned to putwalls in order to balance operator workload and to avoid long recirculation on the conveyor. The assignment and sequencing of totes, when assuming known and constant tote induction times and when the goal is total completion time, is a problem similar to parallel machine scheduling. However, the goal from order consolidation is minimizing order, as opposed to tote, consolidation times. Order consolidation also depends on the putting operation. Moreover, both induction and putting times are uncertain and may vary because of different sizes and weights of SKUs and because of variations in manual and automatic processing. In order to reflect a realistic system performance accurately, we need to take randomness into account when evaluating measures like order completion time, resource utilization, and congestion.

In this work we analyze the order consolidation process under uncertainty. First, we build a simulation model for the consolidation process using a given tote assignment and sequencing. We use the empirical distributions derived from the industry data to run the simulation. Second, we develop a simulated annealing heuristic and apply it to solve the deterministic tote sequencing problem with the goal of minimizing order completion time or order processing time. Third, we develop stochastic simulated annealing algorithms to optimize the whole consolidation process. The simulation optimization approach decides on tote assignment and sequencing and evaluates the sequences using the simulation model.

We test and report results using real data.

The organization of the thesis is as follows. Chapter 2 defines the different problems addressed in this work and reviews related literature. Chapter 3 details the simulation model and the simulated annealing algorithms. Chapter 4 provides results of numerical testing using real industry instances under deterministic and stochastic settings. Chapter 5 presents the conclusion.

Chapter 2

Problem Definitions and Literature Review

In this chapter, we define the two main optimization problems in the order consolidation process. First, we define the tote sequencing problem and relate it to parallel machine scheduling. Second, we describe the order consolidation problem and explain the uncertainty involved in that operation. Finally, we review the literature related to parallel machine scheduling, simulation optimization, and simulated annealing.

2.1 Tote Sequencing and Related Literature

We use tote sequencing to refer to the two main decisions in tote sortation: assigning and sequencing totes to the multiple induction lines. Totes are inducted at the assigned

induction line in the order they arrive. Consequently, the sequence of totes determines their completion time. This is known in the literature as the Parallel Machine Scheduling (PMS) problem where induction lines correspond to machines, totes represent jobs and the objective is to minimize total completion time. This problem is known to be NP-hard ([Du et al., 1990]).

Within order consolidation, an order may have items in multiple totes and the totes may be inducted at different induction lines. An order is considered complete, i.e., consolidated and ready for packing, when all the totes that have a SKU from that order are inducted and put in the corresponding cubby. If we ignore the put operation, we may use tote completion times to approximate order completion times.Order completion time is equal to the completion time of the last inducted tote containing a SKU from the order. So the goal from tote sequencing becomes minimizing the total completion time of all orders. When SKUs of an order start arriving at the putwall, they occupy a cubby until all SKUs arrive. So cubby utilization may be approximated by order processing time defined by the difference between the first and last totes corresponding to an order. In the system under study, the size of a wave is set by the number of cubbies so that the number of orders in a wave does not exceed the number of cubbies. Consequently, minimizing order processing time may free up cubbies which may lead to forming larger waves.

To the best of our knowledge, the only work that studies the tote sequencing problem as defined above is by [Yildiz, 2017]. However, given that it relates closely to MPS, we review related literature briefly.

The first paper on parallel machine scheduling is [McNaughton, 1959]. It provides three

performance metrics, completion time, makespan and total weighted tardiness. [Cheng and Sin, 1990] provide a comprehensive review on parallel machine scheduling works that appeared before 1990. [Du et al., 1990] proves that when the number of machine is at least 2, the machine scheduling problem is NP-hard. [Eck and Pinedo, 1993] provides a heuristic to minimize makespan through completion-time optimal solutions and guarantee that the makespan achieved will be almost 5/4 times the optimal value. [McCormick and Pinedo, 1995] develop an algorithm to search for Pareto-optimal solution for both completion time and makespan. [Schutten and Leussink, 1996], [Lee and Pinedo, 1997] provides a three-phase heuristic algorithm for parallel machine scheduling with sequence-dependent setup times. The first phase is to compute the factors that characterize the instances (job-machine factor and the setup time severity factor). The second phase is to construct a sequence by a dispatching rule, which is a function depending on the factors from the first phase. The third phase is to use simulated annealing, with the sequence from the second phase as an initial solution. In the computational experiments of [Lee and Pinedo, 1997], there are 60 items for all the instances and the number of machines ranges from 1 to 12. [Chen and Powell, 1999] propose a column generation method. [Chen and Powell, 1999] first use Dantzig-Wolfe Decomposition to get a set partitioning problem, then they use the branch-and-bound to solve the set partitioning problem. In the branch-and-bound tree, [Chen and Powell, 1999] further use column generation to solve the linear relaxation problem at each node with a single machine subproblem. In the computational experiments of [Chen and Powell, 1999], they have used instances with at most 20 machines and 100 items. [van Den Akker et al., 1999] is another work that use column generation on parallel machine scheduling problem. They first obtain the set-covering reformulation, then solve

the LP-relaxation of the reformulation with column generation. For the pricing problem in the column generation, they provide a pseudo-polynomial time algorithm. [Dell'Amico et al., 2008] provides both metaheuristic and exact algorithms for identical parallel machine scheduling problem. [Dell'Amico et al., 2008] associate parallel machine scheduling problem to the bin packing problem, another well-known NP-hard problem. They propose an exact algorithm that solves bin packing problem in its subroutine to verify if the upper bound needs to be updated. They also provide a scatter search metaheuristic. The instances in [Dell'Amico et al., 2008] have at most 25 machines and 1000 items. [Tavakkoli-Moghaddam et al., 2009] uses genetic algorithm to minimize the number of tardy item and the total completion time of items simultaneously. They assume that there are sequence-dependent setup times and precedence constraints. [Tran et al., 2016] publish their work on solving parallel machine scheduling with setup time by logic-based benders' decomposition and branch and check.

[Lo and Bavarian, 1991], [Ruiz-Torres et al., 1997], [Radhakrishnan and Ventura, 2000] and [Lee et al., 2006] are the papers on using simulated annealing to solve parallel machine scheduling problem. [Lo and Bavarian, 1991] is the first research using simulated annealing on parallel machine scheduling. They use "item insertion" as the neighborhood structure, i.e., a new solution is obtained by moving an item from its current slot to another. Our work use the same neighborhood structure as [Lo and Bavarian, 1991]. [Ruiz-Torres et al., 1997] uses both pairwise interchange (swapping) and item insertion in their simulated annealing algorithm. Their objective is to minimize both the number of tardy item and the average completion time. [Radhakrishnan and Ventura, 2000] adopts a strategy to get the best succeeded solution from the neighborhood of the current solution, instead of

randomly picking a solution from its neighborhood. [Lee et al., 2006] uses the following neighborhood. They randomly select a item from a machine that has the largest completion time and randomly choose another item from the remaining machines, and interchanging the positions of the two selected items.

For the above work, the objective is to minimize the makespan of the jobs (or totes). To the best of our knowledge, [Yildiz, 2017] is the first study PMS with the objective of minimizing order completion. [Yildiz, 2017] provides a nonlinear mixed integer programming formulation of the problem, and a simulated annealing metaheuristic. They also develop a set partitioning formulation and column generation based solution. Our neighborhood structure is "insertion" while the neighborhood structure in [Yildiz, 2017] is "swapping".

2.2 Order Consolidation

In this section, we extend the scope to the whole consolidation process. After an SKU is inducted, it circulates on the conveyor until it reaches the destined putwall. An order is not consolidated until all of its SKUs are put in the cubby. If we think of putwalls as a set of second stage parallel machines, the problem becomes similar to a two stage hybrid flow shop problem. The hybrid flow shop problem is a generalization of the parallel machine scheduling problem with multiple stages of parallel machines. However, in the system under study, the traveling time from the induction lines through the conveyor to the putwalls are different. Also, SKUs from the same order should be assigned to the same putwall. These characteristics make order consolidation different from the hybrid flow shop problem. We refer readers to the review paper by [Ribas et al., 2010] on the hybrid flow

shop problem. Another important characteristic of the system under study is uncertainty in processing times. Induction is performed either manually or using automatic induction stations and putting is done manually by operators. Consequently, induction times, conveyor travel times, and putting times are all variable. In order to accurately account for order completion times, we study the order consolidation process under uncertainty where induction times and put times are variable and follow empirical distributions derived from the data. Our approach uses simulation and optimization to optimize order consolidation by deciding on tote sequencing and simulating the process to evaluate system performance. Our approach falls within the simulation optimization framework which we review next.

Simulation optimization (SO) is also called optimization via simulation (OvS), simulation-based optimization or black-box optimization. While the meaning can vary slightly depending on context, in this study we mainly refer to the definition given by [Fu, 2002]: "optimization of an objective function subject to constraints, both of which can be evaluated through a stochastic simulation." The field of SO is further divided into discrete simulated optimization (DSO) and continuous simulated optimization (CSO), depending on whether the solution space is discrete or continuous. The decisions in our study, namely, the sequence of totes and the assignment of orders to putwalls, are discrete decisions. Consequently, our work relates to DSO. For readers who are interested in SO, we recommend the review by [Amaran et al., 2016].

[Bianchi et al., 2009] characterizes stochastic optimization problems based on how the objective value is estimated. In the first case, exact closed-form expressions of the expected values are available. In the second case, closed-form expression of the expected value exists but it is very time consuming to calculate, so an estimate of the expected values

is applied. In the third case, there is no closed-form expression for the expected value. In that case the objective value is estimated by simulation, such a situation is also called Simulation Approximation. In this work, we use simulation to estimate the performance of the order consolidation system with different tote sequencing solutions. Consequently, our work falls under simulation approximation. [Amaran et al., 2016] gives a survey for the field of simulation optimization. Some of the common solution methodologies that can be applied to DSO include genetic algorithms ([Whitley, 1994]), simulated annealing ([Bertsimas et al., 1993]) and tabu search ([Glover, 1990]. In our work, we use simulated annealing as the main approach to optimize the order consolidation process. We review simulating annealing for deterministic and stochastic optimization problems next.

2.3 Review on Simulated Annealing

Simulated annealing (SA) originates from [Metropolis et al., 1953] which provides a procedure to optimize the properties of substances in the field of statistical mechanics. Later, [Kirkpatrick et al., 1983] propose the simulated annealing method by generalizing the concept of energy from [Metropolis et al., 1953] into objective minimization in optimization. Assuming a minimization problem, SA starts with an initial feasible solution referred to as the current solution. It selects a candidate solution from the neighborhood of the current solution. The neighborhood of a solution is the set of feasible solutions that can be reached from the current solution using a simple operation that is problem specific. It evaluates the objective value of the candidate solution. If the objective value of the candidate solution is smaller than that of the current solution, the candidate solution is accepted and

becomes the current solution. Otherwise, the candidate solution may still be accepted with a certain probability. The probability of accepting inferior solutions is a function of a temperature parameter and the gap between the objective values of the current and candidate solutions. The reason behind accepting inferior solutions is for the algorithm to escape local optimal solutions and search more neighborhoods that may be promising and lead to better solutions.

Simulated annealing has been applied to solve many classes of deterministic problems, including parallel machine scheduling ([Radhakrishnan and Ventura, 2000]), facility location ([Yigit et al., 2006]), and analog circuit design. [Bertsimas et al., 1993] gives an analysis of the convergence property of SA and discusses some of its applications. However, when the optimization problem is stochastic and the evaluation of the objective value is randomized, the solution may diverge ([Branke et al., 2008]). Because the algorithm does not in general know whether it has accepted a better solution or not, the solution may diverge over time. To handle the randomness in evaluations, some researchers present stochastic simulated annealing (SSA). [Bulgak and Sanders, 1988] propose a confidence-interval-based SA that can be applied to instances with randomness in their parameters. At each iteration, the algorithm evaluates the objective value of the candidate solution multiple times. If the confidence interval of the candidate solution covers the current solution, more evaluations are performed. [Gelfand and Mitter, 1989] add a random variable to the difference of the current and the previous objective values in transition probability. They prove that their SA is able to converge in probability to the global minimal solution, when there is a certain level of noises in evaluating the difference of the current and previous objective values. [Gutjahr and Pflug, 1996] show the methodology proposed by [Gelfand

and Mitter, 1989] guarantees convergence to an optimal solution when the evaluation noise is normally distributed and the standard deviation of the noise reduces at a certain rate. [Alrefaei and Andradóttir, 1999] provide two SA algorithms with constant temperature; these two algorithms involve storing and tracking the number of visits to past solutions. [Alkhamis et al., 1999] present another confidence interval based SA algorithm and proves that the algorithm converges to a global optimal solution with a probability.

Chapter 3

Optimizing Order Consolidation using Simulated Annealing

In this chapter, we describe the simulated system in detail, and we discuss the details of the simulation model of the order consolidation process. Then we provide the simulated annealing algorithms developed to optimize both the deterministic tote sequencing problem and the stochastic order consolidation process.

3.1 Simulation Model

The system under study is composed of three main components. After totes leave the picking area, they are stored temporarily in a multi-shuttle system that is fully automatic. When enough totes of a particular wave are in the multi-shuttle system, order consolidation

of the wave is initiated. Totes that leave the multi-shuttle system, are picked by induction operators and are processed at one of four induction lines. At the induction line, SKUs are emptied one at a time, scanned and placed on the conveyor. Once on the conveyor, SKUs are directed to the assigned putwall. There are three putwalls and a wave is assigned to only one of the putwall. Each putwall is divided into sections. Each section has one dedicated operator who picks up SKUs and place them in the right cubby. Orders are assigned to cubbies apriori and the whole operation is guided by a put-to-light system. Since orders are small, an order generally uses only one cubby. Consequently, the size of a wave is limited by the number cubbies in the putwall. An order is complete when all its SKUs arrive at the cubby. It is then packed and shipped to customers.

The simulation model represents the induction lines, conveyor, and putwall. The input to the model is an assignment of totes to induction lines and their sequence at each induction line. The model then simulates the order consolidation process according to the following rules.

- Totes are inducted in the order they arrive at the induction station.
- First, SKUs are emptied and scanned followed by the empty tote.
- SKUs are emptied from a tote in no predetermined order.
- SKUs are put in cubbies in the order they arrive at the putwall.

Induction times are variable and depend on whether a SKU or a tote is inducted and on whether the induction is done manually or automatically. We use the empirical

distributions derived from the data for the automatic induction lines to generate SKU and tote induction times randomly. Let the induction time between two SKUs be (t_1), the induction time between a SKU and the previous tote be (t_2), and the induction time between an empty tote and the last SKU be (t_3), as in Figure 3.1. Figures 3.2, 3.3, 3.4, 3.5, 3.6 and 3.7 display the distributions of t_1, t_2 and t_3 for induction lines 1 and 2. In the current implementation we assume that the travel time on the conveyor is fixed but depends on the relative location of the induction stations from the putwall. The travel time from induction lines 1, 2, 3 and 4 is set to 6, 5, 4 and 3 seconds, respectively. SKUs then join a queue waiting to be put in the cubby. At the putwall, we assume that there are three identical operators assigned equal sections of the putwall. The put times are random and are generated using an empirical distribution. When a SKU is inducted it either joins the putwall section with smallest queue if it is the first SKU of an order or it joins the same section as its order.

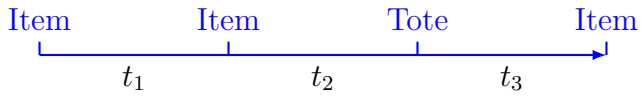

Figure 3.1: Timeline of item and tote induction

3.2 Simulated Annealing Algorithms

This section describes the simulated annealing algorithms developed to optimize order consolidation. To solve the deterministic tote sequencing problem, we use the framework given in [Bertsimas et al., 1993]. We use the confidence-interval based stochastic simulated annealing algorithms by [Bulgak and Sanders, 1988] and [Alkhamis et al., 1999] to optimize

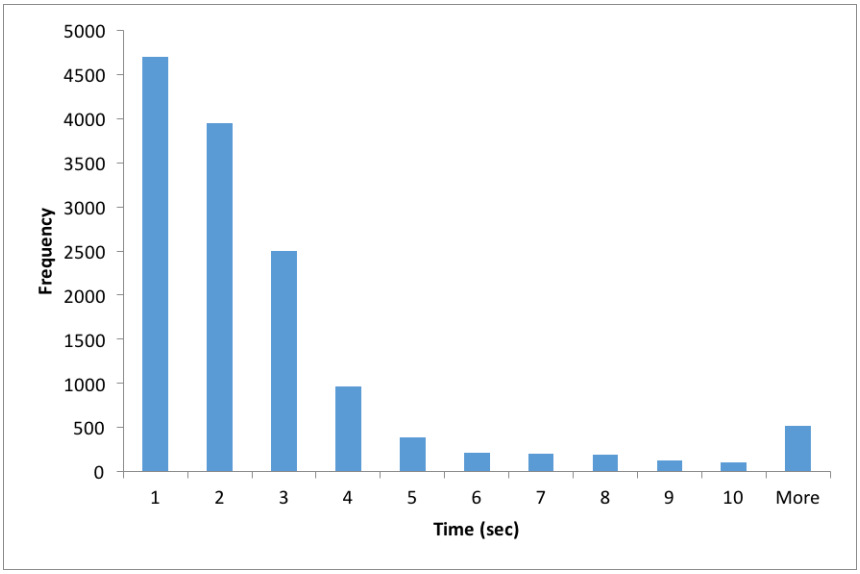

Figure 3.2: Distribution of t_1 for line 1

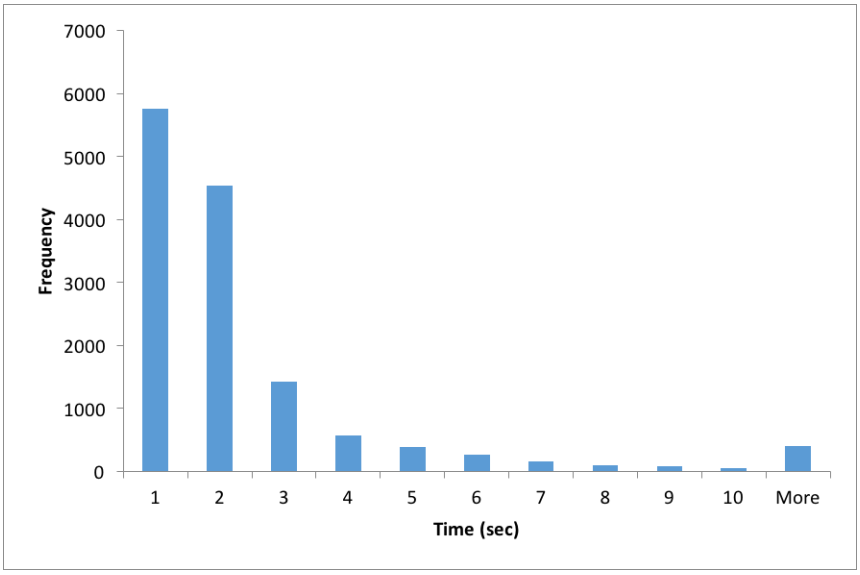

Figure 3.3: Distribution of t_1 for line 2

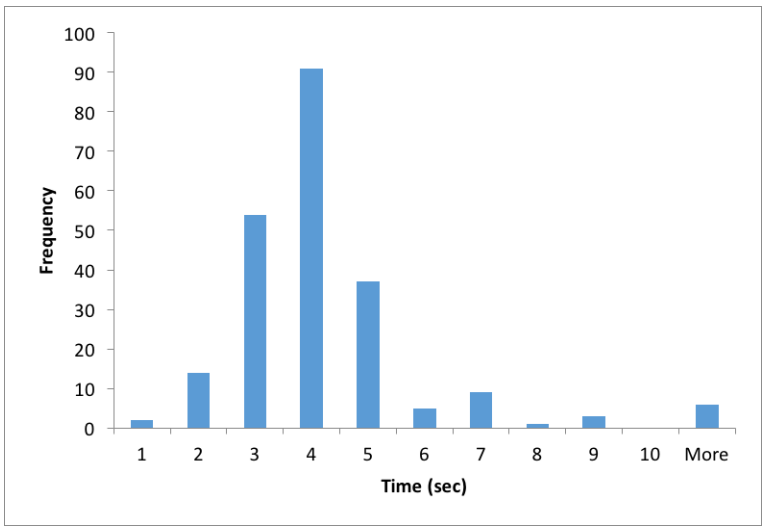

Figure 3.4: Distribution of t_2 for line 1

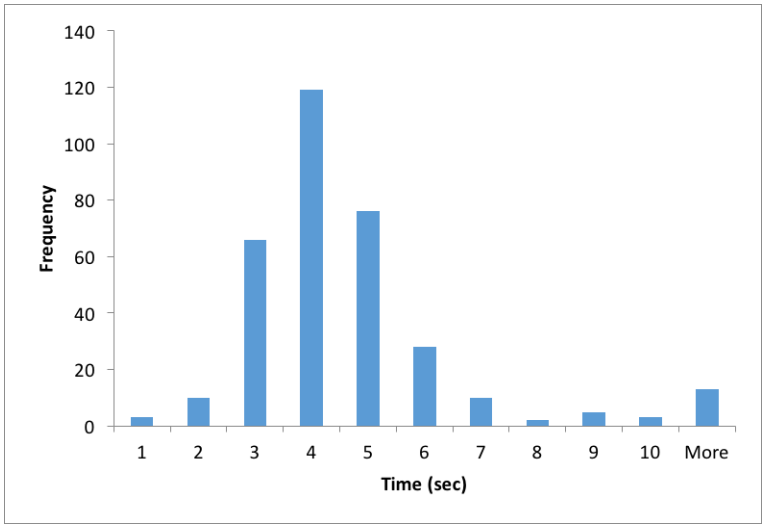

Figure 3.5: Distribution of t_2 for line 2

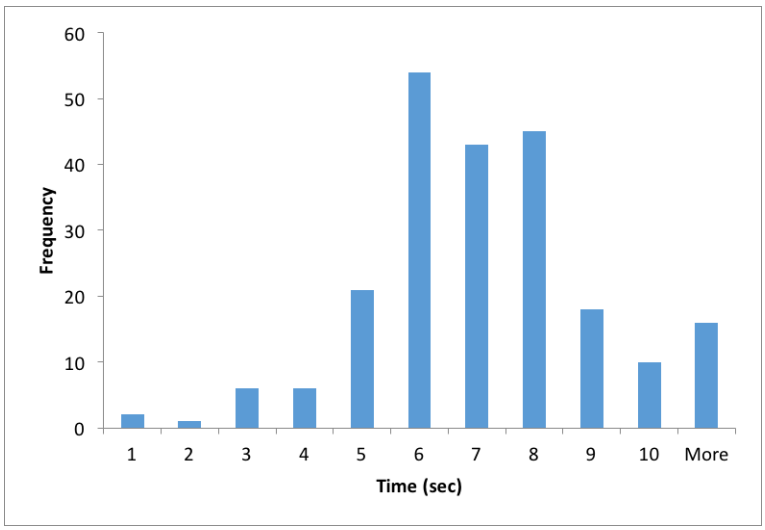

Figure 3.6: Distribution of t_3 for line 1

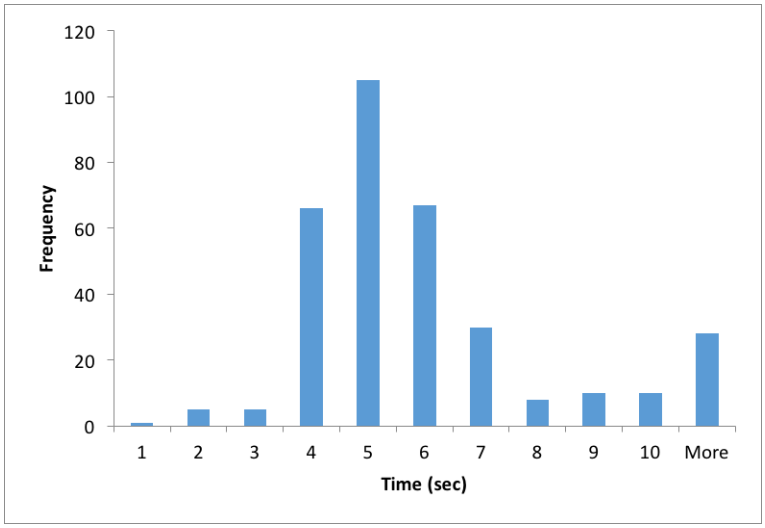

Figure 3.7: Distribution of t_3 for line 2

the whole operation. In discussing the algorithm, we assume the objective function is a minimization.

3.2.1 Deterministic Simulated Annealing

Simulated annealing is a metaheuristic algorithm that avoids local optimal solutions by accepting worse solutions. At each iteration, the algorithm randomly selects a candidate solution from the neighborhood of the current solution, and evaluates its objective value. A neighborhood is a set of solutions that may be obtained by some modification of the current solution. The evaluated objective value of the candidate solution may or may not be better than the current solution. If the objective of the candidate solution is better than that of the current solution, the candidate solution becomes the new current solution. Otherwise, the algorithm accepts the worse candidate solution using a probability function. The probability function relates to a temperature parameter T, and is non-increasing. The algorithm stops either if the temperature T has reached zero, or if the maximum number of iterations is reached. Before presenting the pseudo-code of generic simulated annealing, we first give a list of notations.

Notation

S: The finite solution space.

k: Iteration number.

i: Current solution.

S^*: The set of global minima, which is assumed to be a proper subset of S.

f: A deterministic cost function $f : S \to R$.

$\mathcal{N}(i)$: The neighbor of $i \in S$, $i \notin \mathcal{N}(i)$.

x_k: The vector of decision variables at iteration k.

x^*: The best solution so far.

q_{ij}: A collection of positive coefficients q_{ij}, $j \in N(i)$, such that $\sum_{j \in N(i)} q_{ij} = 1$. $j \in N(i)$ if and only if $i \in N(j)$.

T_k: A nonincreasing function $T_k : Z \to (0, \infty)$, called the cooling schedule. Z is the set of nonnegative integers, and $T(k)$ is called the temperature at iteration k.

Algorithm 1 is the generic simulated annealing algorithm, and we present a pseudo-code as follows.

Algorithm 1:

Step 1: Determine x_k arbitrarily at $k = 0$ and obtain an initial temperature $T_k > 0$.

Step 2: While stopping criteria = false do the following:

2.1 Given $x_k = i$ choose a candidate solution $z_k \in \mathcal{N}(i)$ with probability distribution

$$P[z_k = j | x_k = i] = q_{ij},\ j \in \mathcal{N}(i)$$

2.2 Evaluate $f(j)$, if $f(j) < f(i)$, set $x_{k+1} = z_k$, else go to step 2.3

2.3 Given $z_k = \mathrm{j}$, generate $U_k \sim U[0, 1]$, and set

$$x_{k+1} = \begin{cases} z_k, & \text{if } U_k \leq \exp[\frac{-[f(z_k) - f(x_k)]}{T_k}] \\ x_k, & \text{otherwise} \end{cases}$$

2.4 Set $k = k + 1$.

Step 3: Update T_k and go to Step 2.

The initial solution, objective function, and neighborhood structure used in the case of deterministic tote sequencing are defined in Section 3.3.

3.2.2 Stochastic Simulated Annealing

[Bulgak and Sanders, 1988] provide a simulated annealing algorithm that may be applied to stochastic optimization problems. Compared to the generic simulated annealing, there are two major differences. First, as the evaluation of the objective function is an approximation, [Bulgak and Sanders, 1988] require more than one evaluation of the candidate solution, and take the average of the calculated objective values from multiple evaluations as an estimate. The objective values of the candidate solution and the current solution in the probability acceptance function, are replaced by the sample average estimate of the candidate solution and the sample average estimate of the current solution, respectively. Second, the number of samplings in each iteration may vary depending on how significant is the difference between the estimated objective values of the candidate solution and the current solution. [Bulgak and Sanders, 1988] uses a statistical confidence interval to decide on whether more evaluations are required. The confidence interval is set so that the mean falls into with a certain probability. For example, if the mean value of a random number is in the interval of $[0, 1]$ with 95% probability, then we call $[0, 1]$ a 95% confidence interval for the random variable. The method determines whether to accept a candidate solution by calculating the confidence interval of its objective value. If the objective value of candidate

solution is estimated to be smaller than that of the current solution, and its confidence interval covers the current solution, then the algorithm evaluates the objective value of candidate solution again.

In addition to the notation introduced in Section 3.2.1, we list the following notation in order to present the pseudo code of the algorithm in [Bulgak and Sanders, 1988]. In the rest of this thesis, we refer to the algorithm of [Bulgak and Sanders, 1988] as Algorithm 2.

Notation:

$F(x_k)$: The objective function whose value is estimated by discrete event simulation at x_k.

r: The number of discrete event simulation replications currently performed.

R: The maximum number of replications allowed where R is chosen *a priori*.

m_{xkr}: Statistical estimate of the mean of $F(x_k)$ based on r replications.

$\hat{\delta}_{xkr}$: Statistical estimate of the variance of $F(x_k)$ based on r replications.

CI_{xkr}: Confidence interval defined as $[m_{xkr} - t_{\alpha/2,r-1}(\hat{\delta}_{xkr})^{1/2}, m_{xkr} + t_{\alpha/2,r-1}(\hat{\delta}_{xkr})^{1/2}]$.

Algorithm 2:

Step 1: Determine x_k arbitrarily at $k = 0$ and obtain an initial temperature $T_k > 0$.

Step 2: Assign $x_k \rightarrow x^*$ and compute m^*_{xkr} where $r = R$ at $k = 0$.

Step 3: Generate $x_k \in N(x^*)$.

Step 4: Compute m_{xkr} and CI_{xkr} based on r replications where initially $r = 2$ at j.

Step 5a: Check if $m^*_{xkr} \in [m_{xkr} - t_{\alpha/2,r-1}(\hat{\delta}_{xkr})^{1/2}, m_{xkr}]$, go to Step 5b, otherwise go to Step 6a.

Step 5b: If $r < R$, $r \rightarrow r + 1$; if $r = R$, go to Step 3.

Step 6a: If $m_{xkr} < m^*_{xkr}$, go to Step 6b, otherwise go to Step 6c.

Step 6b: $x_k \rightarrow x^*$.

Step 6c: Generate u, if $u < \exp[\frac{-(m_{xkr} - m^*_{xkr})}{T_k}]$, $x_j \rightarrow x^*$, otherwise reject x_k. In both cases, go to Step 7.

Step 7: $j \rightarrow k + 1$, and update T_k. If stopping criterion is satisfied, go to Step 8, otherwise go to Step 3.

Step 8: Stop

Another confidence interval based simulated annealing algorithm is proposed by [Alkhamis et al., 1999]. The main difference between [Bulgak and Sanders, 1988] and [Alkhamis et al., 1999] is that the latter uses the standard deviation of the estimate $\hat{\delta}_{xkr}$ at iteration k to modify the probability of accepting the candidate solution. While [Bulgak and Sanders, 1988] does not theoretically guarantee a convergence to a global optimal solution, [Alkhamis et al., 1999] prove that their algorithm converges under suitable conditions on the random error. The algorithm is described next, and in the rest of this thesis, we refer to the algorithm of [Alkhamis et al., 1999] as Algorithm 3.

Algorithm 3:

Step 1: Determine x_k arbitrarily at $k = 0$ and obtain an initial temperature $T_k > 0$.

Step 2: While stopping criteria = false do the following:

2.1 Given $x_k = i$ choose a candidate $z_k \in \mathcal{N}(i)$ with probability distribution

$$P[z_k = j | x_k = i] = q_{ij},\ j \in \mathcal{N}(i)$$

2.2 Evaluate m^*_{xkr}, m_{xkr}, and $\hat{\delta}_k$.

2.3 Given $z_k = \text{j}$, generate $U_k \sim U[0,1]$, and set

$$x_{k+1} = \begin{cases} z_k, & \text{if } U_k \leq \exp[\frac{-(m_{xkr} - m^*_{xkr} - t_k\hat{\delta}_{xkr})^+}{T_k}] \\ x_k, & \text{otherwise} \end{cases}$$

2.4 Set $k = k + 1$.

Step 3: Update T_k and go to Step 2.

3.3 Simulated Annealing for Tote Sequencing and Order Consolidation

To apply simulated annealing to our problems, we specify the initial solution, the neighborhood structure, the temperature and the stopping criteria. For the initial solution, we use the sequence generated by the warehouse system in all our testing. The solutions of tote sequencing can be represented by a set of queue data structures. Each induction line is a queue and the elements in the queues are the totes. Every tote of a wave should be in one and only one induction line, and they are represented unique indices. We use an insertion type neighborhood structure. By insertion, we mean that we first randomly select an induction line with a uniform distribution and randomly select a tote from that induction line using a uniform distribution. We then randomly select an induction line and a tote, with uniform distribution as well, and insert the former selected tote to the front. As we can regard the end of each induction line queue as a dummy tote, we can also insert

the totes to the end of the induction line. Figure 3.8 gives two examples of how to obtain new solutions by insertion. The same neighborhood structure is used for the deterministic and stochastic problems.

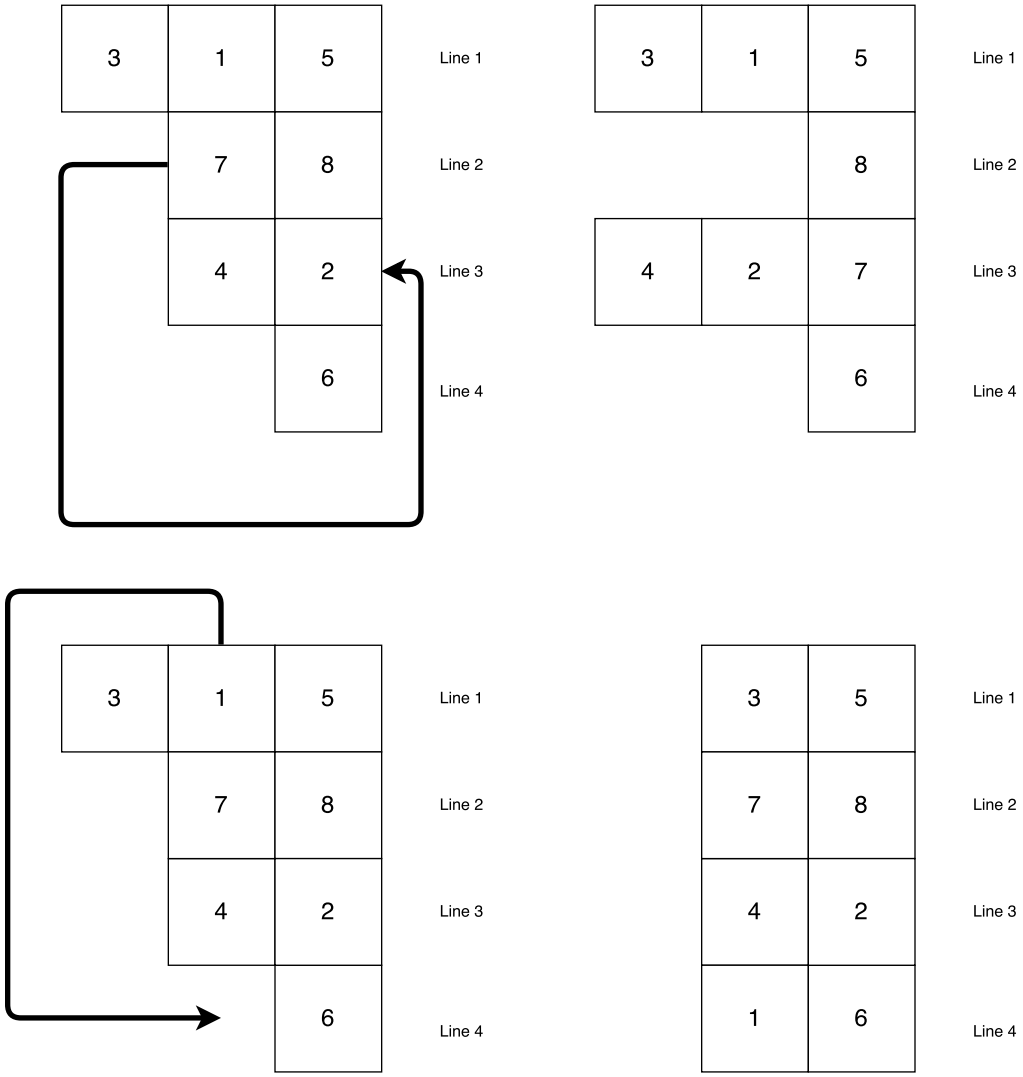

Figure 3.8: Neighborhood structure

[Nourani and Andresen, 1998] summarize and benchmark different cooling schedules for

SA. Our work adopts the exponential schedule and the logarithmic schedule. In exponential cooling schedule, in subiteration k, $T_k = \alpha \times T_{k-1}$, where α is a constant between 0 and 1. logarithmic schedule is defined as $T_k = \frac{C}{\ln(1+k)}$, where C is a constant. For Algorithm 1, we use exponential schedule and set the initial temperature $T_0 = 1$ and $\alpha = 0.99$ if we do not specify otherwise. For Algorithms 2 and 3, we use logarithmic schedule, where C is the magnitude of the objective value of the initial solution. Figure 3.10 displays the cooling schedule when $C = 100$.

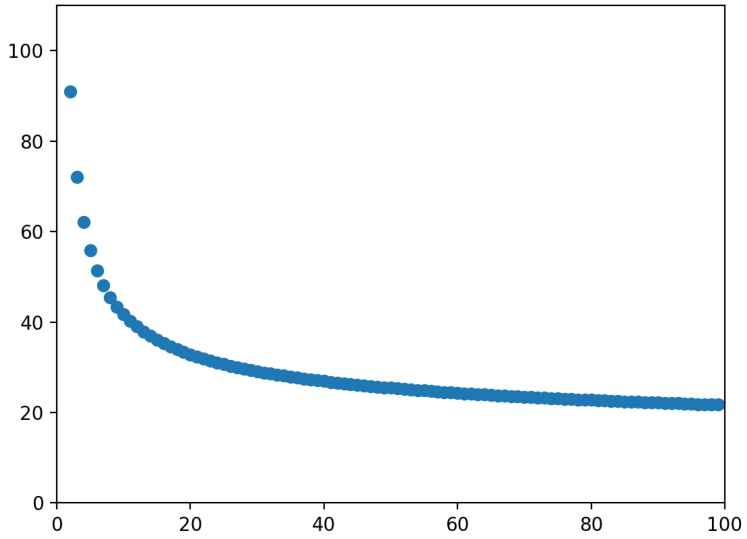

Figure 3.9: Cooling schedule of exponential schedule with $C = 100$

We set the initial number of replications $r_0 = 30$ and the maximum number of replications $R = 50$ and a 95% confidence interval in Algorithm 2. We perform 30 replications at each iteration k in Algorithm 3. The stopping criterion is based on the number of iterations, and is set to 10,000 iterations.

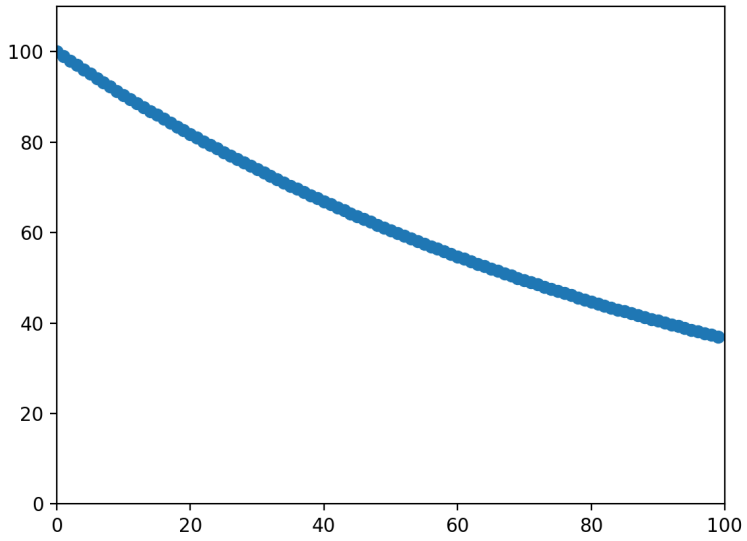

Figure 3.10: Cooling schedule of logarithmic schedule with $T_0 = 100$

Chapter 4

Numerical Results

In this chapter, we present the numerical testing of the simulated annealing algorithm for deterministic tote sequencing and stochastic order consolidation. All the algorithms including the simulation model are implemented in C++. We carried out the testing on a 2014 version Macbook Pro with 2.5 GHz Intel Core i7 Processor. We use 10 industry instances based on 10 waves processed by the order consolidation system under study and used by [Yildiz, 2017]. A wave is characterized by the number of totes, number of orders and number of SKUs. Except for wave number 292, an average wave has 321 totes, 1147 orders and 4881 SKUs. Table 4.1 summarizes the number of totes, orders and items for each wave. Since we have the scan times for all operations in order consolidation, we are able to determine the tote assignment and sequencing solution used by the system. We use this solution as an initial solution in our testing. We also use the system data to calculate actual tote induction times and use these as tote processing times in the deterministic tote sequencing problem. While in the stochastic problems, we use the empirical distributions

of induction times to run the simulation model.

Waves	#Totes	#Orders	#SKUs
67	353	1243	4202
73	374	1384	6702
74	420	1374	6275
75	368	1373	5646
76	336	1367	5601
78	257	851	2978
90	253	932	3925
188	377	989	4667
193	169	949	3932
292	121	267	1234

Table 4.1: Summary of the industry instances

4.1 Results on Deterministic Tote Sequencing

In deterministic tote sequencing, we minimize the total completion time of all orders. We measure order completion time as the finish time of the last tote that has SKUs of the order. We also calculate order processing time as the difference between its completion time and the start time of the first tote that has SKUs of the order processing time is an estimate of the time an order occupies a cubby. The tote processing times are calculated as in [Yildiz, 2017].

Table 4.2 shows the results of using simulated annealing to solve the industry instances. The second and third columns are the average order completion time in seconds for the initial solution (Initial obj val) and the optimized solution (Final obj val), respectively. The fourth column is for the percentage improvement calculated as $\frac{\text{Initial obj val - Final obj val}}{\text{Initial obj val}}$. The last column gives the CPU time in seconds.

Waves	Initial obj val	Final obj val	Yildiz, 2017	Improvement %	Time (sec)
67	1516.76	871.17	879.60	42.56	63.63
73	2639.32	1660.82	1662.60	37.07	96.17
74	2236.04	1551.85	1560.60	30.60	91.94
75	1911.32	1278.31	1286.40	33.12	82.41
76	1930.38	1289.37	1293.60	33.21	79.14
78	1129.63	681.13	687.60	39.70	44.38
90	1650.42	1062.98	1066.20	35.59	55.16
188	1977.98	1685.35	1845.00	14.79	67.25
193	1247.91	1133.37	1135.80	9.18	51.24
292	489.74	315.11	318.00	35.66	16.69

Table 4.2: Simulated annealing for deterministic tote sequencing with 10000 iterations

By optimizing tote sequencing, the total order completion time is reduced by 31.15% on average and by more than 30% in 8 out of the 10 instances.

We also perform a set of experiments on using exponential scheduling and logarithmic scheduling for the deterministic tote sequencing with different initial temperatures T_0 and constant parameters C, respectively. Table 4.3 shows that in the case of using exponential cooling schedule, T_0 does not impact solution quality within the iteration limit. On the other hand, the value of C plays a significant role in the quality of the solutions. As C increases, solution quality decreases significantly.

4.2 Results on Order Consolidation Optimization

Before simulating the whole operation, we experiment with solving the tote sequencing problem with uncertain processing times. Instead of using deterministic processing times recorded by the system, we use the empirical distributions of SKU and empty tote induction times to simulate the tote induction operation. In this case, we use items as opposed to

Waves	Exponential schedule				Logarithmic schedule			
	$T_0 = 1$	$T_0 = 10^3$	$T_0 = 10^4$	$T_0 = 10^6$	$C = 1$	$C = 10^3$	$C = 10^4$	$C = 10^6$
67	871.17	873.45	873.28	874.90	874.50	878.70	975.30	1322.38
73	1660.82	1661.25	1664.66	1662.74	1664.46	1663.34	1750.27	2210.14
74	1551.85	1550.27	1554.18	1554.51	1553.79	1555.77	1664.16	2072.84
75	1278.31	1280.70	1282.05	1278.29	1282.66	1287.96	1386.77	1754.45
76	1289.37	1284.03	1283.75	1283.39	1284.26	1287.17	1381.05	1732.32
78	681.13	680.06	679.67	679.73	681.10	688.73	787.82	1017.95
90	1062.98	1059.71	1062.81	1059.31	1062.43	1068.30	1151.00	1500.46
188	1685.35	1674.57	1678.63	1678.48	1678.50	1679.70	1819.45	1919.37
193	1133.37	1134.41	1131.09	1136.31	1130.01	1129.67	1189.04	1234.94
292	315.11	314.88	314.19	313.75	313.54	330.79	403.15	437.94
Avg	1152.95	1151.33	1152.43	1152.14	1152.52	1157.01	1250.80	1520.28

Table 4.3: Simulated annealing for deterministic tote sequencing with different cooling schedules and initial temperature T_0

totes to approximate order start and finish times. Order start time is the induction time of its first SKU, and its completion time is the induction time of its last SKU. We apply both algorithms 2 and 3 to minimize the order completion time. To better evaluate the goodness of these two methods, we also simulate the deterministic solution 100 times and take the average as the estimated objective value. Table 4.4 shows the test results of the three methods where the initial solution and the best solutions found are simulated 100 replications and the average is presented. In general, the deterministic solution performs the best, followed by Algorithm 2, then Algorithm 3. In comparison with the deterministic solution, the stochastic simulated annealing solutions only achieve small improvements.

We now present the results on the whole order consolidation process. A tote sequence is generated at every iteration of simulated annealing and is evaluated using the simulation model. In addition to order completion time, we also calculate order processing times and SKU wait time for the operators at the putwall. To better understand the difference between the three metrics, we refer to the timeline in Figure 4.1 for the main events of

Waves	Initial obj val	Final obj val			Improvement%			Time (sec)		
		Algorithm 1	Algorithm 2	Algorithm 3	Algorithm 1	Algorithm 2	Algorithm 3	Algorithm 1	Algorithm 2	Algorithm 3
67	2362.68	2002.62	2308.59	2294.39	15.24	2.29	2.89	89.58	3655.15	1917.98
73	4397.52	3776.79	4174.15	4207.16	14.12	5.08	4.33	137.74	4410.88	2885.56
74	3722.45	3219.23	3650.97	3601.50	13.52	1.92	3.25	139.32	4033.00	2652.73
75	3183.07	2738.24	3146.45	3129.01	13.97	1.15	1.70	121.05	4046.39	2590.22
76	3230.27	2697.53	3106.15	3060.73	16.49	3.84	5.25	122.06	3941.16	2457.28
78	3921.51	3318.25	1544.64	1541.29	15.38	4.40	4.61	137.09	2701.29	1389.90
90	2410.98	2038.20	2293.62	2262.22	15.46	4.87	6.17	84.99	3124.03	1623.98
188	2834.31	2402.31	2765.36	2728.85	15.24	2.43	3.72	101.86	3608.31	2018.98
192	2620.33	2351.19	2573.81	2569.20	10.27	1.78	1.95	105.03	3685.99	2312.77
193	2161.68	1955.48	2099.92	2139.60	9.54	2.86	1.02	83.33	3522.97	1610.60
292	720.51	564.91	642.53	682.84	21.60	10.82	5.23	26.31	2153.32	396.94

Table 4.4: Comparison of deterministic and stochastic simulated annealing algorithms on tote sequencing solutions

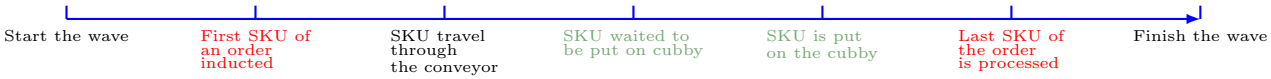

Figure 4.1: Timeline of events in order consolidation process

the consolidation operation. We use red text to specify the events for an order and we use green text to specify the events for a SKU. The order processing time and SKU wait time are represented by the intervals between events with red text and events with green text on the time line, respectively.

Similar to the tests for stochastic tote sequencing, we apply Algorithms 2 and 3 and compare to a simulation of the deterministic tote sequencing solution. At the putwall, the first SKU of an order is assigned to the operator with smallest queue, and the rest of its SKUs use the same assignment. Because we do not have the data for item put processing times at the putwall, we assume that all operators are identical and use the distribution of SKU induction times at line 1 as generate put times.

The computational results in Table 4.5 show that the deterministic solution outperforms the solutions found by the stochastic simulated annealing algorithms. Table 4.6 shows similar results with the objective of minimizing order processing times. With this objective too, the deterministic solution outperforms the stochastic solutions. Table 4.7 shows the results when the objective is to minimize the total SKU wait time for putting. With this objective, the stochastic simulated annealing solutions do as well as the deterministic solution, and achieve significant improvement compared to the initial solutions.

To analyze the impact of the number of induction lines on the consolidation operaiton, we vary the number of induction lines between 3 and 6. We assume the distribution of

Waves	Initial obj val	Final obj val			Improvement%			Time (sec)		
		Algorithm 1	Algorithm 2	Algorithm 3	Algorithm 1	Algorithm 2	Algorithm 3	Algorithm 1	Algorithm 2	Algorithm 3
67	2688.93	2317.21	2660.85	2671.40	13.82	1.04	0.69	97.07	3776.70	1998.35
73	4684.50	4097.45	4507.99	4532.39	12.53	3.77	3.81	150.46	4662.58	2920.96
74	4382.56	3823.62	4351.90	4359.84	12.75	0.70	0.56	141.60	4486.21	2762.19
75	3801.57	3328.88	3785.95	3835.40	12.43	0.41	0.14	124.08	4312.13	2660.27
76	3825.48	3257.90	3723.04	3758.08	14.84	2.68	1.42	129.88	4420.50	2590.01
78	1840.78	1578.33	1794.17	1743.08	14.26	2.53	5.31	141.06	3321.65	1437.85
90	2802.34	2487.94	2718.41	2741.73	11.22	2.99	4.30	85.98	3554.45	1785.83
188	3248.88	2821.86	3136.12	3173.54	13.14	3.47	3.47	104.29	3917.00	2118.88
193	2688.73	2439.66	2604.81	2648.09	9.26	3.12	3.31	84.58	3628.91	1842.68
292	790.84	636.10	756.72	775.38	19.57	4.31	6.54	27.77	2515.74	573.26

Table 4.5: Comparison of deterministic and stochastic solutions for order consolidation with the objective of minimizing order completion time

Waves	Initial obj val	Final Objective Estimations			Improvement%			Time (sec)		
		Algorithm 1	Algorithm 2	Algorithm 3	Algorithm 1	Algorithm 2	Algorithm 3	Algorithm 1	Algorithm 2	Algorithm 3
67	1723.66	1458.35	1739.16	1712.64	15.39	-0.90	0.68	97.07	3822.64	2031.99
73	4024.16	3415.39	3855.97	3916.05	15.13	4.18	5.22	150.46	4577.19	2927.09
74	3350.62	2760.98	3206.14	3268.09	17.60	4.31	1.46	141.60	4559.83	2882.74
75	2783.85	2345.59	2715.03	2756.29	15.74	2.47	2.15	124.08	4305.63	2676.56
76	2830.46	2253.48	2755.50	2743.42	20.38	2.65	4.48	129.88	5332.84	2635.55
78	1091.71	945.07	1049.48	1068.14	13.43	3.87	2.16	141.06	3307.35	1408.67
90	2155.63	1871.00	2088.89	2118.08	13.20	3.10	3.39	85.98	3641.39	1803.65
188	2413.01	2054.03	2299.43	2376.52	14.88	4.71	5.05	104.29	3942.34	2089.06
193	2079.18	1813.85	1990.31	2020.21	12.76	4.27	6.68	84.58	3595.66	1779.99
292	505.41	402.37	475.66	476.36	20.39	5.89	5.75	27.77	2438.25	578.27

Table 4.6: Comparison of deterministic and stochastic solutions for order consolidation with the objective of minimizing order processing time

Waves	Initial obj val	Final Objective Estimations			Improvement%			Time (sec)		
		Algorithm 1	Algorithm 2	Algorithm 3	Algorithm 1	Algorithm 2	Algorithm 3	Algorithm 1	Algorithm 2	Algorithm 3
67	232.81	234.34	220.04	215.89	-0.66	5.49	7.27	97.07	3745.90	1981.39
73	480.98	418.46	394.33	389.98	13.00	18.01	18.92	150.46	4511.64	2995.21
74	385.60	365.23	348.17	366.17	5.28	9.71	5.04	141.60	4311.65	2732.64
75	388.56	354.95	350.41	348.36	8.65	9.82	10.35	124.08	4025.38	2487.79
76	399.67	332.46	336.94	325.52	16.82	15.70	18.55	129.88	4225.99	2479.28
78	152.79	155.52	135.74	117.83	-1.79	11.16	22.88	141.06	3199.64	1389.73
90	281.13	258.60	261.82	264.44	8.01	6.87	5.94	85.98	3449.79	1732.30
188	284.86	277.90	194.81	187.67	2.44	31.61	34.12	100.68	3766.78	2003.26
193	293.68	253.78	254.34	263.06	13.59	13.40	10.43	84.58	3451.29	1730.52
292	70.25	59.77	23.57	40.99	14.91	66.45	41.64	27.77	2501.98	558.58

Table 4.7: Comparison of deterministic and stochastic solutions with the objective of minimizing SKU wait time

induction times in line 5 and 6 will be the same as those of the induction line 1 and 2, respectively. Also, we set the traveling time on the conveyor to be 8 and 9 seconds for induction lines 5 and 6, respectively. In addition, we fix the number of operators to be 3, which is the same as the previous setting. We use Algorithm 1 to obtain the tote sequencing solution and provide the operation results of the simulation. Table 4.8 displays the order completion times, order processing times and item wait times when using tote sequencing obtained by Algorithm 1. By comparing to the baseline scenario when there are 4 induction lines, We can observe that when there are more fewer induction lines, the order completion times are much longer, and order processing times are slightly longer for each of the waves. When the number of induction lines increases to 5, the order completion times drop. The order processing time may increase or decrease depending on waves, but in general the changes are not so much in compare to those of the order completion times. When the number of induction lines increases to 6, the order completion time still decreases in compare to those of the 5 induction lines scenario. However, the order processing time and the item wait time do not increase significantly, and in some wave they can even lightly decrease. A possible explanation is that the traveling times from induction lines have buffered the items from queue up in front of the putting area.

Waves	3 induction lines			4 induction lines		
	Order completion time	Order processing time	Item wait time	Order completion time	Order processing time	Item wait time
67	2739.49	1528.50	7.69	2317.21	1458.35	234.34
73	4631.12	3201.34	13.57	4097.45	3415.39	418.46
74	4383.09	2950.56	10.43	3823.62	2760.98	365.23
75	3754.31	2434.90	11.25	3328.88	2345.59	354.95
76	3690.83	2333.01	10.58	3257.90	2253.48	332.46
78	1872.70	988.17	8.15	1578.33	945.07	155.52
90	2806.64	1968.67	10.89	2487.94	1871.00	258.60
188	3278.48	2205.38	10.17	2821.86	2054.03	277.90
193	2713.36	1873.17	15.67	2439.66	1813.85	253.78
292	784.60	439.97	9.16	636.10	402.37	59.77
Waves	5 induction lines			6 induction lines		
	Order completion time	Order processing time	Item wait time	Order completion time	Order processing time	Item wait time
67	2218.23	1508.12	310.97	2105.54	1513.10	304.17
73	3877.27	3031.30	488.91	3717.94	3004.46	493.00
74	3595.66	2764.63	464.05	3449.35	2773.66	459.84
75	3165.76	2388.21	431.66	3026.56	2370.68	419.71
76	3069.89	2286.85	408.08	2948.13	2300.57	403.95
78	1502.79	980.80	219.52	1424.17	995.05	212.43
90	2356.23	1853.70	312.60	2254.32	1853.94	299.23
188	2668.56	2013.12	357.63	2517.24	1977.44	340.22
193	2312.69	1835.33	289.77	2206.50	1791.90	273.45
292	609.46	410.51	90.13	578.25	412.25	89.98

Table 4.8: Comparison of order completion time, order processing time and item wait time under different numbers of inductions lines

Chapter 5

Conclusion

This thesis studies the order consolidation system of an e-commerce warehouse using real data. The goal is to optimize the order consolidation process and use the resources efficiently. We focus on two main optimization problems: the deterministic tote sequencing problem and the stochastic order consolidation problem. We propose a simulation optimization approach where we develop a simulation of the operations and use it to evaluate solutions within a simulated annealing framework. We developed three simulated annealing algorithms for deterministic tote sequencing and stochastic order consolidation. In particular, for the order consolidation problem, we carried out experiments with three different objectives including order completion time, order processing time and SKU wait time. Our experiments show that using deterministic simulated annealing to solve a deterministic tote sequencing yields better solutions when the objective is minimizing order completion time or order processing time. However, for the wait time objective, deterministic solutions may result in that the SKUs need to wait longer for putting. We also evaluate the impact of

changing the number of induction lines on the order consolidation process.

References

[Al-Khamis and MHallah, 2011] Al-Khamis, T. and MHallah, R. (2011). A two-stage stochastic programming model for the parallel machine scheduling problem with machine capacity. *Computers & Operations Research*, 38(12):1747–1759.

[Alkhamis et al., 1999] Alkhamis, T. M., Ahmed, M. A., and Tuan, V. K. (1999). Simulated annealing for discrete optimization with estimation. *European Journal of Operational Research*, 116(3):530–544.

[Alrefaei and Andradóttir, 1999] Alrefaei, M. H. and Andradóttir, S. (1999). A simulated annealing algorithm with constant temperature for discrete stochastic optimization. *Management science*, 45(5):748–764.

[Amaran et al., 2014] Amaran, S., Sahinidis, N. V., Sharda, B., and Bury, S. J. (2014). Simulation optimization: a review of algorithms and applications. *4OR*, 12(4):301–333.

[Amaran et al., 2016] Amaran, S., Sahinidis, N. V., Sharda, B., and Bury, S. J. (2016). Simulation optimization: a review of algorithms and applications. *Annals of Operations Research*, 240(1):351–380.

[Amazon, 2017] Amazon (2017). Amazon q4 2016 financial report.

[Arnaout, 2014] Arnaout, J.-P. (2014). Rescheduling of parallel machines with stochastic processing and setup times. *Journal of Manufacturing Systems*, 33(3):376–384.

[Bertsimas et al., 1993] Bertsimas, D., Tsitsiklis, J., et al. (1993). Simulated annealing. *Statistical science*, 8(1):10–15.

[Bianchi et al., 2009] Bianchi, L., Dorigo, M., Gambardella, L. M., and Gutjahr, W. J. (2009). A survey on metaheuristics for stochastic combinatorial optimization. *Natural Computing*, 8(2):239–287.

[Branke et al., 2008] Branke, J., Meisel, S., and Schmidt, C. (2008). Simulated annealing in the presence of noise. *Journal of Heuristics*, 14(6):627–654.

[Bulgak and Sanders, 1988] Bulgak, A. A. and Sanders, J. L. (1988). Integrating a modified simulated annealing algorithm with the simulation of a manufacturing system to optimize buffer sizes in automatic assembly systems. In *Simulation Conference Proceedings, 1988 Winter*, pages 684–690. IEEE.

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

[Cheng and Sin, 1990] Cheng, T. and Sin, C. (1990). A state-of-the-art review of parallel-machine scheduling research. *European Journal of Operational Research*, 47(3):271–292.

[Dell'Amico et al., 2008] Dell'Amico, M., Iori, M., Martello, S., and Monaci, M. (2008). Heuristic and exact algorithms for the identical parallel machine scheduling problem. *INFORMS Journal on Computing*, 20(3):333–344.

[Du et al., 1990] Du, J., Leung, J. Y.-T., and Young, G. H. (1990). Minimizing mean flow time with release time constraint. *Theoretical Computer Science*, 75(3):347–355.

[Eck and Pinedo, 1993] Eck, B. T. and Pinedo, M. (1993). On the minimization of the makespan subject to flowtime optimality. *Operations Research*, 41(4):797–801.

[Fu, 2002] Fu, M. C. (2002). Optimization for simulation: Theory vs. practice. *INFORMS Journal on Computing*, 14(3):192–215.

[Gelfand and Mitter, 1989] Gelfand, S. B. and Mitter, S. K. (1989). Simulated annealing with noisy or imprecise energy measurements. *Journal of Optimization Theory and Applications*, 62(1):49–62.

[Gielen et al., 1989] Gielen, G., Walscharts, H., and Sansen, W. (1989). Analog circuit design optimization based on symbolic simulation and simulated annealing. In *Solid-State Circuits Conference, 1989. ESSCIRC'89. Proceedings of the 15th European*, pages 252–255. IEEE.

[Glover, 1990] Glover, F. (1990). Tabu search: A tutorial. *Interfaces*, 20(4):74–94.

[Gutjahr and Pflug, 1996] Gutjahr, W. J. and Pflug, G. C. (1996). Simulated annealing for noisy cost functions. *Journal of Global Optimization*, 8(1):1–13.

[Hajek, 1988] Hajek, B. (1988). Cooling schedules for optimal annealing. *Mathematics of operations research*, 13(2):311–329.

[Hamzadayi and Yildiz, 2016] Hamzadayi, A. and Yildiz, G. (2016). Event driven strategy based complete rescheduling approaches for dynamic m identical parallel machines scheduling problem with a common server. *Computers & Industrial Engineering*, 91:66–84.

[Hong and Nelson, 2006] Hong, L. J. and Nelson, B. L. (2006). Discrete optimization via simulation using compass. *Operations Research*, 54(1):115–129.

[Hong and Nelson, 2007] Hong, L. J. and Nelson, B. L. (2007). A framework for locally convergent random-search algorithms for discrete optimization via simulation. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 17(4):19.

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

[Lee et al., 2006] Lee, W.-C., Wu, C.-C., and Chen, P. (2006). A simulated annealing approach to makespan minimization on identical parallel machines. *The International Journal of Advanced Manufacturing Technology*, 31(3-4):328–334.

[Lee and Pinedo, 1997] Lee, Y. H. and Pinedo, M. (1997). Scheduling jobs on parallel machines with sequence-dependent setup times. *European Journal of Operational Research*, 100(3):464–474.

[Leung and Pinedo, 2003] Leung, J. Y. and Pinedo, M. (2003). Minimizing total completion time on parallel machines with deadline constraints. *SIAM Journal on Computing*, 32(5):1370–1388.

[Lo and Bavarian, 1991] Lo, Z.-P. and Bavarian, B. (1991). Job scheduling on parallel machines using simulated annealing. In *Systems, Man, and Cybernetics, 1991.'Decision Aiding for Complex Systems, Conference Proceedings., 1991 IEEE International Conference on*, pages 391–396. IEEE.

[McCormick and Pinedo, 1995] McCormick, S. T. and Pinedo, M. L. (1995). Scheduling n independent jobs on m uniform machines with both flowtime and makespan objectives: a parametric analysis. *ORSA Journal on Computing*, 7(1):63–77.

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

[Megow et al., 2006] Megow, N., Uetz, M., and Vredeveld, T. (2006). Models and algorithms for stochastic online scheduling. *Mathematics of Operations Research*, 31(3):513–525.

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

[Nourani and Andresen, 1998] Nourani, Y. and Andresen, B. (1998). A comparison of simulated annealing cooling strategies. *Journal of Physics A: Mathematical and General*, 31(41):8373.

[Piersma and van Dijk, 1996] Piersma, N. and van Dijk, W. (1996). A local search heuristic for unrelated parallel machine scheduling with efficient neighborhood search. *Mathematical and Computer Modelling*, 24(9):11–19.

[Pinedo, 2015] Pinedo, M. (2015). *Scheduling*. Springer.

[Radhakrishnan and Ventura, 2000] Radhakrishnan, S. and 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.

[Ribas et al., 2010] Ribas, I., Leisten, R., and Framiñan, J. M. (2010). Review and classification of hybrid flow shop scheduling problems from a production system and a solutions procedure perspective. *Computers & Operations Research*, 37(8):1439–1454.

[Ruiz-Torres et al., 1997] Ruiz-Torres, A. J., Enscore, E. E., and Barton, R. R. (1997). Simulated annealing heuristics for the average flow-time and the number of tardy jobs bi-criteria identical parallel machine problem. *Computers & industrial engineering*, 33(1-2):257–260.

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

[Skutella et al., 2016] Skutella, M., Sviridenko, M., and Uetz, M. (2016). Unrelated machine scheduling with stochastic processing times. *Mathematics of operations research*, 41(3):851–864.

[Skutella and Uetz, 2005] Skutella, M. and Uetz, M. (2005). Stochastic machine scheduling with precedence constraints. *SIAM Journal on Computing*, 34(4):788–802.

[Statista, 2017] Statista (2017). Retail e-commerce sales worldwide from 2014 to 2020 (in billion u.s. dollars).

[Tavakkoli-Moghaddam et al., 2009] Tavakkoli-Moghaddam, R., Taheri, F., Bazzazi, M., Izadi, M., and 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.

[Tran et al., 2016] Tran, T. T., Araujo, A., and Beck, J. C. (2016). Decomposition methods for the parallel machine scheduling problem with setups. *INFORMS Journal on Computing*, 28(1):83–95.

[Van den Akker et al., 2000] Van den Akker, J., Hurkens, C. A., and Savelsbergh, M. W. (2000). Time-indexed formulations for machine scheduling problems: Column generation. *INFORMS Journal on Computing*, 12(2):111–124.

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

[Whitley, 1994] Whitley, D. (1994). A genetic algorithm tutorial. *Statistics and computing*, 4(2):65–85.

[Xu et al., 2013] Xu, J., Nelson, B. L., and Hong, L. J. (2013). An adaptive hyperbox algorithm for high-dimensional discrete optimization via simulation problems. *INFORMS Journal on Computing*, 25(1):133–146.

[Yigit et al., 2006] Yigit, V., Aydin, M. E., and Turkbey, O. (2006). Solving large-scale uncapacitated facility location problems with evolutionary simulated annealing. *International Journal of Production Research*, 44(22):4773–4791.

[Yildiz, 2017] Yildiz, U. (2017). Using data analytics and optimization for efficient e-warehousing. *Unpublished*.

[Yu et al., 2016] Yu, Y., Wang, X., Zhong, R. Y., and Huang, G. Q. (2016). E-commerce logistics in supply chain management: Practice perspective. *Procedia CIRP*, 52:179–185.