Erasmus University Rotterdam
Erasmus School of Economics

Performance Analysis of Online Algorithms for Logistics Models with Online Customer Selection

Bachelor Thesis
Econometrics and Operations Research

Seong Ho Lee
384055

Supervised by
Dr. Wilco van den Heuvel

Second assessor
R. B. O. Kerkkamp, MSc

July, 2016

Abstract

In this thesis, we review and extend the results of the recent paper "Supply Chain Management with Online Custmoer Selection" by Elmachtoub and Levi (2016). The authors study online versions of four logistics models with rejection option, where the supplier has a freedom to decide whether to serve or reject a customer request without having any prior knowledge of the future customer arrivals. In order to make the cost minimizing customer selections, they provide two unique online algorithms, *CopyCat Algorithm* and *StablePair Algorithm*, that apply the concept of repeated reoptimization of corresponding offline subproblems. They provide computational studies of both algorithms for two types of inventory control problems, Economic Lot Sizing Problem and Joint Replenishment Problem, with online customer selection (online ELSP-CS, online JRP-CS) as well as a Facility Location Problem with online customer selection (online FLP-CS).

We first reproduce the computational results of the CopyCat and the StablePair for the online ELSP-CS and the online JRP-CS. Then we consider three interesting applications of the CopyCat and the StablePair for (i) the online ELSP-CS and the online JRP-CS with production availability date, (ii) the online JRP-CS with a submodular cost structure, and (iii) an online version of Prize Collecting Travelling Salesman Problem (online PCTSP). Using the well-known competitive ratio framework, we demonstrate that our experiments for the online ELSP-CS and the online JRP-CS correctly reproduce the performance of the benchmark results. Moreover, we observe that the inclusion of production availability date substantially decreases the computation time of the CopyCat at a cost of increased competitive ratios. Also, we show that associating the submodular cost structure to the online JRP-CS largely increases the complexity for both CopyCat and StablePair. Finally, the CopyCat Algorithm on the online PCTSP show a comparable strength to the J-L online algorithm provided by Jaillet and Lu (2013) for a small number of requests, but significantly worse for large instances.

Contents

1 Introduction

The biggest challenges facing modern logistics industry involve satisfying customer demands, providing on-time delivery and minimizing operations cost. In order to alleviate these challenges, many practitioners and researchers attempt to model the logistics systems by forming appropriate optimization problems with the goal of maximizing the profits by minimizing the total production costs and/or maximizing the total sales revenues.

While most of classical optimization models require satisfying all market demands, the recent trend is to account for a flexible decision on the supply side as well. That is, supplier selects which customer to serve and matches its production capabilities. The freedom to accept or reject a customer's request extends the conventional logistics model to a model with *customer selection (CS)*. The profit maximizing supplier base his customer selection on his cost structure and the customer's demand characteristics, such as her order quantity and item type.

The goal of our thesis is to review and extend on the recent paper "Supply Chain Management with Online Custmoer Selection" by Elmachtoub and Levi (2016). Specifically, we consider their two interesting algorithms, which are designed to solve online versions of the logistics models with customer selection, where the supplier makes his accept/reject decision in real time. In the online setting, every customer request arrives one after another, and whenever a new request arrives the supplier needs to make an immediate accept/reject decision before receiving the next new request. This *selection step* is the first step from the two-step process of the online algorithm. The selection step ends when the supplier completes all his accept/reject decisions on every customer request. Then, the *production step* begins, whereby the supplier serves his accepted customers with the minimum production costs. The goal of the supplier is to increase his profits by minimizing the total cost from the customer rejection and the production cost incurred from serving the accepted customers. In this thesis, we focus on the cost minimization problems rather than the profit maximization problems since finding an optimal solution to the profit maximization problems is proven to be intractable (Van den Heuvel et al. 2012).

1.1 Thesis Outline and Contributions

Our research objective is to obtain the empirical performance of the two novel *online algorithms* developed by Elmachtoub and Levi (2016). To demonstrate their performance, we solve a class of different logistics problems with the variant of online customer selection. In particular, we consider two types of logistics problems, inventory control problem and vehicle routing problem, which we explain in more detail in the next few paragraphs.

First, we refer to an *online algorithm* as a policy that allows a supplier to make efficient online decisions in specific logistics models. Elmachtoub and Levi consider two such online algorithms, *CopyCat* and *StablePair*, which are applications of repeated reoptimization. They assist the supplier's accept/reject decision for the customer that just arrived based on his past decisions. More specifically, these algorithms make decisions for the online customer selection problem based on solving a problem with *offline customer selection*, which we define in the next paragraph, with respect to all customers that arrived thus far and assuming that no selection decisions have yet been made.

The performance of the two online algorithms is evaluated using the famous *competitive ratio* framework. The competitive ratio summarizes the relative quality of the solution provided by the online algorithms to the *optimal offline solution* that can be obtained if the supplier has all information about the future customer arrivals before making his first selection decision. If we denote the optimal offline solution as C^* and the online solution as C, a competitive ratio of α means that the following inequality holds: $C \leq \alpha C^*$, where α is a constant. The competitive ratio is most informative when the customer arrivals are generated by a worst-case adversary, who aims to maximize the ratio between the cost of the online and the optimal offline solution. That is, the competitive ratio of α guarantees that the cost of the online policies is at most the α times the optimal offline cost for any sequence of customers and their order quantities.

The first class of logistics models under our consideration is the *inventory control model*, which captures the uncertainty of how much to order to meet the market demand in each time period. In this thesis, we are interested in the uncapacitated inventory models whose cost structures have the economies of scale. This way, the supplier becomes more generous with accepting future customers and less with the current customer. Also, the economies of scale relate the marginal costs of accepting the current

customer to the customers accepted in the past, hence the supplier cannot separate one from another. For demonstration, we consider three different cases of such inventory control problems and apply the online customer selection framework to these problems.

Our first inventory control problem is the *economic lot sizing problem with online customer selection (online ELSP-CS)*. In the offline setting, the supplier must place production orders over a discrete planning horizon and meet all sequence of demands over the horizon. Each customer request is satisfied by the latest order prior to it due date. Placing an order requires a setup procedure, which incurs a constant setup cost irrespective of the amount of order. Holding an order in the inventory until the due date incurs a per time holding cost per every unit of order quantity. The offline ELSP-CS allows the supplier to reject customers at a per unit rejection cost. In this thesis, we look at the application of online customer selection, where the customers arrive one at a time with a due date and a demand quantity. The supplier either accepts or rejects the request immediately upon its arrival before the beginning of the production step. This decision is made without any prior knowledge about future customer arrivals. When all customers have arrived, one proceeds to the production step where the supplier optimally serves all the accepted customers.

The second inventory control problem is the online variant of the *multi-item joint replenishment problem with customer selection (online JRP-CS)*. This is a generalization of the online ELSP-CS with multiple item types. The setup costs consist of a joint setup cost and a type-specific setup cost. As with the online ELSP-CS, holding the demand quantity until the due date incurs a per unit, per time holding cost and rejecting a customer incurs the rejection cost. One can easily convert these costs into type-specific costs, but we do not consider them in this thesis.

Our last inventory control problem is the *cardinality joint replenishment problem with online customer selection (online CJRP-CS)*. The classical CJRP is a generalization of the traditional JRP with a strictly *submodular* joint setup cost with respect to the number of item types included in every order. We explain the concept of submodularity in Section 7. Cheung et al.(2015) introduced this NP-hard problem and provided an efficient algorithm with a constant approximation ratio. However, we are not aware of any literature that provide the computational study on the online CJRP-CS.

The second class of logistics models that we consider in this thesis is a vehicle routing problem. The single vehicle routing problem aims to plan an optimal route for a vehicle or a server to traverse all customer locations and deliver the requested quantity. Here, we consider an online version of a famous vehicle routing application known as *travelling salesman problem (TSP)*. In this NP-hard problem, a server begins his route from the origin, visit all customers located in a general metric space, and returns to the origin with the shortest travelling distance. A customer-selection variant of the TSP is called the *prize-collecting travelling salesman problem (PCTSP)*. In the offline PCTSP, the server is aware of all customers before he begins his tour. Rejecting to visit a customer incurs a penalty cost that represents the lost revenue estimated from the rejection cost and the demand quantity. We consider the online version of the PCTSP, which we refer as *online PCTSP*, where the requests arrive sequentially over time in a general metric space and a server makes a real-time decision on which customer to visit and in what order to visit them without prior knowledge of future requests.

We present our work in four big Parts, each containing relevant Sections. In Part I, we identify and describe the problem with literature review (Section 2), general notations (Section 3) and the idea of the CopyCat and the StablePair online algorithms (Section 4). In Part II, we provide the formulations of the relevant logistics models and provide the solution approaches using the CopyCat and the StablePair. In Part III, we present our main computational results for the logistics models with those online algorithms. We round our thesis with the conclusion in Part IV. In the next subsection, we explicitly describe what results have been replicated from the original work of Elmachtoub and Levi (2016).

1.2 Replication and Extensions

We first outline the results replicated from the paper "Supply Chain Management with Online Custmoer Selection" written by Elmachtoub and Levi (2016). First, we use their theorems and lemmas in order to use their online algorithms, CopyCat and StablePair. Next, we implement online algorithms to the same set of online inventory control problems with customer selection, online ELSP-CS and the online JRP-CS, to get as close results as theirs. We note that we therefore used the same experiment scenarios and simulation parameters.

Beyond our replications, we consider three extensions to their work. Our first extension associates the online ELSP-CS and the online JRP-CS with the production availability dates, where the supplier

can only setup an order between a specific time interval due to perishable products or semi-finished products that require extra processing time before setting up an order. It was suggested by N. Absi et al.(2011) in their study of uncapacitated lot sizing problem with production time window. Our second extension is the online Cardinality JRP-CS, which was not contained in the original paper. However, the authors do consider the submodular cost problems with online customer rejection. Our motivation comes from merging those two models and demonstrating their effects on performance. Lastly, and perhaps most interestingly, we consider the online PCTSP to experiment how their online algorithms perform in comparison to the contemporary best online algorithm.

Part I
Problem Description

2 Literature Review and Problem Motivation

The most relevant paper to this thesis is the original paper written by Elmachtoub and Levi (2016), which considers supply chain management with online customer selection. They prove that the CopyCat and the StablePair Algorithm are 3-competitive for the online ELSP-CS, and 4-competitive and 3-competitive for the online ELSP-CS and online JRP-CS, respectively.

In the offline context, there has been a stream of noteworthy literature on logistics models with customer selection. Bienstock et al. (1993) study the offline prize-collecting travelling salesman problem, providing an approximation algorithm with constant bound. Geunes et al. (2011) develop a general linear programming rounding framework to approximately solve several supply chain problems with offline market selection and stochastic demand. T'kindt and Croce (2012) consider two-machine flow-shop scheduling problem with rejection and with due date assignment.

Interesting findings in the context of online optimization in operations research include the work by Van den Heuvel and Wagelmans (2010), where they show the competitive ratio of the online economic lot sizing problem has a lower bound of two. Jaillet and Lu (2013) study the online PCTSP and propose an asymptotically best possible $O(\sqrt{\log n})$ in a general metric space with prior knowledge about the total number of requests n. Fotakis (2008) provides a lower and a customer-size-dependent upper bound for the online facility location problem. Finally, Garg et al. (2008) provide $O(\log n)$ asymptotic bound for the online Steiner tree problem.

The online algorithms, in general, have many real life applications. Unlike the classical approach of assuming a demand distribution to anticipate the future customer arrivals, online policies do not base their decisions on unknown future information, making it more robust to the demand volatility. Also, under many logistics settings, the decisions have to be made quickly and precisely, since otherwise the extra waiting time will incur a large sum of delay costs. It is especially severe in the case of multi-level logistics systems, where multiple parties would suffer from the delay.

3 General Notation

The models discussed in this thesis involve decisions that are made in two steps. The first step is the selection step, where the supplier decides on which customers to serve or reject. In stage k, customer k is observed by the supplier. Each customer brings her specific requirements I_k and the components of the requirements vary for different logistics problems. The requirements always include the demand quantity d_k, but may contain the due date t_k, item type i_k or location coordinates (x_k, y_k). Once a new customer is observed, the supplier has to either accept or reject customer k, while he can only leverage the information from the past customers until stage k. If customer k is rejected, a per unit cost of r_k is incurred, which corresponds with the lost revenues. If customer k is accepted, her requirements must be served in the production step, which would incur relevant production costs. The selection step is complete when the supplier does not observe any more new customers. At the end of the selection step, the customers are partitioned into groups of accepted and rejected customers denoted by A and Ω, respectively. It holds that for these sets defined over the first k customers, A_k and Ω_k: $A_k \cap \Omega_k = \emptyset$, $A_k \cup \Omega_k = U_k$, $A_{k-1} \subseteq A_k$ and $\Omega_{k-1} \subseteq \Omega_k$. The total rejection cost from the selection step is denoted by $R(\Omega)$.

The second step of the procedure is the production step. Let $Q \subseteq \mathbb{Q}$ be the set of production options the supplier can use to serve customers. The production options can represent various production settings. In case of the inventory control problems, the production option is the set of potential order dates. Define $U = \{1, ..., M\}$ to be thee the production costs by $P(B)$. In Sections 5-8 we specify the definitions for I_k, r_k and Q. set of all customers and $U_k \subseteq U$ denote the first k customers. (For convenience, we drop the subscript for stage M: $U_M = U$) Then, for a nonempty set of production options Q and a set of customers $B \subseteq U$, $P(Q, B)$ denotes the minimum possible production costs incurred to serve all customers in B by using the production options Q. When there is only one production option, such that $|\mathbb{Q}| = 1$, omit Q: $P(Q, B) = P(B)$. The online problem defined on the production options Q and the customers U_k is denoted by $\phi(Q, U_k)$, and as $\phi(U_k)$ when $|\mathbb{Q}| = 1$.

If the supplier has a prior information of future customers, one can solve the offline customer selection problem defined over a set of production options $Q \subseteq \mathbb{Q}$ and a set of customers U: $\phi^*(Q,U) = \min_{A \subseteq U}\{P(Q,A) + R(\Omega)\}$. If there is only one production option ($|\mathbb{Q}| = 1$), then we denote the offline problem as $\phi^*(U)$. The optimal offline cost with respect to first k stages is denoted by $C^*(U_k)$, which is equal to the sum of production and rejection costs of optimally chosen A^* and Ω^*. For the first k customers, we denote the sets as A_k^* and Ω_k^*, respectively.

4 Online Algorithms

In this section, we introduce the two online algorithms developed by Elmachtoub and Levi; CopyCat and StablePair Algorithm. Also, we briefly present their relevant theoretical results.

4.1 CopyCat Algorithm

The idea behind the CopyCat Algorithm is fairly straightforward. The CopyCat gets its name as it "copies" the optimal offline selection decision. It accepts the request from customer k if the respective offline problem defined over all customer requests observed thus far ($\phi^*(U_k)$) accepts the customer k ($k \in A_k^*$). If the respective offline problem rejects the customer k ($k \in \Omega_k^*$), then the CopyCat Algorithm also rejects the customer k and incurs a per unit rejection cost r. Therefore, the selection process for the CopyCat Algorithm involves solving a respective offline problem defined on all the observable set of customers at stage k ($\phi^*(U_k)$). The accepted and rejected customers by the CopyCat Algorithm are partitioned into the sets A_k^{CC} and Ω_k^{CC}, respectively.

CopyCat Algorithm
Accept current customer k if and only if $k \in A_k^*$.

The principle of the CopyCat Algorithm is intuitive and simple, and despite of its simple structure Elmachtoub and Levi show that the rejection cost for CopyCat will never be too large for any problem with online customer selection as long as the production cost $P(\cdot)$ is monotonically non-decreasing. Hence, it can be shown that $R(\Omega^{\text{CC}}) \leq C^*(U)$ (**Lemma 1**, proof in the appendix). If the production costs of CopyCat can be bounded by α times the optimal offline cost C^*, then CopyCat is $(\alpha + 1)$-competitive.

THEOREM 1
Let A^{CC} be all the customers that the CopyCat Algorithm accepts and let α be a positive scalar. If $P(A^{\text{CC}}) \leq \alpha C^*(U)$, then CopyCat is $(\alpha + 1)$-competitive.

The key feature of the CopyCat Algorithm is that in every new stage k, the policy solves the respective offline problem $\phi^*(U_k)$ that does not consider the set of previously accepted customers by the CopyCat Algorithm denoted by A_{k-1}^{CC} (i.e. we reoptimize the problem with the customers in U_k). If one fixes the previously made decisions during the reoptimization, the CopyCat Algorithm would have rejected most of the customers and result in poor performance.

A major drawback of the CopyCat Algorithm is that it requires solving the exact offline problem in each stage, which may be difficult to achieve when the offline problem itself is already NP-hard. Indeed, most of the problems considered in this thesis are NP-hard problems and CopyCat Algorithm may show poor quality in such problems. Therefore, Elmachtoub and Levi present another algorithm that is efficiently computable for a variety of inventory control problems.

4.2 StablePair Algorithm

The StablePair Algorithm requires finding a stable pair whenever a new customer arrives. A pair of production options $Q \subseteq \mathbb{Q}$ and customers $B \subseteq U$, (Q, B), is a stable pair if there exists an optimal solution to the respective offline customer selection problem defined on Q and B, denoted by $\phi^*(Q,B)$, that accepts all customers in B. Then the StablePair accepts customer k if $k \in B$. The accepted and rejected customers by the StablePair Algorithm are partitioned into the sets A_k^{SP} and Ω_k^{SP}, respectively.

StablePair Algorithm
Accept current customer k if and only if there exists a stable pair $(Q, B) \subseteq (\mathbb{Q}, U_k)$, such that $k \in B$.

As with the CopyCat Algorithm, Elmachtoub and Levi provide the theoretical bound for the StablePair rejection cost: $R(\Omega^{\text{SP}}) \leq R(\Omega^{\text{SP}} \cap \Omega^*) + P(A^*)$ (**Lemma 2**, proof in the appendix). Then using the above bound and a bound for the production cost, we can obtain strong competitive ratios that can be strictly better than CopyCat.

THEOREM 2
Let A^{SP} be all the customers that the StablePair Algorithm accepts and let α and γ be positive scalars. If there exists an optimal offline solution (A^*, Ω^*) such that the bound $P(A^{\text{SP}}) \leq \alpha P(A^*) + \gamma R(A^{\text{SP}} \cap \Omega^*) + R(A^{\text{SP}} \cap \Omega^*)$ holds, then StablePair is $\max\{\alpha + 1, \gamma + 1\}$-competitive.

Elmachtoub and Levi explain three main benefits of StablePair Algorithm over CopyCat Algorithm. The first benefit is the stronger bound of StablePair Algorithm on the rejection costs, and the second benefit is that the selection step takes polynomial time, supporting its computational efficiency compared to the CopyCat Algorithm. Lastly, the StablePair Algorithm does not "regret" its decision unlike the CopyCat Algorithm. The CopyCat Algorithm regrets its decision if the optimal solution of the offline subproblem $\phi^*(U_k)$ rejects any of the customers in A^{CC}_{k-1} at stage k. This is not true for the StablePair Algorithm since once a customer is accepted the original stable pair always accepts the customer in later stages.

Finally, it can be proved that the set of accepted customers for the CopyCat Algorithm is always contained in the set of accepted customers for the StablePair Algorithm. Let A^{CC} be the set of accepted customers by the CopyCat Algorithm and A^{SP} be that of the StablePair Algorithm. Then the accepted set of customers by StablePair is at least that of CopyCat. That is, $A^{\text{CC}} \subseteq A^{\text{SP}}$ (**Lemma 3**, proof in the appendix).

Despite the strong benefits of StablePair Algorithm in both empirical applications and theoretical bounds, implementing it to a general class of logistics models can be quite tricky. Elmachtoub and Levi provide efficient implementation steps for StablePair on online ELSP-CS and online JRP-CS, but we are not aware of efficient StablePair implementation steps for the online PCTSP or in any other variants of inventory control problems that we consider. This difficulty arises due to the abstract definition of production option Q. Hence, CopyCat Algorithm has a strict advantage over the StablePair Algorithm with its simple implementation.

4.3 Production Step

At the end of the CopyCat and/or StablePair selection step, the selection process outputs the set of accepted customers over all arrived customers $A \subseteq U$. Hence, we obtain the rejection costs after the selection step but not the production costs, yet.

In the production step, one solves the traditional logistics problem without the rejection option defined on the set of all accepted customers. For example, after completing the StablePair selection algorithm on the online ELSP-CS (with $A^{\text{SP}}_{\text{ELSP-CS}}$), the production costs are determined by solving the traditional ELSP defined on the set $A^{\text{SP}}_{\text{ELSP-CS}}$.

In Part II, we consider four online versions of logistics models with customer selection and outline our solution approaches based on the original work by Elmachtoub and Levi (2016). Note that in Part II, we only discuss the offline formulations and the StablePair implementation of those models, and we do not elaborate on CopyCat implementation or the production step. This is because one can easily implement both CopyCat and production procedure by solving a respective offline problem.

Part II
Logistics Models and Solution Approaches

5 Economic Lot Sizing Problem with Online Customer Selection

The *economic lot sizing problem (ELSP)* is a discrete time inventory model with a single item type. The objective of the ELSP is to serve a set of customers, each with a specific demand requirement, by a sequence of production orders over a planning horizon. Each order incurs a setup cost of K and the order date must be placed before the due date of the customer being served by that order. Every customer request is served by the latest order prior to its due date. After making an order, if the order date is strictly earlier than the due date of the served customer, the supplier receives a per unit holding cost h. The objective of ELSP without the customer selection is to minimize the total setup cost plus the total holding cost.

The ELSP with customer selection (ELSP-CS) allows the supplier to be more flexible with selecting the set of customers to be served. The supplier can reject a request at a per unit rejection cost of r and rejecting a customer k with demand quantity d_k would incur the cost of lost revenue rd_k. The goal of the ELSP-CS is to minimize the total rejection costs plus the total production costs for serving the accepted customers. This is known as the *ELSP with offline customer selection (offline ELSP-CS)*.

Unlike in the offline setting, the online ELSP-CS requires the supplier to immediately accept or reject a customer request without information about the future customer arrivals. This means that the supplier can only observe the requirements of the customers who already arrived and not those to arrive. This is different from the offline ELSP-CS where the supplier has all knowledge about the future customers upfront. The objective of the online ELSP-CS is to minimize the total production cost $P(A)$ and rejection costs $R(\Omega)$. Note that for the online ELSP-CS, the set of production options $Q \subseteq \mathbb{Q}$ is defined to be the potential order dates.

5.1 MIP Formulation of the Offline ELSP-CS

The exact optimal offline solution to the offline ELSP-CS is obtained from the mixed integer programming (MIP) formulation introduced by Geunes et al. (2011). The objective of the MIP formulation is to minimize the total production and the rejection costs throughout the discrete time horizon. In this section, we present the MIP formulation for the offline ELSP-CS developed by Geunes et al. (2011).

For each customer $j \in \{1, ..., M\}$, she has a quantity of demand with the due date of t, which is denoted by d_{tj}. Setting up an order at time s incurs the fixed cost of K_s. The total holding cost associated with satisfying the demand d_{tj} by an order placed at time s with a per unit holding cost of h_{st} is denoted by H_{stj}. This value can be computed from $H_{stj} = d_{tj}h_{st}$.

For each time period $s = \{1, ..., T\}$, let x_{stj} be a decision variable representing the proportion of demand d_{tj} satisfied from an order in period s. y_s is a binary decision variable that equals 1 if an order is placed at time s and 0 otherwise. Finally, the binary decision variable z_j is equal to 1 if the customer is selected and 0 otherwise.

The MIP formulation of the offline ELSP-CS is defined over a set of all customers with size M and it is denoted by $\phi_{\text{ELSP-CS}}(M)$.

Problem $\phi_{\text{ELSP-CS}}(M)$

$$\text{minimize} \quad \sum_{t=1}^{T}\sum_{j=1}^{M}(1-z_j)rd_{tj} + \sum_{s=1}^{T}K_s y_s + \sum_{j=1}^{M}\sum_{t=1}^{T}\sum_{s=1}^{t}H_{stj}x_{stj} \tag{1}$$

$$\text{subject to} \quad \sum_{s=1}^{t}x_{stj} = z_j \qquad j=1,...,M; t=1,...,T, \tag{1.1}$$

$$x_{stj} \leq y_s \qquad j=1,...,M; t=1,...,T; s=1,...,t, \tag{1.2}$$

$$0 \leq x_{stj} \leq 1 \qquad j=1,...,M; t=1,...,T; s=1,...,t, \tag{1.3}$$

$$y_s, z_j \in \{0,1\} \qquad j=1,...,M; s=1,...,t. \tag{1.4}$$

The objective function (1) minimizes the sum of total rejection costs, the setup costs, and the holding costs. The first constraint (1.1) defines that if the customer j is accepted, then her demand must be completely satisfied. The constraint (1.2) imposes that if an order is set at time s, the binary decision variable y_s is equal to 1 as x_{stj} will be larger than 0 for some $s \in \{1,...,t\}$. The last two constraints (1.3) and (1.4) define the range of the decision variables x_{stj}, y_s and z_j.

5.2 StablePair Algorithm for the Online ELSP-CS

The StablePair algorithm accepts the customer k if there is a stable pair (Q, B) where Q is the set of production options (the order dates) and B is a set of customers, such that $k \in B$. Algorithm 1 outlines the selection process, which first identifies the stable pair for the online ELSP-CS and then make the accept/reject decision for the current customer k. The idea of the StablePair is to reduce the search space for stable pairs by only considering those with one production option, i.e. a single order date.

The StablePair implementation in Algorithm 1 applies the property of zero inventory ordering (ZIO) and the results of two lemmas provided by Elmachtoub and Levi (2016). Wagner and Whitin (1958) show that the optimal solution for the classical ELSP has a ZIO property and therefore the orders are placed when the inventory level hits zero. Based on the ZIO property of the optimal offline ELSP solution, one can see that every order serves all requests whose due dates are between the order date and the due date of the last customer that is served by that order ("last due date"). The *setup interval* of the order is the interval between the order date and the last due date for that order. Then, Elmachtoub and Levi prove two lemmas stating that the maximum size of the setup interval is r/h (**Lemma 4** in appendix) and that the the setup intervals from the StablePair solution and the optimal offline solution intersect (**Lemma 5** in appendix). Algorithm 1 uses these results.

Algorithm 1 Online ELSP-CS StablePair Selection Algorithm

Input: Order costs (K, h), rejection cost (r), a set of customers observable (U_k) and customer demand $(d_k$ where $k \in U_k)$

Output: Binary decision variable z_k that equals 1 if the customer k is accepted and 0 otherwise.

1: Initialize: $z_k = 0$
2: **Outer Loop:**
3: **for** Every due date t of each customer in U_k **do**
4: Set $Q = \{t\}$
5: Define set $D = \{j \in U_k : t_j \in [t, t + r/h]\}$
6: Define condition A: $R(D) \geq K + \sum_{j \in D} h(t_j - t)d_j$
7: **if** condition A is true **then**
8: $z_k = 1$
9: Exit **Outer Loop**
10: **end if**
11: **end for**
12: Output z_k

Due to line 4, the production option Q contains a single due date t. Then line 5 defines a set of customers D whose due dates are in the setup interval of the order at Q. Line 6 defines the condition to accept customer k if serving all customers in D is cheaper than or equal to rejecting them. Finally, the same process is repeated for every customer in U_k until the accepting condition is satisfied. If satisfied, the algorithm outputs the binary variable z_k that is equal to 1 and completes the algorithm.

The ZIO property implies that the potential production options (order dates) are the due dates of U_k since the inventory level is equal to zero on some due dates of customers. This property effectively reduces the search space for the stable pairs - there are at most N candidates of the stable pairs, and Algorithm 1 takes $O(N)$ time. Elmachtoub and Levi (2016) prove that both the StablePair and the CopyCat for the online ELSP-CS are 3-competitive.

5.3 Online ELSP-CS with production availability date

An interesting extension of the ELSP-CS is to include the *production availability date* for serving customer k, which is denoted by e_k. The production availability date is the earliest date in which the supplier can set an order to serve the customer k before her due date. In many real life settings, the logistics system applies the framework of production availability dates. As suggested by N. Absi et al. (2011), the availability date is implicitly and explicitly modelled for perishable products or items that involve semi-finished products in their production. Although the availability date is not observable from information set $I_k = \{d_k, t_k\}$ that a customer specifies upon her arrival, one can assume that the experienced supplier can correctly estimate the availability date from the customer's information set. Indeed, under the lot-sizing model, the supplier only receives demand requests of a single item type throughout the whole planning horizon. It is likely to be that an experienced supplier can correctly estimate the availability date through his repeated practices. This also implies that the size of interval $[e_k, t_k]$ will be the same for all customers under the single-item ELSP-CS model. In addition, we note that including the production availability date largely improves the computation time of the CopyCat Algorithm as it effectively reduces the search space for the optimal order dates by reducing the number of candidate order dates. For notational convenience, we include the production availability date in the customer's information set, $I_k = (d_k, e_k, t_k)$. Then, the supplier can only serve customer k by an order placed on the interval $[e_k, t_k]$.

The inclusion of production availability date can be modelled by replacing the constraint (1.2) by two additional extra constraints (1.2.1) and (1.2.2):

$$x_{stj} \leq y_s \qquad j = 1, ..., M; t = 1, ..., T; s = e_j, ..., t, \qquad (1.2.1)$$
$$x_{stj} = 0 \qquad j = 1, ..., M; t = 1, ..., T; s = 1, ..., (e_j - 1). \qquad (1.2.2)$$

Two constraints ensure that the customer k is served by an order placed at or after the availability date e_k. Note that since we do not have an efficient implementation plan for the StablePair Algorithm,

we only demonstrate the performance of the CopyCat Algorithm for this variant.

6 Joint Replenishment Problem with Online Customer Selection

The *joint replenishment problem (JRP)* is a variant of the ELSP with multiple item types i indexed from 1 to N. The information set of customer k includes the item type i_k. Each order incurs a joint setup cost of K_0 and for each item type i ordered, an item setup cost of K_i is incurred. Unlike the ELSP, each customer can only be served by orders that contain her item type and the order date must be placed before her due date. Again, each customer is served by the last order prior to her due date. After an order is made, there is a per unit holding cost h for all types of items to carry in inventory from the order date to the due date of the customer that is served by the order. The objective of JRP without the rejection option is to minimize the total ordering cost plus the holding cost. Arkin et al. (1989) show that this problem is NP-hard.

The JRP with customer selection (JRP-CS) enables the supplier to reject a request with any type of item at a per unit rejection cost of r and rejecting a customer k with demand quantity d_k would incur the cost of lost revenue: rd_k. The objective of the JRP-CS is to minimize the total rejection costs plus the cost of serving the accepted customers. This is known as the *offline JRP-CS*. This problem was studied by Geunes et al. (2011) and they further gave a 2.35-approximation algorithm that works efficiently for a large problem instance.

The online JRP-CS is the multi-item online ELSP-CS. The objective of the online JRP-CS is to minimize the total production cost of accepted customers $P(A)$ and the total rejection costs $R(\Omega)$. As with the online ELSP-CS, the production options Q are defined to be the order dates.

6.1 MIP Formulation of the Offline JRP-CS

The exact optimal offline solution to the JRP is obtained from the mixed integer programming (MIP) formulation provided by Geunes et al. (2011).

For each customer $j = 1, ..., M$, she has a quantity of demand for item of type i with the due date of t, which is denoted by d_{itj}. Setting up an order for all types of items at time s incurs the fixed cost of K_{0s} as well as a type-specific setup cost of K_{is}. The total holding cost associated with satisfying the demand d_{itj} by an order placed at time s with the per unit holding cost of h_{st} is denoted by H_{istj} : $H_{istj} = d_{itj}h_{st}$. Rejecting a customer incurs a per unit rejection cost of r.

For each time point $s = 1, ..., T$, let x_{istj} be the proportion of demand d_{itj} satisfied from an order in period s. Let y_{0s} be the binary variable that equals 1 if an order is placed at time s and 0 otherwise. Similarly, let the binary variable y_{is} equal to 1 if a type i item is order is placed at time s. Finally, the binary decision variable z_j is equal to 1 if customer j is accepted and 0 otherwise.

The MIP formulation of the offline JRP-CS is defined over a set of all customers with size M and it is denoted by $\phi_{\text{JRP-CS}}(M)$.

Problem $\phi_{\text{JRP-CS}}(M)$

$$\text{minimize} \quad \sum_{t=1}^{T}\sum_{i=1}^{N}\sum_{j=1}^{M}(1-z_j)rd_{itj} + \sum_{s=1}^{T}K_{0s}y_{0s} + \sum_{i=1}^{N}\sum_{s=1}^{T}K_{is}y_{is} + \sum_{j=1}^{M}\sum_{i=1}^{N}\sum_{t=1}^{T}\sum_{s=1}^{t}H_{istj}x_{istj} \tag{2}$$

$$\text{subject to} \quad \sum_{s=1}^{t}x_{istj} = z_j \qquad j=1,...,M; i=1,...,N; t=1,...,T, s=1,...,t \tag{2.1}$$

$$x_{istj} \leq y_{is} \qquad j=1,...,M; i=1,...,N; t=1,...,T; s=1,...,t, \tag{2.2}$$

$$x_{istj} \leq y_{0s} \qquad j=1,...,M; i=1,...,N; t=1,...,T; s=1,...,t, \tag{2.3}$$

$$0 \leq x_{istj} \leq 1 \qquad j=1,...,M; i=1,...,N; t=1,...,T; s=1,...,t, \tag{2.4}$$

$$y_{is}, z_j \in \{0,1\} \qquad i=0,...,N; j=1,...,M; s=1,...,t. \tag{2.5}$$

The objective function (2) minimizes the sum of total rejection costs, the joint and the type-dependent setup costs, and the holding costs. The first constraint (2.1) defines that if the customer j is accepted, then her demand must be completely satisfied. The constraints (2.2) and (2.3) impose that if a type i order is made at time s, y_{is} and y_{0s} must equal to 1. The last two constraints (2.4) and (2.5) define the range of the decision variables x_{istj}, y_{is} and z_j.

6.2 StablePair Algorithm for the Online JRP-CS

The StablePair algorithm accepts the customer k if there is a stable pair (Q, B) where Q is the set of production options (the order dates) and B is a set of customers, such that $k \in B$. Algorithm 3 in appendix outlines the selection process, which first identifies the stable pair for the online JRP-CS and then make the accept/reject decision for the current customer k. It also uses the ZIO property and the results of Lemma 4 and Lemma 5 as with the online ELS-CS - Elmachtoub and Levi (2016) show that both lemmas also hold for the online JRP-CS.

Again, the ZIO property implies that the potential production options (order dates). The set of potential stable pairs is at most N and the algorithm takes $O(MN)$ time. Elmachtoub and Levi (2016) prove that the StablePair is 3-competitive and the CopyCat is 4-competitive for the online JRP-CS.

6.3 Online JRP-CS with production availability date

Again, we consider the extension of associating the production availability date to customer demands for the online JRP-CS. This means that the information set for customer k will be $I_k = (i_k, d_k, t_k, e_k)$ with the availability date e_k for every customer. The supplier can only serve type i customer k by a type i order placed in the interval $[e_k, t_k]$. However, we note that our earlier assumption on the supplier's "correct estimation" of the production availability date for any demand requests is less plausible for the multi-item JRP-CS model. When the supplier deals with a large number of item types N, the supplier's estimation is less likely to be correct in the real life scenario. However, we believe that our framework is still plausible when the number of item types are kept at a moderate size. We further note that the the size of the interval $[e_k, t_k]$ is the same for all customers with the same item type, while the interval for different item types needs not be the same. Again, one of the motivations for including the production availability date is to improve the computation time of the CopyCat Algorithm by reducing the number of candidate order dates.

The inclusion of production availability date can be modelled by replacing the constraint (2.2) by two additional extra constraints (2.2.1) and (2.2.2):

$$x_{istj} \leq y_{is} \qquad j = 1, ..., M; i = 1, ..., N; t = 1, ..., T; s = e_j, ..., t, \tag{2.2.1}$$
$$x_{istj} = 0 \qquad j = 1, ..., M; i = 1, ..., N; t = 1, ..., T; s = 1, ..., (e_j - 1). \tag{2.2.2}$$

Two constraints ensure that the type i order to serve the type i customer k must be at or after the arrival time of the customer e_k.

7 Cardinality Joint Replenishment Problem with Online Customer Selection

The *cardinality joint replenishment problem (CJRP)* is a generalization of classical JRP that assumes for a *submodular* cost structure. A function $g(\cdot)$ is *submodular* if for all $S, T \subseteq U$, $g(S)+g(T) \geq g(S \cap T)+g(S \cup T)$. For the CJRP, the joint setup cost is a submodular function of the cardinality of the subset of item types being ordered (Cheung et al. 2015). Simply put, the submodular joint order setup cost function models the decreasing marginal cost as the order contains more item types - capturing the economies of scale in containing more item types in orders. The CJRP formulation is identical to the JRP formulation with its joint setup cost K_0 replaced by a nondecreasing, nonnegative and submodular function $g(\cdot)$ of the number of item types included in an order. Arkin et al. (1989) show that the cardinality JRP is NP-hard.

7.1 MIP Formulation of the Offline CJRP-CS

The MIP formulation of offline CJRP-CS is defined over a set of all customers with size M, which is denoted by $\phi_{\text{CJRP-CS}}(M)$. The MIP formulation uses the same notation for the sets, parameters and decision variables as those defined for offline JRP-CS in Section 6.

Based on the formulation provided by Cheung et al.(2015), we introduce new notation for the offline CJRP-CS. Recall that the index for item type i belongs to the set $\{1, ..., N\}$. Then, let $b \in \{0, ..., N\}$ indicate the number of item types within an order and let q_{sb} denote a strictly submodular function of the number of item types b. Without loss of generality, assume that $g(0) = 0$. Also, we define a new binary decision variable q_{sb} that equals to 1 if the number of item types in an order placed at time s is larger than or equal to b and 0 otherwise.

Problem $\phi_{\text{CJRP-CS}}(M)$

$$\text{minimize} \quad \sum_{t=1}^{T}\sum_{i=1}^{N}\sum_{j=1}^{M}(1 - z_j)rd_{itj} + \sum_{s=1}^{T}\sum_{b=1}^{N}(g(b) - g(b-1))q_{sb} + \sum_{i=1}^{N}\sum_{s=1}^{T}K_{is}y_{is} + \sum_{j=1}^{M}\sum_{i=1}^{N}\sum_{t=1}^{T}\sum_{s=1}^{t}H_{istj}x_{istj} \tag{3}$$

$$\text{subject to} \quad \sum_{s=1}^{t}x_{istj} = z_j \qquad j = 1, ..., M; i = 1, ..., N; t = 1, ..., T, \tag{3.1}$$

$$x_{istj} \leq y_{is} \qquad j = 1, ..., M; i = 1, ..., N; t = 1, ..., T; s = 1, ..., t, \tag{3.2}$$

$$q_{s(b+1)} \leq q_{sb} \qquad s = 1, ..., t; b = 0, ..., (N-1), \tag{3.3}$$

$$\sum_{i=1}^{N}y_{is} = \sum_{b=1}^{N}q_{sb} \qquad s = 1, ..., t, \tag{3.4}$$

$$0 \leq x_{istj} \leq 1 \qquad j = 1, ..., M; i = 1, ..., N; t = 1, ..., T; s = 1, ..., t, \tag{3.5}$$

$$x_{istj}, y_{is}, z_j, q_{sb} \in \{0, 1\} \qquad i = 1, ..., N; b = 0, ..., N; j = 1, ..., M; s = 1, ..., t, \quad t = 1, ..., T. \tag{3.6}$$

The second component of the objective function (3) sums up the marginal costs for including an additional item type in an order at time s. The constraints (3.3) and (3.4) define the variable $q_{s,b}$. (3.3) states that when an order contains $N-1$ item types, q_{sb} for every $b \in \{0, N-1\}$ is equal to 1 but 0 for $b = N$. (3.4) relates y_{is} to q_{sb}.

7.2 StablePair Algorithm for the Online CJRP-CS

One can use the StablePair Algorithm for the online JRP-CS (Algorithm 3) for the online CJRP-CS after making following adjustments: (i) input the submodular function $g(b)$ instead of K_0 in Algorithm 3, (ii) replace the condition B in line 13 with $R(D) \geq g(|X|) + \sum_{i \in X}(K_i + \sum_{j \in D_i} h(t_j - t)d_j)$. These two steps simply replace the initial joint set up cost K_0 by a submodular function with respect to the number of item types within an order $g(b)$.

8 Prize Collecting Travelling Salesman Problem with Online Demand Selection

The *prize collecting travelling salesman problem (PCTSP)* is a type of vehicle routing problem where the goal is to find the shortest route, which begins and ends at the origin, that visits a set of selected customers. Each customer j specifies demand quantity and her location in 2-dimensional Euclidean coordinates ((x, y)-coordinate). In the offline setting, the server who begins his route from the origin O has all information about the customers upfront and selects a set of customers to serve. Moving from the location of customer j_1 (we also use j_1 to denote the location of j_1 for convenience) to the location of j_2 incurs a travelling cost v_{j_1,j_2} that is equal to the Euclidean distance between the two locations ($v_{j_1,j_2} = \sqrt{(x_{j_1} - x_{j_2})^2 + (y_{j_1} - y_{j_2})^2}$). Rejecting a customer incurs a per unit rejection cost r on the demand quantity d_j. Bienstock et al. (1993) show that the offline PCTSP is an NP-hard problem.

As with the previous models, the online version of PCTSP assumes that customers are released one by one and must be either accepted or rejected by the server immediately. When a customer k is observed, her requirements $I_k = (d_k, x_k, y_k)$ are revealed to the server, where d_k is the demand quantity, x_k and y_k are her x and y-coordinates.

8.1 Flow Formulation of the Offline PCTSP

The exact optimal offline solution to the PCTSP is obtained from the Single Commodity Flow formulation provided by Garvish et al. (1978). Note, however, that their formulation is designed for the asymmetric TSP (ATSP). In this section, we present the PCTSP-adjusted flow formulation.

Let j_1 and j_2 be two different indices for customers $j_1, j_2 \in \{1, ..., M\}$, where $j_1 \neq j_2$. Each customer j has information set $I_j = \{d_j, xx_j, yy_j\}$, where d_j is the demand quantity, xx_j and yy_j are the x, y-coordinates of the customer j (we use double x and y to avoid confusion with the decision variables). For convenience, we also use j_1 and j_2 to represent the locations of customer j_1 and j_2. Then, v_{j_1,j_2} is the Euclidean distance between j_1 and j_2. Let x_{j_1,j_2} be a binary variable that equals to 1 if the route visits customers j_1 and j_2, 0 otherwise. y_{j_1} is a binary variable that equals to 0 if customer j_1 is served, and 1 otherwise. Finally, f_{j_1,j_2} denotes the flow variable that represents the flow in the arc between customer j_1 and j_2. The flow variable also represents the number of remaining customers that the server needs to visit.

The IP flow formulation of the offline PCTSP is defined over a set of all customers with size M, which we denote by $\phi_{\text{PCTSP}}(M)$. We denote the origin by the index $j = 0$, with $d_0 = 0$.

Problem $\phi_{\text{PCTSP}}(M)$

$$\text{minimize} \quad \sum_{j_1=0}^{M} \sum_{\substack{j_2=0 \\ j_2 \neq j_1}}^{M} v_{j_1,j_2} x_{j_1,j_2} + \sum_{j_1=1}^{M} r d_{j_1} y_{j_1} \tag{4}$$

subject to

$$\sum_{\substack{j_1=0 \\ j_1 \neq j2}}^{M} x_{j_1,j_2} = 1 - y_{j_1} \qquad j_1, j_2 = 1, ..., M, \tag{4.1}$$

$$\sum_{\substack{j_2=0 \\ j_2 \neq j_1}}^{M} x_{j_1,j_2} = 1 - y_{j_2} \qquad j_1, j_2 = 1, ..., M, \tag{4.2}$$

$$f_{j_1,j_2} \leq M - 1 \qquad j_1, j_2 = 1, ..., M, \tag{4.3}$$

$$\sum_{j_2=1}^{M} f_{0,j_2} = \sum_{j_1=1}^{M} (1 - y_{j_1}) \qquad j_1, j_2 = 1, ..., M, \tag{4.4}$$

$$\sum_{\substack{j_1=1 \\ j_1 \neq j_2}}^{M} f_{j_1,j_2} - \sum_{\substack{k=1 \\ k \neq j_2}}^{M} f_{j_2,k} = (1 - y_{j_2}) \qquad j_1, j_2, k = 1, ..., M, \tag{4.5}$$

$$f_{j_1,j_2} \geq x_{j_1,j_2} \qquad j_1, j_2 = 1, ..., M, \tag{4.6}$$

$$x_{j_1,j_2}, y_{j_1} \in \{0, 1\} \qquad j_1, j_2 = 1, ..., M. \tag{4.7}$$

$$y_0 = 0 \tag{4.8}$$

The objective function (4) minimizes the sum of travelling costs and the rejection costs. The constraints (4.1) and (4.2) implement that the route visits every selected customer.

Perhaps the most important set of constraints in this formulation are the constraints (4.3)-(4.6). They are *subtour elimination constraints (SEC)*, which prevent the optimal tour from creating routes among a strict subset of all selected customers. In other words, it ensures that there is only one tour that begins and ends at the origin and visits all the accepted customers exactly once. The main role of the flow variables in the SEC is to countdown the number of remaining customers who were accepted but not visited yet. Constraint (4.3) defines the upper bound value for the flow variable. (4.4) ensures that the flow from the origin to all the accepted customers (or the number of accepted but not yet visited customers when the server is at the origin) is exactly equal to the number of accepted customers. (4.5) states that every visited customer deducts a unit of flow from the amount of flow in. This way, the subtour can be eliminated as the flow value must equal to 0 when the server returns to the origin. Constraint (4.6) ensures that the value of the flow variable in the travelled arc must be larger than or equal to 1. (4.7) defines the binaary decision variables and (4.8) ensures that the origin is always accepted by default.

8.2 Jaillet-Lu Online Algorithm for the Online PCTSP

While the implementation of the CopyCat Algorithm to the online PCTSP is straightforward, we do not know how to implement the StablePair Algorithm for the online PCTSP. For the purpose of performance comparison, we implemented the best known online algorithm for the online PCTSP that is provided by Jaillet and Lu (2013). In this thesis, we call this algorithm as *J-L Algorithm*.

A necessary assumption for the J-L algorithm to work is that the decision maker has a priori knowledge on the total number requests M. Although this assumption may appear to be unrealistic, it is still plausible as a travelling server may (at least roughly) be aware of the total number of customers that he needs to visit, while not knowing their exact locations and demand quantities.

The original J-L Algorithm provided by Jaillet and Lu (2013) is composed of two steps: decision making and route construction step. However, as we have different problem settings from theirs, our implementation is slightly different. In particular, the original J-L Algorithm assumes that the server can

revert all its previously made online decisions (i.e., $A_{k-1}^{\text{J-L}} \subseteq A_k^{\text{J-L}}$ may not hold at some stage k). Then, it constructs a new route with the updated online decisions. As this is not applicable under our problem setting, we remove the possibility of reverting online decisions and replace the route construction step by solving a respective offline TSP defined on set of accepted customers until stage k, $A_k^{\text{J-L}}$. These adjustments are modifications of the original J-L Algorithm and the resulting algorithm is still the J-L algorithm as it does not violate the necessary conditions for the original J-L Algorithm.

Algorithm 2 outlines the selection process of the J-L Algorithm.

Algorithm 2 Online PCTSP J-L Selection Algorithm

Input: Euclidean coordinates of customer location (xx_k, yy_k), the number of customers known a priori (M), and a per unit rejection cost (r).
Output: The set of accepted customers (A_M) and rejected customers (Ω_M).

1: Initialize: $\Omega_0 = \emptyset$, $A_0 = \{0\}$, and k = 1.
2: **while** $k < M$ **do**
3: Compute $v(\{k\}, V_{k-1}) = \inf\{\sqrt{(xx_k - xx_j)^2 + (yy_k - yy_j)^2)} : j \in V_{k-1}\}$
4: Accept request k if and only if $v(\{k\}, V_{k-1}) \leq r\sqrt{\log M}$
5: **if** Request k is accepted **then**
6: $\Omega_k = \Omega_{k-1}$
7: $A_k = A_{k-1} \cup \{k\}$.
8: **else**
9: $\Omega_k = \Omega_{k-1} \cup \{k\}$
10: $A_k = A_{k-1}$.
11: **end if**
12: $k \leftarrow k + 1$
13: **end while**
14: Output Ω_M and A_M

Line 1 initializes the set of accepted and rejected customers. Line 3 computes the shortest distance from the location of current customer k to the set of accepted customers until stage $k-1$. Then, line 4 specifies the condition for selecting customer k. Lines from 5 to 12 update the sets and the customer. When all selection decisions have been made, the algorithm outputs the set of accepted and rejected customers. Jaillet and Lu prove that the above selection algorithm is the best possible $O(\sqrt{\log M})$-competitive algorithm in a general metric space without release dates of the customers.

In Part 3, we demonstrate the empirical performance of the two online algorithms on the defined set of logistics models in Part 2.

Part III
Empirical Competitive Ratio

In Part III, we present our computational results for the CopyCat and StablePair Algorithms. In addition, we consider the StablePair Algorithm with a scaling factor of two. This means that we apply the StablePair selection policy with a per unit rejection cost r multiplied by the factor of two. The scaling technique is based on the idea of hedging against the future by using a higher rejection cost, softening the condition for accepting a request. The empirical performances of these three algorithms were evaluated for every logistics model described in PartII with the competitive ratio framework. That is, whenever a new customer k arrives, we calculated the performance ratio: $C(U_k)/C^*(U_k)$ which compares the online cost to the optimal offline cost that is impossible to achieve in the online setting. For all three algorithms, we conducted the experiments using three different scenarios, where in each scenario the sequence of arriving customers have different characteristics. The first scenario is "Conservative" scenario where all requests have a unit demand quantity ($d_k = 1$). It is conservative in the sense that all arriving requests have the homogeneous demand quantity. The second "More Demands" scenario simulates the demand quantity by an integer value drawn uniformly at random between 1 and 10 ($d_k \sim$ UNI$(1, 10)$). The last "Large Orders First" scenario also draws the demand quantity from a uniform random distribution between 1 and 10, but the first two orders are very large orders with due dates placed at times 1 and 15 ($d_k = 100$ for $k = 1, 2$). It simulates the situation where the supplier has two regular customers making large orders, and hence rejecting their requests is extremely expensive. For all of the three scenarios, the due dates of the customers are drawn from a uniform distribution over the discrete time horizon from 1 to 30. We used uniformly random due dates since the supplier cannot make a logical guess on when the next request will arrive.

For each logistics model, we present our outcome in three figures, each containing the average performance ratios for each scenario, and in three tables containing the maximum performance ratio, average final performance ratio and the average computation times (in seconds) during the selection and production step, over all runs. We made these runs with a computer using Intel Core(TM) i7-3610QM CPU 2.30GHz with 8.00 GB RAM.

9 Online ELSP-CS

For all three scenarios, we used the following simulation parameters to simulate an instance of the online ELSP-CS. We set the fixed setup cost $K = 100$, a per unit time holding cost $h = 1$ and a per unit rejection cost $r = 5$. The discrete planning horizon is a month in days: $[1, 30]$. Note that these parameters are equal to the parameters used in the paper by Elmachtoub and Levi, which means we expect to achieve nearly identical empirical competitive ratios for all three online algorithms. We performed the analysis for 100 independent runs, each with 300 simulated customer arrivals.

The outcomes from the experiments are contained in three figures and three tables. Figure 1, 2 and 3 contain the average competitive ratios over 100 runs for all three online algorithms, for each scenario. Tables from 1 to 3 contain the maximum ratios, average final ratios and average computational times over all runs.

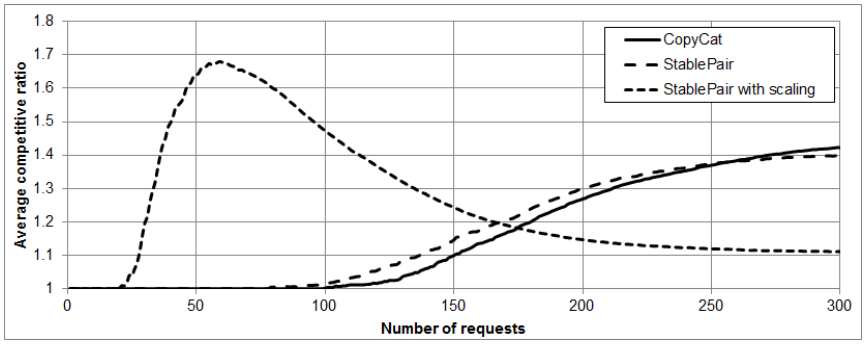

Figure 1: ELSP-Conservative scenario

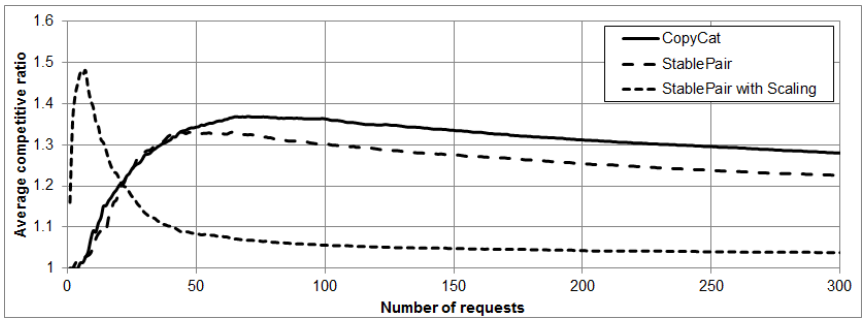

Figure 2: ELSP-More Demands scenario

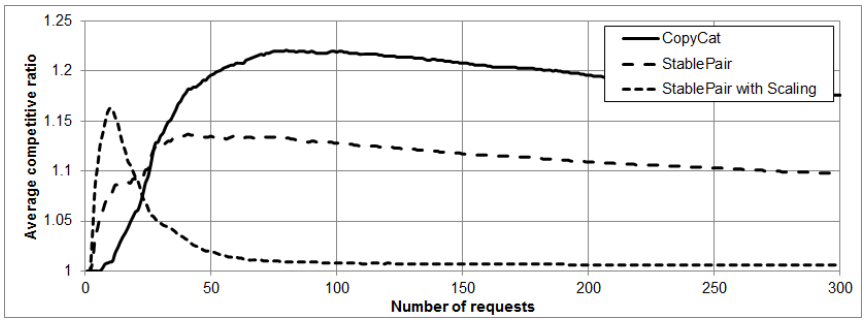

Figure 3: ELSP-Large Orders First scenario

The results from the Conservative scenario give us an impression that the CopyCat and the StablePair algorithms tend towards the upper bound of 1.5. For the first 80 requests, both online algorithms and offline solution reject all customers since there are not enough requests to encourage production. That is, due to the small number of requests the supplier is better to reject all of them, than to incur an expensive setup cost. Then the ratio begins to increase for the StablePair from around 100 customer arrivals (around 125 requests for the CopyCat) as the offline solution starts to accept some requests. The ratios further increase at a moderate slope as more customers arrive and the effect of economies of scale begins to visualize. Finally, the ratios stabilize as the offline solution, CopyCat and StablePair algorithms accept every new request when the requests are in abundance – the marginal cost of serving a customer became small with the economies of scale. Notice that the StablePair ratio gets below the CopyCat ratio at around 270 customer arrivals. It is because all customers who are accepted by the CopyCat are also accepted by the StablePair according to Lemma 1, explaining its winning performance as the number of customers becomes large. A more dramatic behavior occurs with the ratios for StablePair with scaling. In the Conservative scenario, it begins to accept requests at the early stage of 25 customer arrivals, causing the sharp rise in the ratios until the 60^{th} customer arrival. But in later stages its ratios are significantly lower than

for the other two algorithms, supporting the effect of hedging for the future customer arrivals. For More Demands scenario, the CopyCat and the StablePair accept customers at earlier stages than in the Conservative scenario, therefore having their peaks at around stage 50. Lastly, the Large Orders First graph shows that the gap between CopyCat and StablePair is noticeably large, indicating the poor performance of CopyCat in coping with large orders under online ELSP-CS model.

The maximum performance ratios in Table 1 show that the CopyCat and StablePair achieve similar upper bound ratios, which are below 2 for all three scenarios. This is much lower than the theoretical bound of 3 as suggested by Elmachtoub and Levi (2016). However, their average computation times through all 100 runs are the key differentiators between the two algorithms. In the Conservative scenario, the CopyCat requires nearly 8 times longer run time than the StablePair. This makes the StablePair much more appealing in practice.

Comparing our results to the results of Elmachtoub and Levi, the peaks are placed at around the same height (1.7 for Conservative scenario) and CopyCat/StablePair ratios begin to increase from 1 at around the same stage. However, one can see that the graph for the Large Orders First scenario show remarkably different outcomes. Our graph has the peaks and the final ratios which are substantially higher than their graph (by 0.05). Also, one can see that in our graph, StablePair takes the performance ratio different from 1 when there are 3 customer arrivals. This is certainly not the case in the graph of Elmachtoub and Levi, because in their graph, the StablePair ratios depart from 1 at around the same number of requests as the CopyCat Algorithm. The maximum performance ratios are different by at most a factor of 0.20, which is, although large, explainable by the randomness of the simulated demand quantities.

9.1 Online ELSP-CS with production availability date

A natural extension of online ELSP-CS is to associate each request with a production availability date. We simulate the availability date e_k for customer k by setting it equal to 10 days prior to the due date t_k. That is, $e_k = \max\{t_k - 10, 0\}$.

Figure 4 in the appendix show the CopyCat performance ratios for all three scenarios. The graphs of More Demands and Large Orders First show that the CopyCat ratios stabilize at a higher value than the graphs without availability date (by a factor of 0.16 at most). It also achieves higher maximum ratios, but much faster computations. The direct effect of including the availability dates is that the supplier is more restricted for his choice of possible order dates. Therefore, the search space is reduced, which brings the substantial reduction of the computation times (the computation time halved for the Conservative scenario). However, one can clearly see that the performance ratios became worse. Therefore, at the cost of the increasing performance ratios, one can significantly reduce the computation time for the CopyCat Algorithm by making use of extra information that the supplier can infer from customers (production availability date in this case).

10 Online JRP-CS

For all three scenarios, we used the following simulation parameters to simulate an instance of the online JRP-CS. We set the fixed joint setup cost $K_0 = 100$, $N = 3$ item types which are drawn at uniformly random, type-specific setup costs $K_1 = K_2 = K_3 = 20$, a per unit time holding cost $h = 1$ and a per unit rejection cost $r = 10$. The discrete planning horizon is a month in days: $[1, 30]$. Note that these parameters are equal to the parameters used in the paper by Elmachtoub and Levi, which means we expect to achieve nearly identical empirical competitive ratios for all three online algorithms. We performed the analysis for 100 independent runs, each with $M = 300$ simulated customer arrivals.

The outcomes from the experiments are contained in three figures and three tables. Figure 5, 6 and 7 in the appendix contain the average competitive ratios over 100 runs for all three online algorithms, for each scenario. Tables from 5 to 7 contain the maximum ratios, average final ratios and average computational times over all runs.

The Conservative graph suggests that the CopyCat and StablePair have an upper bound of 1.5, while the StablePair with scaling has much stronger peak at the ratio of 2. As with the online ELSP-CS, the CopyCat/StablePair ratios begin to increase from a certain number of requests (50 in this scenario) and stabilize at around stage 125. The scaling technique shows an extremely poor performance in the early stage of customer arrivals, but it quickly recovers to a substantially lower competitive ratios than the

CopyCat/StablePair. For the More Demands and Large Orders First scenarios, one can see the similar trends as with the online ELSP-CS. However, the StablePair with scaling shows a very high performance at later stages, which is nearly equal the optimal ratio of 1.

The maximum performance ratios in Table 5 show that the CopyCat and StablePair achieve similar upper bound ratios, which are below 1.65 for all three scenarios. It is much lower than the theoretical bound of 3 (for StablePair) and 4 (for CopyCat) as suggested by Elmachtoub and Levi (2016). However, their average computation times through all 100 runs were the key differentiators between the two algorithms. In Conservative scenario, the CopyCat required almost 8 times longer run time than the StablePair. This made the StablePair much more appealing in practice.

Comparing our results to the results of Elmachtoub and Levi, the peaks are placed at around the same height for all three scenarios and CopyCat/StablePair ratios begin to increase from 1 at around the same number of requests. However, one must observe that the Large Orders First graph show remarkably different outcomes. Notice that in our graph the peak for the StablePair lies at around 25 number of requests, which is to the right of the peak at the StablePair with scaling (which lies at 6 requests). However the graph from Elmachtoub and Levi have both peaks at around 6 requests and the StablePair behaves almost exactly like the StablePair with scaling with an upward shift. This difference may have occurred because Elmachtoub and Levi fixed the two large orders to be type 1 orders. On the contrary, we simulated their item types uniformly at random between 1 and N. Seeing from this perspective, our StablePair would accept more requests at early stages as the first two large orders may be of different type. Once the type-specific setup costs are incurred, it is good for the StablePair to accept the requests with the same item types as the setup costs are already sunk. This is not the case with the Elmachtoub and Levi's StablePair, likely to be causing the difference. The maximum ratios from our experiments are significantly lower than those of Elmachtoub and Levi (with the largest difference of 0.17 in Large Orders First Scenario). Again, such a big gap may have been due to the difference in the simulation parameter.

10.1 Online JRP-CS with production availability date

We can also associate each request with an individual availability date. With $N = 3$, we simulate the availability dates of type $1, 2, 3$ items differently. For type 1 customer, the availability date is at most 10 days before her due date. For type 2 customer, the availability date is at most 6 days before her due date. Finally, for type 3 customer, the availability date is at most 4 days before her due date. This simulates when the different types of perishable products perish at different rates (e.g., type 1 products perish at the lowest rate, type 3 products perish at the fastest rate).

Figure 8 in the appendix shows the CopyCat performance ratios for all three scenarios. The ratios are higher in average when compared to Figure 5-7. What is noticeable in this figure is that unlike in Figure 7, the average ratio for the Large Orders First scenario significantly became higher and intersects the graph of More Demands scenario. Hence, considering the availability date severely affects the CopyCat performance when the supplier receives large size orders. In general, when the supplier deals with multiple item types, the cost of reduced number of order dates (from to the availability date) is much more severe than in the single-type ELSP-CS. Seeing from Table 8, this variant of the online JRP-CS achieves much faster computations than in Table 5-7. However, even though the computation times nearly halved, the maximum ratio and the final ratio increased by a large fraction for Conservative and Large Orders First scenarios (largest increase of 0.20 for max. ratio). Hence, one must be more careful with including the availability date for the multi-item inventory control problems.

11 Online CJRP-CS

For the online CJRP-CS, the simulation parameters were set as follows. The joint order set up cost was set as a strictly concave function of the number of item types that are included in an order. This concave function was set to be a ceiling of a natural logarithm: $K_{0,s}(|S|) = \lceil \log(|S| + 1) * 100 \rceil$, with $|S|$ being the number of item types included in an order placed at time s. For example, if an order includes only 1 item type, then it is equal to 70. If an order includes two item types, it is equal to 110, and 139 for three item types and so on. We expanded on the number of item types in this simulation to $N = 5$. The individual set up costs were equal to 20 for every type of item and the rest of the parameters such as rejection costs and holding costs were set as $r = 10$ per unit, $h = 1$ per time unit. Also, due to the extremely long

computation times, we simulated the instance for thee total number of customers equal to 200 and the total number of repetitions equal to 50. Figures 9 - 11 show the plots for all three scenarios, and Table 9-11 show the relevant statistics.

Seeing from Figure 9-11, the results may appear to be the same as with the online JRP-CS in Figure 5-7. Indeed, since the CJRP-CS is a generalization of the JRP-CS, it is not bizarre to find similarities between Figure 9-11 and Figure 5-7. However, one can observe that for the Conservative scenario, the peaks for all three graphs are located at later stages than in Figure 5. This is due to the submodularity of the setup cost function as the optimal offline solution and the online solutions attempt to capture as many item types as possible in their orders - making them to reject customers at early stages and wait until a sufficient number of customers with various item types are observed.

From Table 9, the maximum performance ratios are safely below 1.6 for all scenarios, for CopyCat and StablePair. This is in line with the finding by Elmachtoub and Levi, where they prove that the CopyCat and StablePair attain the competitive ratios of 2 for submodular problems. Interestingly enough, the merge of submodularity to the joint replenishment problem does not appear to increase the competitive ratios in multiplication of their respective competitive ratios (e.g., CopyCat is 2-competitive for the submodular cost problem and 4-competitive for the online JRP-CS; but the submodular JRP-CS is not 2×4-competitive)

Another striking observation was that solving the online CJRP-CS with simulated 200 customer arrivals with 50 repetitions for the Conservative Scenario took over 7 hours to get the complete results from all three algorithms. Therefore, this shows that even a simple tweak to the classical logistics models can substantially complicate the problem and make it impossible to solve exactly. Furthermore, it suggests that the CopyCat Algorithm is not as flexible as with the StablePair Algorithm for general class of inventory control problems.

12 Online PCTSP

For the PCTSP, the total of 50 customers were simulated over 100 independent runs. The customer locations were simulated uniformly at random on a discrete metric space with symmetric vertical range $[-30, 30]$ and horizontal range $[-30, 30]$. For the purpose of our simulation, the rejection cost was set equal to 10 and the variable cost was set to be equal to the Euclidean distance between two locations.

It turns out that solving an offline PCTSP requires nearly 12 times more computation times than solving an ELSP with the similar settings. Indeed, it is because the classical TSP is NP-hard but the classical ELSP can be solved efficiently. More specifically, one should note that the final solution of the PCTSP also depends on the ordering of customer satisfaction. Unlike the inventory control problems where serving a customer does not require an ordering of all accepted customers, the PCTSP with online customer selection requires a strict ordering among the customers, which leads to a significant expansion in search space and an expensive reoptimization. Another remark regarding the simulation time is that the per-iteration computation time was highly volatile and heavily dependent on the simulated locations; in the Conservative scenario, the maximum run time among the 100 runs was 1433.29 seconds, while the minimum run time was 73.13 seconds.

In Figure 12 - 14, we plot the competitive ratios for both CopyCat and the J-L Algorithm introduced in Section 8.2. Note that the J-L approach makes an active use of the prior knowledge on the total number of requests M. This may have brought the following substantial differences between the two online algorithms. In the Conservative scenario, the competitive ratio for the CopyCat Algorithm appears to reach to 1.5, which is similar to that of comparable inventory control problems. However, one can notice that the CopyCat ratio departs from 1 at an early stage. This is possibly due to the absence of (fixed) setup cost, which encourages the supplier to accept customer more generously. For two remaining scenarios the similar trends hold.

The CopyCat algorithm attains a comparable empirical competitive ratio to the J-L Algorithm when the total number of customers is small. Especially in the Conservative Scenario, the CopyCat significantly outperforms the J-L Algorithm because the J-L Algorithm start to accept customers at early stages while the optimal offline solution rejects them. As the number of requests becomes larger and larger, the J-L Algorithm becomes even more generous and accepts a new customer more easily (due to the factor $\sqrt{\log M}$ in the acceptance condition). In consequence, the optimal offline solution reaches at a point where the most of the customers at U_k are accepted (that is, it reverts its previously made accept/reject

decisions) and become closer to the J-L solution. On the contrary, the CopyCat regrets its decisions at later stages, causing the large values for the competitive ratios.

Part IV
Conclusion

In this thesis, we studied and extended the results of the original paper written by Elmachtoub and Levi (2016). Specifically, we evaluate the empirical performance of their two novel online algorithms, CopyCat and StablePair, for a variety of logistics models with online customer selection. Our analysis confirms that for the online ELSP-CS and the online JRP-CS, the empirical competitive ratios for the CopyCat and StablePair are very close to the benchmark values. In all scenarios, their worst (maximum) competitive ratios are much below the theoretical guarantees, making them much more appealing in practice. We also observe that when the number of customers gets large, the StablePair, with and without scaling technique, begins to outperform the CopyCat both in performance ratios and in computation, which follows the computational results from Elmachtoub and Levi. Furthermore, we continue with our analysis by considering three different types of logistics problems with online customer selection: (i) online ELSP-CS and JRP-CS with production availability dates, (ii) online Cardinality JRP-CS, and (iii) online PCTSP. Introducing the production availability dates to the online ELSP-CS and JRP-CS nearly halved the CopyCat computation times, but significantly increased its performance ratio. On the contrary, the CopyCat for the online CJRP-CS took a tremendous amount of computation time, which was almost 8 times longer than the StablePair, clearly indicates that the quality of the CopyCat solution is highly vulnerable to the complexity of the model under consideration. It even suggests that the CopyCat is inappropriate to solve a general class of complex optimization models. Finally, we study the online version of the PCTSP and compare the performance of the CopyCat to the asymptotically best possible J-L online algorithm. For a large number of customers, the J-L clearly outperforms the CopyCat but it shows a comparable weakness when the number of customers is small.

Now, we outline two interesting points that are worth a discussion. First point concerns the relative performance of the CopyCat Algorithm to the J-L Algorithm for the online PCTSP. Although the J-L Algorithm showed strictly better performance ratios for a large number of customers, we do not think that the CopyCat is strictly worse than the J-L Algorithm. One should keep in mind that the J-L Algorithm requires the server to know the total number of customers exactly prior to his departure, which is often difficult to achieve in real life settings due to the stochastic demand fluctuations and the customer's option to cancel the delivery. Also, for More Demands and Large Orders First scenarios, the gap between the graphs of CopyCat and J-L is not large in absolute term (Figure 13, Figure 14). It is absolutely large for the Conservative scenario (Figure 12), but in that scenario the CopyCat strictly outperforms the J-L Algorithm for the first 25 stages. Hence, we find it difficult to say that the J-L Algorithm is always preferred to the CopyCat. Another point concerns the asymptotic behavior of CopyCat for the online PCTSP. Since our simulation ends at stage 50, we cannot tell if the CopyCat will always remain worse than the J-L for later stages. There is always a possibility that the CopyCat graph converges to 1 at a faster rate for the large number of customers. Hence, it is worth considering the asymptotic ratios for the CopyCat by implementing an efficient approximation algorithm for the offline PCTSP.

From a theoretical viewpoint, it is interesting to further consider the online PCTSP with demand time window, where every customer needs to be visited within a reasonable time interval. It has many real life applications since customers have limited patience unlike in our experiments. Another interesting topic concerns the online JRP-CS with online production, whereby the production step overlaps with the selection step. Elmachtoub and Levi (2016) already showed that the online ELSP-CS with online production is intractable, but including the online production option in inventory control problems with online customer selection is useful in many real world applications.

Appendices

A Algorithm

Algorithm 3 Online JRP-CS StablePair Selection Algorithm

Input: Order costs (K_0, K_i, h), rejection cost (r), a set of customers observable at stage k (U_k) and demands (d_k)
Output: Binary decision variable z_k that equals 1 if customer k is accepted and 0 otherwise.

1: Initialize: $z_k = 0$
2: **Outer Loop:**
3: **for** Every due date t for each customer in U_k **do**
4: Set $Q = \{t\}$
5: **for** Every $i \in \{1, ..., N\}$ **do**
6: Define set $D_i = \{j \in U_k : t_j \in [t, t + r/h] \text{ and } i_j = i\}$
7: **end for**
8: Define set $X = \{i : R(D_i) \geq K_i + \sum_{j \in D_i} h(t_j - t)d_j, i \in \{1, ..., N\}\}$
9: Define set $D = \bigcup_{i \in X} D_i$.
10: **if** $k \notin D$ **then**
11: Continue **Outer Loop** with the next due date
12: **end if**
13: Define condition B: $R(D) \geq K_0 + \sum_{i \in X} (K_i + \sum_{j \in D_i} h(t_j - t)d_j)$
14: **if** Condition B is true **then**
15: $z_k = 1$
16: Exit **Outer Loop**
17: **end if**
18: **end for**
19: Output z_k

Line 4 sets the production option Q to contain a single due date t. Then the loop from line 5 to 7 defines a set of type i customers D_i whose due dates are in the setup interval of the type i order at Q, for every type of item. Line 8 defines a set of item types X whose corresponding set D_i has lower or equal production costs for accepting all customers in the set than their rejection costs. Line 9 unites all D_i for the item types in X, and form a customer set for the potential stable pair. The loop from line 10 to 12 continues to the next due date if customer k is not in the set D. Finally, the same process is repeated for every customer in U_k until the accepting condition is satisfied. If satisfied, the algorithm outputs the binary variable z_k that is equal to 1 and completes the algorithm.

B Proofs

Lemma 1
Assume that $P(\cdot)$ is nondecreasing. Then the total rejection costs of the CopyCat Algorithm at each stage k is at most the respective optimal offline cost, i.e., $R(\Omega_k^{\text{CC}}) \leq R(\Omega_k^*)+P(A_k^*) = C^*(U_k)$ for all k. Specifically, the final rejection cost $R(\Omega_k^{\text{CC}}) \leq C^*(U_k)$.

PROOF:
Proof by induction. Start with the base case $k = 1$, where $R(\Omega_1^{\text{CC}}) = R(\Omega_1^*) \leq R(\Omega_1^*) + P(A_1^*) = C^*(U_1)$. The first equality is from $\Omega_1^{\text{CC}} \leq \Omega_1^*$ since the selection decision made by $\phi^*(U_1)$ is copied. The inequality follows from the nonnegativity of $P(\cdot)$. The last equality follows directly from the definition of C^*. Now assume the inductive hypothesis that $\Omega_{k-1}^* \leq C^*(U_{k-1})$. Consider the following two cases depending on whether customer k was accepted or rejected in the solution of $\phi^*(U_k)$.
Case (1). If customer k is accepted in the solution of $\phi^*(U_k)$ (i.e., $k \in A_k^*$), then $R(\Omega_k^{\text{CC}}) = R(\Omega_{k-1}^*) \leq C^*(U_{k-1}) \leq C^*(U_k)$. The first equality holds because $k \in A^*$, which implies that Copycat accepted k, and thus $\Omega_k^{\text{CC}} = \Omega_{k-1}^{\text{CC}}$. The inequality follows from the inductive hypothesis. The last inequality follows from the fact that any solution to $\phi^*(U_k)$ induces a solution to $\phi^*(U_{k-1})$ with cost at least $C^*(U_{k-1})$.
Case (2). If customer k is rejected in the solution of $\phi^*(U_k)$ (i.e., $k \in \Omega_k^*$), then $R(\Omega_k^{\text{CC}}) = R(\Omega_{k-1}^*) + r_k \leq C^*(U_{k-1}) + r_k = C^*(U_k)$. The first equality holds since $k \in \Omega_k^*$, which implies that Copycat also rejected k. The inequality follows from the inductive hypothesis. The last equality holds by the linearity of $R(\cdot)$ and the fact that there is an optimal solution to $\phi^*(U_k)$ that rejected customer k.

Lemma 2
Assume that $P(\cdot)$ is nondecreasing. Then the total rejection costs of the StablePair Algorithm is $R(\Omega^{\text{SP}}) \leq R(\Omega^* \cap \Omega^{\text{SP}}) + P(\Omega^{\text{SP}} \cap A^*) \leq R(\Omega^* \cap \Omega^{\text{SP}}) + P(A^*)$ for any offline optimal solution (A^*, Ω^*).

PROOF:
We first show that for all $X \subseteq \Omega^{\text{SP}}$, $R(X) \leq P(X)$. Let $X \subseteq \Omega^{\text{SP}}$ and assume for contradiction $R(X) > P(\mathbb{Q}, X) = P(X)$. **(1)**
Let k be the last arriving customer in X. Since k was rejected by StablePair, it follows that $(\mathbb{Q}, X)$ is not a stable pair for customer k. Thus, by the equation of the stable pair there exists a set $S \subseteq X$ such that
$R(S) < P(Q, X) - P(Q, X \setminus S) = P(X) - P(X \setminus S)$. **(2)**
Note that $S \neq \emptyset$ or else $0 < 0$. Subtracting (2) from (1) yields
$R(X \setminus S) > P(X \setminus S)$. **(3)**
Now reset $X \leftarrow X \setminus S$ and repeat the analysis above until $X = \emptyset$. This will eventually give a contradiction that $0 > 0$. Now let (A^*, Ω^*) be any optimal offline solution and let $X = \Omega^{\text{SP}} \cap A^*$. From the previous argument, it follows that
$R(\Omega^{\text{SP}} \cap A^*) \leq P(\Omega^{\text{SP}} \cap A^*)$. **(4)**
Adding $R(\Omega^{\text{SP}} \cap \Omega^*)$ to both sides of (4) completes the proof.

Lemma 3
The accepted set of customers by StablePair is at least that of CopyCat, i.e., $A_k^{\text{CC}} \subseteq A_k^{\text{SP}}$.

PROOF:
Consider a customer $k \in A^{\text{CC}}$ and let A_k^* be the optimal solution induced by $\phi^*(Q, U_k)$. Since k was accepted by the Copycat Algorithm, then $k \in A_k^*$. This implies that (Q, A_k^*) is a stable pair for k since there is a solution to $\phi^*(Q, U_k)$ that accepts all of A_k^*. Therefore $k \in A^{\text{SP}}$ and $A^{\text{CC}} \subseteq A^{\text{SP}}$.

Lemma 4
Let $(Q, B) \subseteq (\mathbb{Q}, U)$ be a stable pair. Each setup interval induced by the ELS solution to $P(Q, B)$ must have a length of at most r/h periods.

PROOF:
Assume there is a setup interval $[a, b]$ in the solution to $P(Q, T)$ such that $b - a > r/h$. By the ZIO property, there must be an order at a. This means there exists a customer k with due date at time b that pays a per unit holding cost greater than r. This implies that the optimal solution of $\phi^*(Q, U_k)$ is better off rejecting k and keeping everything else the same, which is a contradiction to the stability of (Q, T)

Lemma 5
Let $(Q, B) \subseteq (\mathbb{Q}, U)$ be a stable pair and let $[a, b]$ be a setup interval from the ELS solution for $P(Q, B)$. Then the interval $[a, b]$ must contain the due date of a customer in A^*. Therefore, $[a, b]$ must intersect a setup interval from the ELS solution for $P(A^*)$.

PROOF:
Assume that $[a, b]$ does not contain the due date of any customers in A^*. By the stability of (Q, T), the rejection cost of the customers served by the order at a are greater than the production costs incurred in the interval $[a, b]$. This implies that A^* could be augmented to include all customers having due dates in $[a, b]$ without increasing the overall cost. Since we chose A^* to be maximal, then this is a contradiction.

C Tables and Figures

C.1 Online ELSP-CS

Table 1: ELSP-maximum performance ratio

	Maximum performance ratio		
Scenarios	CopyCat	StablePair	StablePair with scaling (2)
Conservative	1.492	1.476	1.984
More Demands	1.757	1.595	2.133
Large Orders First	1.329	1.304	1.365

Table 2: ELSP-final performance ratio

	Final performance ratio		
Scenarios	CopyCat	StablePair	StablePair with scaling (2)
Conservative	1.423	1.413	1.111
More Demands	1.280	1.225	1.038
Large Orders First	1.176	1.098	1.006

Table 3: ELSP-average computation time in seconds

selection & production	Average computation time (sec)		
Scenarios	CopyCat	StablePair	StablePair with scaling (2)
Conservative	8,349	2,293	2,064
More Demands	7,272	1,138	909
Large Orders First	6,281	785	706

C.1.1 Onlin ELSP-CS with production availability date

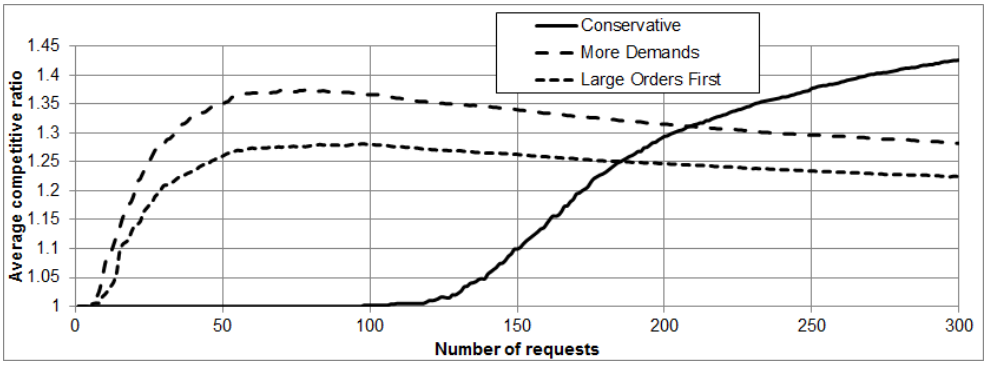

Figure 4: ELSP with production availability date - all scenarios

Table 4: ELSP with production availability dates - CopyCat

	CopyCat Algorithm		
Scenarios	Maximum ratio	Final ratio	Avg. computation time (sec)
Conservative	1.479	1.426	4,102
More Demands	1.504	1.282	4,697
Large Orders First	1.423	1.224	4,202

C.2 Online JRP-CS

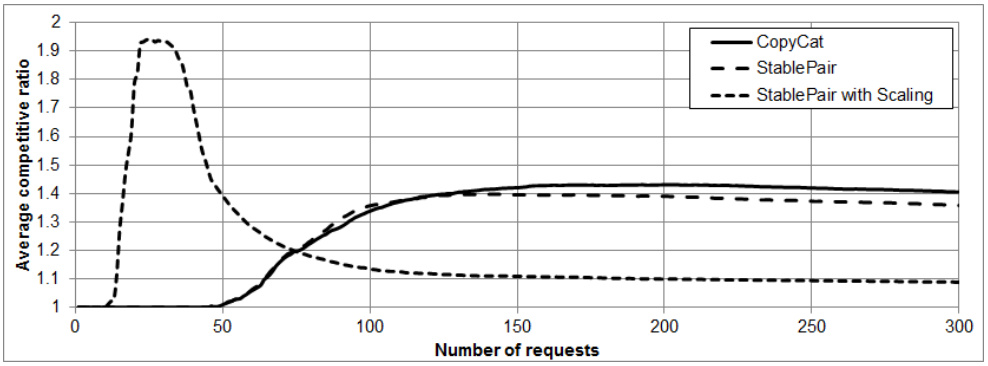

Figure 5: JRP - Conservative scenario

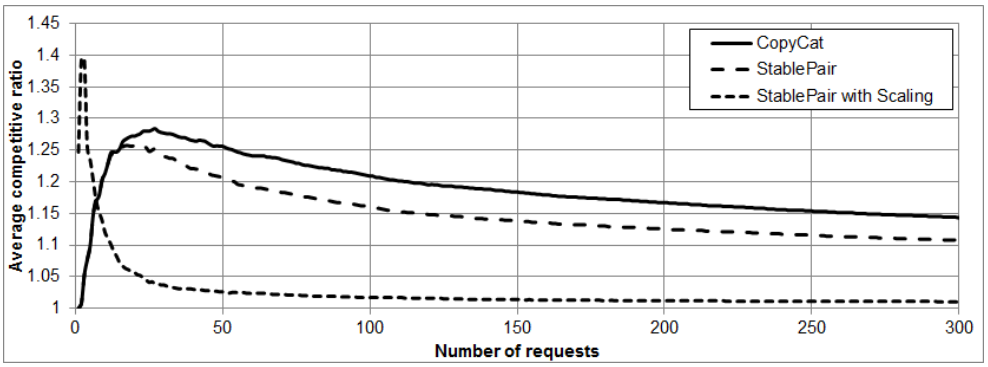

Figure 6: JRP - More Demands scenario

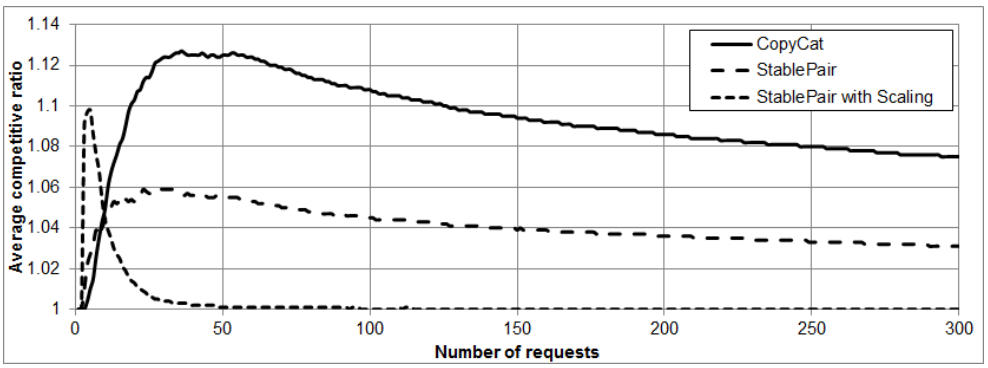

Figure 7: JRP - Large Orders First scenario

Table 5: JRP - maximum performance ratio

	Maximum performance ratio		
Scenarios	CopyCat	StablePair	StablePair with scaling (2)
Conservative	1.515	1.502	2.642
More Demands	1.611	1.611	2.950
Large Orders First	1.294	1.276	1.505

Table 6: JRP - final performance ratio

	Final performance ratio		
Scenarios	CopyCat	StablePair	StablePair with scaling (2)
Conservative	1.405	1.358	1.088
More Demands	1.143	1.108	1.021
Large Orders First	1.075	1.031	1.000

Table 7: JRP - average computation time in seconds

selection & production	Average computation time (sec)		
Scenarios	CopyCat	StablePair	StablePair with scaling (2)
Conservative	16,846	2,106	1,895
More Demands	7,461	932	914
Large Orders First	10,541	1,317	1,265

C.2.1 Onlin JRP-CS with production availability date

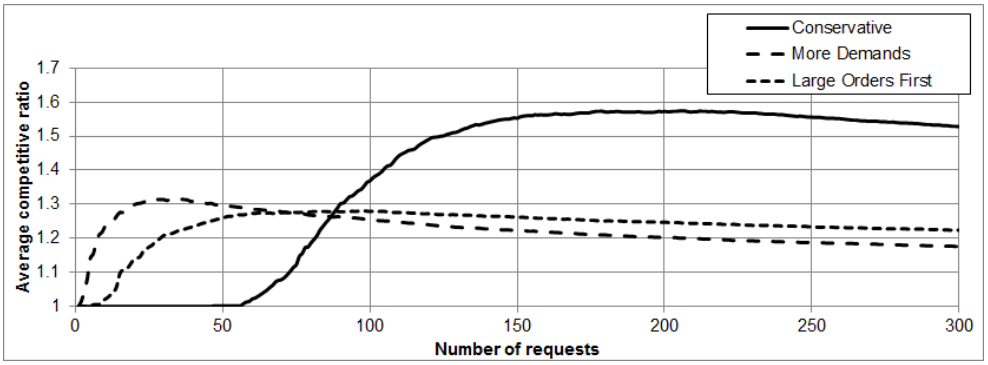

Figure 8: JRP with production availability date - all scenarios

Table 8: JRP with production availability dates - CopyCat Algorithm

	CopyCat Algorithm		
Scenarios	Maximum ratio	Final ratio	Avg. computation time (sec)
Conservative	1.729	1.528	7,009
More Demands	1.657	1.176	6,906
Large Orders First	1.463	1.150	5,820

C.3 Online CJRP-CS

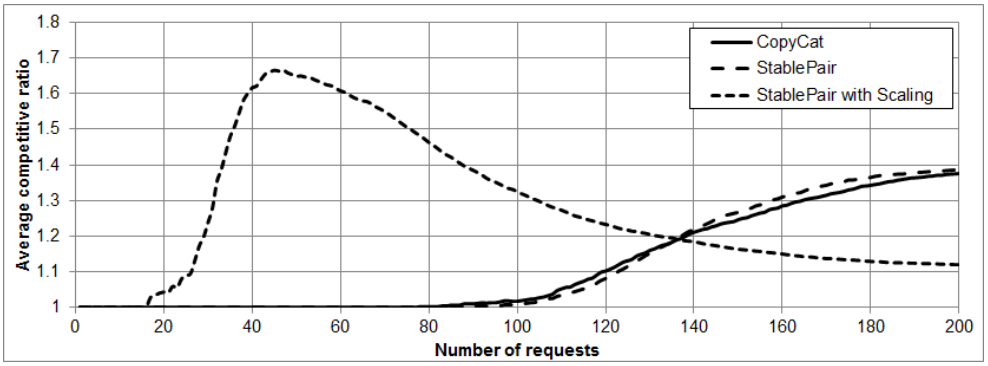

Figure 9: CJRP - Conservative scenario

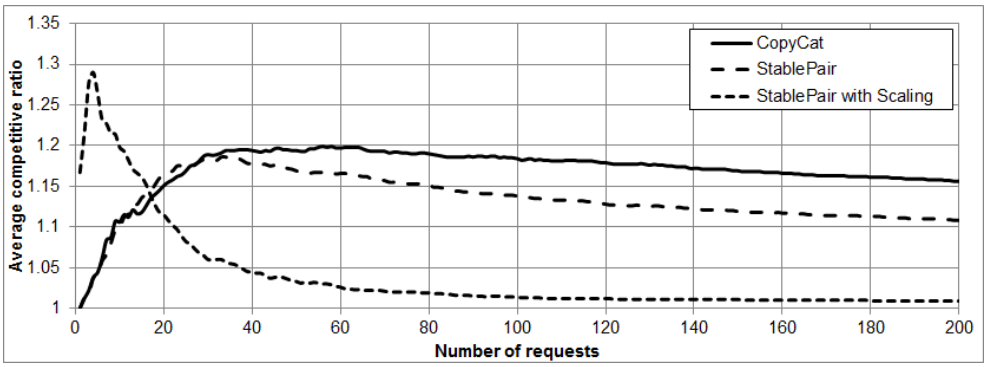

Figure 10: CJRP - More Demands scenario

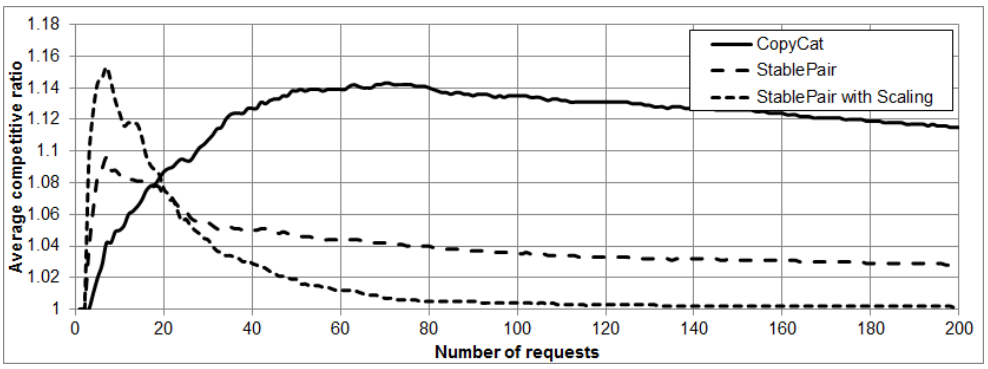

Figure 11: CJRP - Large Orders First scenario

Table 9: CJRP - maximum performance ratio

	Maximum performance ratio		
Scenarios	CopyCat	StablePair	StablePair with scaling (2)
Conservative	1.434	1.448	1.924
More Demands	1.579	1.579	2.000
Large Orders First	1.309	1.500	1.515

Table 10: CJRP - final performance ratio

	Final performance ratio		
Scenarios	CopyCat	StablePair	StablePair with scaling (2)
Conservative	1.376	1.388	1.120
More Demands	1.156	1.108	1.009
Large Orders First	1.115	1.028	1.001

Table 11: CJRP - average computation time in seconds

selection & production	Average computation time (sec)		
Scenarios	CopyCat	StablePair	StablePair with scaling (2)
Conservative	25,179	3,147	2,958
More Demands	20,191	2,523	2,297
Large Orders First	29,888	3,736	3,399

C.4 Online PCTSP

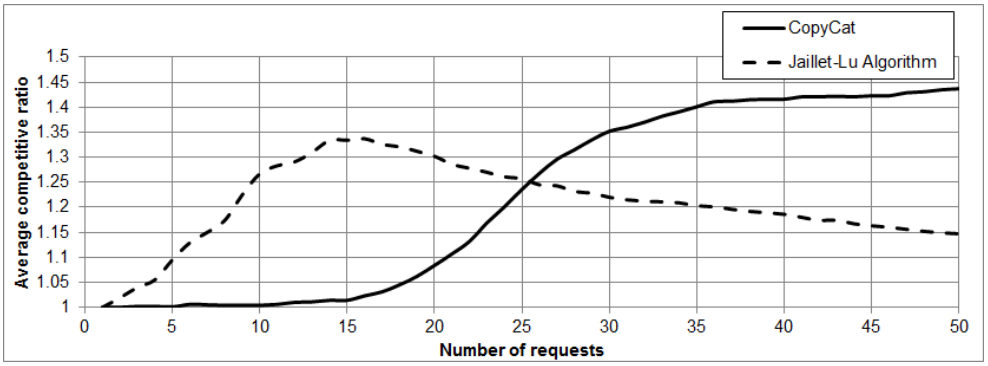

Figure 12: PCTSP - Conservative scenario

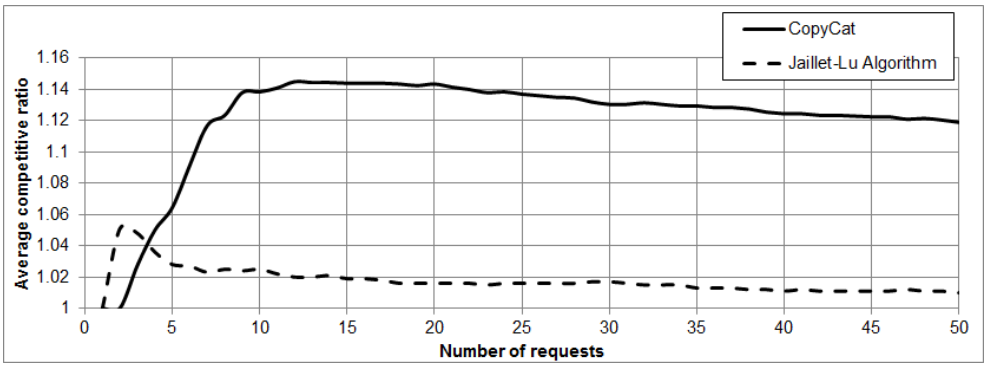

Figure 13: PCTSP - More Demands scenario

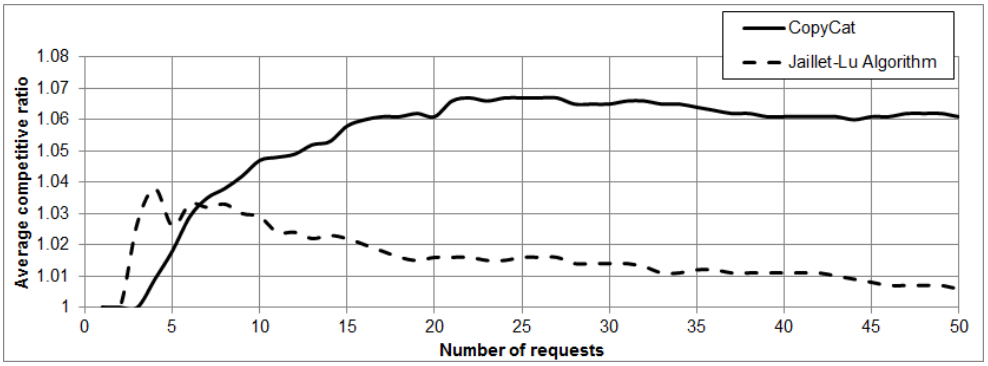

Figure 14: PCTSP - Large Orders scenario

Table 12: PCTSP - maximum performance ratio

	Maximum performance ratio	
Scenarios	CopyCat	JL
Conservative	1.654	2.313
More Demands	1.617	1.737
Large Orders First	1.506	1.510

Table 13: PCTSP - final performance ratio

	Final performance ratio	
Scenarios	CopyCat	JL
Conservative	1.473	1.145
More Demands	1.134	1.010
Large Orders First	1.061	1.007

Table 14: PCTSP - average computation times in seconds

selection & production	Average computation times (sec)	
Scenarios	CopyCat	JL
Conservative	11,909	5,760
More Demands	17,344	2,181
Large Orders First	18,595	2,833

References

[1] Arkin E, Joneja D, Roundy R, *"Computational complexity of uncapacitated multi-echelon production planning problems"* Oper. Res. Lett. 8(2): 61-66, 1989.

[2] BienStock D, Goesmans MX, Simchi-Levi D. Williamson D *"A note on the prize collecting traveling salesman problem."* Math. Programming 59(1):413-420, 1993.

[3] Cheung M, Elmachtoub AN, Levi R, Shmoys DB *"The submodular joint replenishment problem."* Math. Programming 158(1):207-233, 2015.

[4] Elmachtoub AN, Levi R *"Supply chain management with online customer selection."* Oper. Res. 64(2):458-473, 2016.

[5] Fotakis D *"On the competitive ratio for online facility location."* Algorithmica 50(1):1-57, 2008.

[6] Gavish, B. and S.C. Graves *"The travelling salesman problem and related problems."* Operations Research Center, MIT, Cambridge, MA, 1978.

[7] Geunes. J, Levi. R, Edwin Romeijn. H., Shmoys. D. B., *"Approximation algorithms for supply chain planning and logistics problems with market choice."* Math. Program., Ser.A 130: 85-106, 2011.

[8] Jaillet P, Lu X *"Online traveling salesman problems with rejection options."* Networks 64(2):84-85, 2014.

[9] N. Absi, S. Kedad-Sidhoum, S. Dauzère-Pérès, *"Uncapacitated lot-sizing problem with production time windows, early production, backlog and lost sale."* International Journal of Production Research 49(9): 2551-2566, 2011.

[10] Van den Heuvel W, Kundakcioglu OE, Geunes J, Romeijn HE, Sharkey TC, Wagelmans APM *"Integrated market selection and production planning: Complexity and solution approaches."* Math. Programming 134(2):395-424, 2012.

[11] Van den Heuvel W, Wagelmans APM *"Worst-case analysis for a general class of online lot-sizing heuristics."* Oper. Res. 58(1):59-67, 2010.

[12] V. T'kindt, Croce. F.D. *"A note on "Two-machine flow-shop scheduling with rejection" and its link with flow-shop scheduling and common due date assignment."* Computers and Oper. Res. 39(12):3244-3246, 2012.

[13] Wagner H.M., Whitin T.M., *"Dynamic version of the economic lot size model."* Management Sci. 5(1): 89-96, 1958.