Submitted to *Operations Research*
manuscript (Please, provide the mansucript number!)

Supply Chain Management with Online Customer Selection

Adam N. Elmachtoub
Department of Industrial Engineering and Operations Research, Columbia University, New York, NY, 10027, adam@ieor.columbia.edu

Retsef Levi
Sloan School of Management, Massachusetts Institute of Technology, Cambridge, MA, 02139, retsef@mit.edu

We consider new online variants of supply chain management models, where in addition to production decisions, one also has to actively decide on which customers to serve. Specifically, customers arrive sequentially during a selection phase, and one has to decide whether to accept or reject each customer upon arrival. If a customer is rejected, then a lost-sales cost is incurred. Once the selection decisions are all made, one has to satisfy all the accepted customers with minimum possible production cost. The goal is to minimize the total cost of lost sales and production. A key feature of the model is that customers arrive in an online manner, and the decision maker does not require any information about future arrivals.

We provide two novel algorithms for online customer selection problems which are based on repeatedly solving offline subproblems that ignore previously made decisions. For many important settings, our algorithms achieve small constant competitive ratio guarantees. That is, for any sequence of arriving customers, the cost incurred by the online algorithm is within a fixed constant factor of the cost incurred by the respective optimal solution that has full knowledge upfront on the sequence of arriving customers. Finally, we provide a computational study on the performance of these algorithms when applied to the economic lot sizing and joint replenishment problems with online customer selection.

Key words: supply chain management; online optimization; customer selection
History: Received April 2012; revisions received August 2013, March 2015;

1. Introduction

Supply chain management theory provides many streamlined optimization models where the goal is to satisfy an exogenous deterministic or stochastic sequence of customers over a specified planning horizon at minimum cost (for example, see Muckstadt and Sapra (2010)). More recent practice and research trends in supply chain management have led to broader models that consider decisions on the supply side as well as on the demand side. More specifically, the customers to which the supply chain should respond and commit to are not entirely exogenous parameters, but may be influenced by endogenous decisions, such as pricing, promotions, and other strategic marketing-based factors. In particular, a supplier should strive to optimally match demand to the supply chain's production capabilities. A fundamental aspect of this issue is the choice of customers to which the supplier commits to serving, and how these customers are implicitly, or explicitly, chosen. These decisions regarding which customers to serve often depend on the customer's associated revenue as well as the marginal costs expected to be incurred if the customer is satisfied.

We study a broad class of supply chain models that capture situations in which customers arrive to the supply chain sequentially, each with specific requirements and an associated revenue. Our core model has decisions that are made in two phases. First, there is a selection phase, where the decision maker has to decide in real-time which customers to accept and which ones to reject, without assuming any knowledge of future customer arrivals. We refer to this process as *online customer selection*. After the selection decisions are made, there is a production phase where the decision maker must serve all of the accepted customers with minimum production cost. Each of the rejected customers incurs a rejection cost that can be associated with either lost revenue, a fee paid to a third-party supplier, or the cost of satisfying the customer from a spot market. The goal is to minimize the total rejection cost plus the production cost of satisfying the accepted customers.

The online customer selection models studied in this work attempt to capture real-life operational situations of a make-to-order supplier or service provider. Typically, decisions are broken up into phases (weeks, quarters, years). In each phase, the supplier receives customer orders (requests) due in some time period in future phases, and needs to immediately (or by the end of the current phase) decide which customers to serve as their requests arrive. During the same phase, the supplier is also serving the customers that were accepted in previous phases with minimum production cost. In many of these settings, it is often extremely challenging to form a reliable demand distribution due to issues such as market volatility, lack of reliable data, and the fact that customers may have very different needs for customized products (or services). Thus, the assumption that the supplier has minimal knowledge about future customer arrivals could lead to more robust policies.

1.1. Contributions

The major contributions in this work are two-fold. First, we introduce a general class of models for online customer selection problems that captures several important settings. Similar to the concept of yield management, the decision maker aims to select an optimal set of customers to maximize profitability. In most yield management models, there is a finite amount of inventory (capacity) available that is used to generate as much revenue as possible, while the costs are assumed to be sunk. Unlike yield management models, in the setting studied in this paper there are no sunk costs nor finite capacities, but the cost structure has economies of scale. Therefore, as more customers are accepted in our setting, the conditions for accepting future customers become softer whereas the opposite effect occurs in yield management since inventory becomes more scarce as more customers are accepted. In the presence of economies of scale, determining the 'marginal' cost of a single customer is highly dependent on the other customers that are selected, and therefore selection decisions are also dependent on previously accepted customers.

Another notable difference compared to yield management is that the models we propose aim to minimize costs, which are composed of production and rejection costs. In applications where the rejection costs correspond to a spot market or third-party fees, the total incoming revenue is fixed and the objective is a true minimization of the total production and rejection costs. When the rejection costs correspond to lost sales costs, we still focus on minimizing the total production and rejection costs, rather than maximizing profit. Although the two objectives result in equivalent problems, finding an optimal solution to either is typically intractable. Moreover, finding a near-optimal solution to maximize profit, even in an offline setting, is a hopeless endeavor from a theoretical perspective (due to the mixed sign objective). On the contrary, we are able to find solutions that approximate the cost objective well with strong provable guarantees. In particular, we next describe algorithms to generate such solutions in an online manner.

The second set of contributions includes two novel *online* algorithms for making decisions in general online customer selection models. The performance of the proposed online algorithms is evaluated using the well-known *competitive ratio* framework, which is widely used in many optimization settings (see Albers (2003)). In particular, the online policies are compared to the *optimal offline solution* that can be obtained if one has upfront knowledge on the specific customer arrivals and their demands. Moreover, the assumption is that the customer arrivals can be generated by a worst-case adversary, who aims to maximize the ratio between the cost of the online and optimal offline solutions. Under realistic assumptions, the online policies obtain small constant competitive ratios for a broad array of supply chain models with online customer selection. That is, the cost of the online policies is guaranteed to be within a constant factor of the optimal offline policy for *any* sequence of customers and their demands. Furthermore, we provide compelling computational evidence that suggests the proposed policies typically behave significantly better than their theoretical guarantees, and are near-optimal even under conservative experiments.

The two newly proposed online algorithms, *Copycat* and *StablePair*, are based on a new application of repeated re-optimization. A very common heuristic frequently used to make decisions in practice is to re-optimize in every period based on the current state of the system and the decisions made so far. Empirically, these types of heuristics perform well in certain settings and poorly in others. In contrast to this approach, Copycat and StablePair re-optimize assuming that previously made decisions *can be changed*, and use the resulting outcome to guide the online decisions in the current period. (As we shall show, the resulting online policy is still feasible.) Specifically, these algorithms make decisions for the online customer selection problem based on solving a problem with offline customer selection defined with respect to all the customers that arrived thus far and assuming that no selection decisions have yet been made. The resulting offline solution is then used to make a decision regarding the selection (rejection) of the customer that just arrived. Although these re-optimization heuristics ignore previously made decisions, the accept/reject decisions that have been made cannot be reversed. Both algorithms have small constant competitive ratios and perform well in computational experiments for a variety of online customer selection problems.

To demonstrate some of the main ideas of our work, we briefly discuss the *Economic Lot Sizing (ELS) problem with Online Customer Selection*. In the traditional ELS problem, one has to satisfy a pre-specified sequence of demands over a given discrete planning horizon with a sequence of production orders over that horizon. Each order incurs a fixed setup cost K, regardless of the quantity being produced. Each demand is then served from the latest order prior to its due date. Demands incurs a per unit, per period holding cost h if they are served before their due dates. This problem was first introduced and solved by dynamic programming in the seminal paper of Wagner and Whitin (1958). If the supplier can reject customers at a per unit cost r, then this becomes the ELS problem with offline customer selection that can also be solved efficiently using dynamic programming (Geunes et al. (2006)). However, we focus on the ELS problem with online customer selection, where there is a selection phase, in which customers arrive one at a time, each specifying a demand quantity and a due date for some time in a future phase. One must make a decision whether to accept or reject customers immediately upon arrival (or at some later point before the production phase begins) without any knowledge of the future arrivals. After all customers arrive, one has to satisfy all of the selected customers. This is reduced to solving the traditional ELS problem with respect to the selected customers and their demands.

The first online algorithm proposed in this paper is a re-optimization framework called the *Copycat Algorithm*. Upon each customer arrival, an ELS problem with offline customer selection defined only for customers that have already been observed thus far is solved. Then the current customer that just arrived is accepted if and only if it is accepted in the optimal offline solution that was computed. It can be shown that the Copycat Algorithm has cost no more than three times the optimal offline cost, and therefore has a competitive ratio of three. The analysis is shown to be tight by an appropriately designed bad example. In addition, we consider the extension where the customer selection and production phases can overlap, i.e., one has to make selection and production decisions simultaneously, but customers' due dates are monotone in their arrival time. We show that the competitive ratio of 3 is maintained.

Similar results for the Copycat Algorithm also apply to online customer selection variants of offline supply chain models, such as the multi-item *Joint Replenishment (JR)* problem (Levi et al. (2006)) and *Facility Location (FL)* problem (Shmoys et al. (1997)). Specifically, we provide a competitive ratio of four to both of these problems. However, solving JR and FL problems with offline customer selection is NP-hard (Arkin et al. (1989), Cornuéjols et al. (1990)). Motivated by this difficulty, we adapt the Copycat Algorithm to obtain a second algorithm called the *StablePair Algorithm* that can be implemented efficiently, runs in time polynomial in the input size, and no longer has to solve the full offline problem to make each selection decision. Moreover, the StablePair Algorithm achieves improved competitive ratios of three each for both the JR and FL problems with online customer selection. In fact, for FL the StablePair Algorithm with a cost scaling heuristic obtains a competitive ratio of 2.41.

The Copycat and StablePair Algorithms can also be applied effectively to an additional class of important problems. Consider any problem in which the production cost is nonnegative, nondecreasing, and *submodular* with respect to the set of accepted customers. These problems arise when there are economies of scale such as in network routing and continuous replenishment problems (see Deering and Cheriton (1990), Federgruen and Zheng (1992), and Lovász (1983)). For submodular problems with online customer selection, our algorithms have a competitive ratio of two. We note that the ELS, JR, and FL problems do not belong to the class of submodular production problems (Goemans and Skutella (2000)). Finally, we provide a universal lower bound of two on the competitive ratio of any deterministic online algorithm for the submodular, ELS, JR, and FL problems with online customer selection.

1.2. Literature Review

Supply chain models with customer selection have recently gained interest (for a broader literature on order acceptance problems, see the survey of Slotnick (2011)). Xu et al. (2011) considers a stochastic periodic-review inventory model with lead time and the possibility of sales rejection. Bhaskaran et al. (2010) considers an inventory model with convex cost structure and the ability to backlog or refuse demand. Charikar et al. (2001) provide an approximation algorithm for the facility location problem with offline customer selection. Some models considered offline *market* selection, where a market is a sequence or collection of demands requested by a customer over time (or in multiple locations) that must be either fully accepted or rejected. Van den Heuvel et al.

(2012) show that it is NP-hard to approximate the profit maximization variants of these models within any constant. This motivates the focus on cost minimization variants that we and others have studied. 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. Their framework gives constant factor approximations to the ELS, JR, and FL problems with offline market selection. In contrast, in this paper we consider markets that consist only of a single period (or location), but the market/customer selection decisions are made online with *no* information about the future. Earlier work on the offline prize-collecting traveling salesman problem and the prize-collecting Steiner tree problem studied by Bienstock et al. (1993) and Goemans and Williamson (1995), respectively, also falls into our customer selection framework.

There has also been a stream of literature on customer selection in the context of admission control problems. For example, Carr and Duenyas (2000) and Caldentey and Wein (2006) both consider a make-to-stock queue that is committed to serving long term contracts, but also has a spot market from which it needs to accept and reject orders. In Plambeck et al. (2001) and Gallien et al. (2004), there is a queue that admits customers based on their revenue and amount of capacity available for their order. The major difference between this literature and our work is that we are not inherently limited by a capacity, which is the major factor for rejecting customers in these models. Instead, in our work customers are linked together due to the economies of scale in production, and a customer may be rejected based on his own projected cost.

Only recently has the concept of online optimization been used to study models in operations management, although thus far only to problems without market selection. In Van den Heuvel and Wagelmans (2010), the competitive ratio of the online economic lot sizing problem is shown to have a lower bound of two, matching the best known guarantee achieved in Axsater (1982). More general single item models are considered in Wagner (2010). In Buchbinder et al. (2014), an online primal-dual algorithm is proposed for a make-to-order variant of the online joint replenishment problem which has a competitive ratio of three. Fotakis (2008) provides a lower and matching upper bound depending on the number of customers for the online facility location problem. In revenue management, Ball and Queyranne (2009) finds booking policies with small competitive ratios, and Golrezaei et al. (2014) finds online algorithms for dynamic assortment optimization. In Keskinocak et al. (2001), online algorithms are developed for scheduling problems with lead time quotations. Finally, Jaillet and Lu (2013) propose algorithms for the online traveling salesman problem with online customer selection. The ideas used in all these papers are very different than the ones used in this work.

2. General Model

The models studied in this paper involve decisions that are made in two phases, a selection phase and a production phase. Next we describe the details of the core model, and discuss different extensions and applications in Section 2.1. First, there is a *selection phase* in which customers arrive sequentially in an online manner. In *stage* k of the selection phase, customer k arrives with requirements I_k and rejection cost r_k. Requirements (information) I_k may include demand quantities, due dates, and locations needed by the customer. After customer k arrives, the supplier needs to decide whether to accept or reject customer k using only the information regarding the first k customers. If customer k is accepted, then he must be served according to his requirements during the production phase. If customer k is rejected, a cost of r_k is incurred, which may represent lost revenue, a price paid to a third-party supplier, or the cost of satisfying the customer from a spot market. The rejection cost r_k is typically related to the requirements I_k, i.e., proportional to the demand quantity. Since the cost of each customer is not necessarily separable due to the economies of scale in production, evaluating the marginal cost of a customer a priori is challenging and thus makes the selection problem nontrivial. The selection phase completes when the supplier stops observing new customers. At this point the customers that arrived have been partitioned into accepted and rejected sets denoted by $\mathcal{A}$ and $\mathcal{R}$, respectively.

The second phase is the *production phase*, where the accepted customers are served accordingly to meet their requirements. Let $\mathcal{Q}$ be the set of production options that are available to the supplier. The production options may represent potential order dates, locations of facilities, or just a single production setting. For a nonempty set of production options $Q \subseteq \mathcal{Q}$ and a set of customers T, $P(Q,T)$ denotes the minimum possible production cost to serve the customers in T using only the production options in Q. The function $P(Q,T)$ typically represents the cost of an optimal solution to a minimization problem. The production cost for the accepted set of customers $\mathcal{A}$ is then denoted by $P(\mathcal{Q},\mathcal{A})$, which implies that the supplier could potentially use all options available to serve the accepted customers. We make the natural assumptions that $P(\mathcal{Q},T)$ is nondecreasing in T and $P(\mathcal{Q},\emptyset)=0$. The overall goal is to minimize the total production costs of the accepted customers plus the rejection costs of the rejected customers. This two phase problem is generally referred to as an *online customer selection* problem. In Sections 4, 5, 6, and 7 we will describe specific applications of the model by specifying definitions for I_k, r_k, $\mathcal{Q}$, and $P(\cdot,\cdot)$.

We now outline convenient notation used throughout the paper. Let N be the number of customers that arrived (unknown a priori), U be the full set of customers $\{1,\dots,N\}$ and U_k be the first k customers $\{1,\dots,k\}$, implying that $U=U_N$. When referring to the final stage N, the subscript may be dropped for simplicity. The rejection cost of a subset of customers $T \subseteq U$ is defined as $R(T)$, i.e., $R(T) = \sum_{k \in T} r_k$. The notation $\mathcal{A}_k$ and $\mathcal{R}_k$ denote the customers that were accepted and rejected, respectively, by the online algorithm through the first k stages. Note that $\mathcal{A}_k \cup \mathcal{R}_k = U_k$, $\mathcal{A}_k \cap \mathcal{R}_k = \emptyset$, $\mathcal{A}_{k-1} \subseteq \mathcal{A}_k$, and $\mathcal{R}_{k-1} \subseteq \mathcal{R}_k$ for all k.

If all information $I_1, \ldots, I_N$ and $r_1, \ldots, r_N$ is known upfront, then the *offline customer selection problem* is defined as $\mathrm{OPT}(\mathcal{Q}, U) = \min_{A \subseteq U} P(\mathcal{Q}, A) + R(U \backslash A)$. Let $\mathcal{A}_k^*$ and $\mathcal{R}_k^*$ denote an optimal pair of accepted and rejected sets in the offline problem $\mathrm{OPT}(\mathcal{Q}, U_k)$. Note that $\mathcal{A}_k^* \cup \mathcal{R}_k^* = U_k$ and $\mathcal{A}_k^* \cap \mathcal{R}_k^* = \emptyset$ for all k, but monotonicity does not necessarily hold since an offline solution may change its selection decisions as the stages progress. If there are multiple optimal solutions, we will assume that $\mathcal{A}_k^*$ is a maximal one, which means it is not contained in another optimal set of accepted customers.

The optimal offline cost through stage k is denoted by $C^*(U_k)$, which can be expressed as $C^*(U_k) = P(\mathcal{Q}, \mathcal{A}_k^*) + R(\mathcal{R}_k^*)$. The value $C(U_k)$ denotes the total cost incurred by the online algorithm through stage k, i.e., $C(U_k) = P(\mathcal{Q}, \mathcal{A}_k) + R(\mathcal{R}_k)$. Using these definitions, it follows that for any online algorithm and any stage k, $C^*(U_k) \leq C(U_k)$. Finally, we will sometimes drop the production options input from P and OPT which implies that we are using the entire set $\mathcal{Q}$, i.e., $P(\mathcal{Q}, \cdot) = P(\cdot)$ and $\mathrm{OPT}(\mathcal{Q}, \cdot) = \mathrm{OPT}(\cdot)$.

The performance of an online algorithm is evaluated using the notion of *competitive ratio.* An algorithm has a competitive ratio of α and is called α-competitive if $C(U) \leq \alpha C^*(U)$ for any online sequence of customers U and their respective characteristics. In other words, the cost of the algorithm is guaranteed to be at most α times the cost of an optimal offline solution for *any* customer sequence.

2.1. Model Applications

For clarity purposes, the core model described above only has two phases. However, the model and results can be easily extended to the scenario where there are multiple consecutive phases of selection and production, as long as the customer orders corresponding to a certain production phase arrive before that phase begins. For example, if each month is considered to be a distinct production phase, the corresponding selection phase is any time before that month begins. If customers request due dates that are at least a month away, which is very common in many applications, then when every month begins the exact demand is known and a minimum cost production plan can be created for that month. In addition, if we can defer selection decisions or if customers arrive in batches, then this actually provides greater flexibility and information for the decision maker. The algorithms we propose in Section 3 can be naturally adapted to these settings.

Specifically, the core model described above captures various settings, in which (i) customer order selection is a common practice or suppliers have the flexibility to satisfy customer orders from either internal resources or spot markets/third parties; (ii) customer 'patience' is short (not necessarily immediate) relative to the typical requested/acceptable lead time to satisfy customer requests; and (iii) the underlying supply/production cost structure is characterized by economies of scale that make the selection decisions for different customer orders highly dependent. Under this type of cost structure, production is typically planned well in advance.

The steel, glass, and construction industries are examples that fit the characteristics described above (Hintsches et al. (2010)). For example, consider a supplier that sells construction materials. When customers, i.e., contractors, request products with specific due dates, the supplier may decide that the quantity ordered does not cover the costs of materials and delivery. Note that this decision might depend on the requests of previous customers since their orders might have already covered some fixed costs. Since the contractors have deadlines to complete their projects, an instantaneous, i.e., 'online', decision is typically required to allow them to plan accordingly. Typically the supplier will communicate to the customer that they are out of stock as a way to reject the customer if necessary. The production cost functions for this type of application can be modeled by submodular or inventory control problems which we discuss in Sections 4, 5, and 6. In particular, in Section 5.2, we discuss scenarios where the supplier can only serve customer demands in multiples of a fixed quantity, a situation that arises frequently due to truckload capacities and/or batch processes.

The model also captures typical scenarios in the service industry (Zeithaml et al. (2001)). For example, consider a service provider that serves customers through a physical infrastructure. Typically, customers will request service from the provider and the provider will need to decide whether to serve each customer or not. In order to serve a customer, the proper infrastructure or facility must be setup near the customer. If the infrastructure needs to be improved or expanded at a relatively large cost for the customer to receive service, then he might be rejected by the provider ("out of network"). Otherwise, the customer is accepted and incurs a cost corresponding to his usage of the service. The production cost function in this setting could be modeled with the facility location problem or network design problem with submodular cost structure in Sections 4 and 7.

Other scenarios that can be captured by our models are when the 'rejection' cost represents a fee paid to a third-party supplier/logistics company or the use of spot markets (versus internal capacity) to satisfy the customer order. This situation can arise when the cost of serving customers using external resources is sometimes cost effective. For example, a local company may prefer to do some bulk shipping of their own but may rely on national companies to ship smaller or more long-distance packages. The use of third party logistics (3PL) is becoming increasingly commonplace and operationally advantageous (Lieb and Bentz (2005)).

The model can also be used as a tool to enhance *available-to-promise* strategies, in which companies interact with customers regarding their desired orders and delivery dates. Our model can be used as a support tool for negotiating with customers. For example, rather than rejecting a customer outright, a supplier may ask the customer to modify his requirements in order to create a mutually beneficial deal that will be accepted by the algorithm. This may include increasing the demand quantity, being more flexible with the due date, and/or reducing the variety of the order. Specifically, one could increase the demand quantity until the customer selection algorithm accepts the order in order to find a new deal to counteroffer to the customer. Alternatively, given the demand information of the customer, one can find all the due dates for which the customer selection algorithm would lead to accepting that order, and then present these options to the customer.

In other important settings, where the lead time of customers is short relative to the production phase, the selection phase and the corresponding production phase could overlap. The online customer selection model can easily be applied to these situations, but one would need to use, in addition to the online selection algorithm, an online production algorithm to solve the production problem since not all the accepted customers are known in advance. Although these problems are important, they are also much more complex. In fact, for many of the models discussed in this paper, just the online production problem with no customer selection is hard in the sense that there is no online algorithm with a constant competitive ratio. In contrast, in Section 5.3, we discuss an interesting special case where such an algorithm is available, and show how it could be incorporated into our selection algorithms to obtain a fully online selection and production algorithm.

3. Algorithms

3.1. Copycat Algorithm

We now provide a framework to solve problems with online customer selection which we call the *Copycat Algorithm*. Simply put, for every customer arrival k, the offline problem $\mathrm{OPT}(U_k)$ is solved to obtain an optimal offline solution $\mathcal{A}_k^*$. Then the arriving customer k is accepted if and only if customer k is accepted in the respective optimal solution (i.e., $k \in \mathcal{A}_k^*$). Otherwise, $k \in \mathcal{R}_k^*$ and customer k is rejected. This is called the Copycat Algorithm because it simply copies the optimal offline solution's decision at each customer arrival.

Copycat Algorithm: Accept current customer k if and only if $k \in \mathcal{A}_k^*$.

Although Copycat appears naive, it can be shown that it performs well for the class of problems considered in this paper. Note that fixing our previously made decisions when we re-optimize will result in most customers getting rejected and an overall poor performance. See Example 3 in the Online Appendix for a detailed example. We next show a surprising property of this algorithm that only requires the monotonicity of $P(\cdot)$. Specifically, the next lemma asserts that the rejection cost for Copycat will never be too large for *any* problem with online customer selection.

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., $R(\mathcal{R}_k) \leq R(\mathcal{R}_k^*) + P(\mathcal{A}_k^*) = C^*(U_k)$ for all k. Specifically, the final rejection cost $R(\mathcal{R}) \leq C^*(U)$.*

Proof. The proof is by induction. Start with the base case $k = 1$, where

$$R(\mathcal{R}_1) = R(\mathcal{R}_1^*) \leq R(\mathcal{R}_1^*) + P(\mathcal{A}_1^*) = C^*(U_1).$$

The first equality follows from the fact that $\mathcal{R}_1 = \mathcal{R}_1^*$ since the selection decision made by $\mathrm{OPT}(U_1)$ is copied. The inequality follows from the nonnegativity of $P(\cdot)$. The last equality follows directly from the definition of $C^*(\cdot)$.

Now assume the inductive hypothesis that $R(\mathcal{R}_{k-1}) \leq C^*(U_{k-1})$. Consider the following two cases depending on whether customer k was accepted or rejected in the solution of $\mathrm{OPT}(U_k)$.

Case 1) If customer k is accepted in the solution of $\mathrm{OPT}(U_k)$ (i.e., $k \in \mathcal{A}_k^*$), then

$$R(\mathcal{R}_k) = R(\mathcal{R}_{k-1}) \leq C^*(U_{k-1}) \leq C^*(U_k).$$

The first equality holds because $k \in \mathcal{A}_k^*$ which implies that Copycat accepted k, and thus $\mathcal{R}_k = \mathcal{R}_{k-1}$. The inequality follows from the inductive hypothesis. The last inequality follows from the fact that any solution to $\mathrm{OPT}(U_k)$ induces a solution to $\mathrm{OPT}(U_{k-1})$ with cost at least $C^*(U_{k-1})$.

Case 2) If customer k is rejected in the solution of $\mathrm{OPT}(U_k)$ (i.e., $k \in \mathcal{R}_k^*$), then

$$R(\mathcal{R}_k) = R(\mathcal{R}_{k-1}) + r_k \leq C^*(U_{k-1}) + r_k = C^*(U_k).$$

The first equality holds since $k \in \mathcal{R}_k^*$ 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 $\mathrm{OPT}(U_k)$ that rejected customer k. $\square$

Using Lemma 1, it is clear that if the production costs of Copycat are no more than β times the optimal offline cost, a competitive ratio of $\beta + 1$ is obtained. This is stated precisely in the following theorem.

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

Interestingly, we can also show that the Copycat Algorithm works well if the sequence of optimal solutions satisfies a certain property. Specifically, if satisfying $\cup_{i=1}^{k}\mathcal{A}_i^*$ has cost at most $\beta C^*(U_k)$, then the same holds for satisfying $\mathcal{A}$ since $\mathcal{A} \subseteq \cup_{i=1}^{k}\mathcal{A}_i^*$ by definition of the Copycat Algorithm. Combining this fact with Lemma 1, we obtain the following lemma.

LEMMA 2. *If $P(\cup_{i=1}^{k}\mathcal{A}_i^*) \leq \beta C^*(U_k)$, then the Copycat Algorithm is $(\beta+1)$-competitive.*

In the subsequent Sections 4-7, we shall show how to obtain bounds like the one in Theorem 1 above for several interesting problems and cases. However, one potential major flaw with the Copycat Algorithm is that it requires an exact solution to the offline problem in each stage. Even worse, the offline problem may sometimes be NP-hard such as in the JR and FL applications described in Sections 6 and 7, respectively. Motivated by these issues, we present another algorithm in the next subsection that is efficiently computable for all the applications we consider.

3.2. StablePair Algorithm

We now provide another algorithm for online customer selection called the *StablePair Algorithm.* For a nonempty subset of production options $Q \subseteq \mathcal{Q}$ and customers $T \subseteq U$, we call (Q,T) a *stable pair* if there exists an optimal solution to the respective offline customer selection problem defined on Q and T, denoted by $\text{OPT}(Q,T)$, that accepts all of the customers in T. The StablePair Algorithm accepts a given customer k if and only if there exists a stable pair $(Q,T) \subseteq (\mathcal{Q},U_k)$, such that $k \in T$.

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

As we shall show, the StablePair Algorithm has two main benefits. First, a stronger bound on the rejection costs can be obtained, and second, the selection phase can be implemented in polynomial time for many interesting problems that we consider. The stability name arises from the fact that if a customer was accepted by StablePair, then he would also be accepted if he had arrived at a later date. (This is not necessarily true for the Copycat Algorithm.) Moreover, in Lemma 3 below (proof in Online Appendix), it is shown that the StablePair Algorithm is less conservative than the Copycat Algorithm in that each customer accepted by Copycat is also accepted by StablePair. (We let the superscripts C and S refer to the Copycat and StablePair Algorithms decisions, respectively.)

LEMMA 3. *The accepted set of customers by StablePair is at least that of Copycat, i.e., $\mathcal{A}^C \subseteq \mathcal{A}^S$.*

We now give another useful way of characterizing a stable pair which follows immediately from the definition. The pair (Q,T) is stable if and only if

$$R(S) \geq P(Q,T) - P(Q,T\backslash S) \qquad \forall S \subseteq T. \tag{1}$$

Note that (1) above is equivalent to not having any solution to the offline problem defined on Q and T that is strictly better than accepting T. As already mentioned, the StablePair Algorithm achieves a stronger bound for the rejection costs $R(\mathcal{R})$. Specifically, compared to Lemma 1, in Lemma 4 below the term $R(\mathcal{A} \cap \mathcal{R}^*)$ is no longer needed in the bound, which will later lead to better overall competitive ratios than Copycat.

Lemma 4. *Assume that $P(\cdot)$ is nondecreasing. The StablePair Algorithm for online customer selection problems has rejection cost $R(\mathcal{R}) \leq R(\mathcal{R} \cap \mathcal{R}^*) + P(\mathcal{R} \cap \mathcal{A}^*) \leq R(\mathcal{R} \cap \mathcal{R}^*) + P(\mathcal{A}^*)$, for any offline optimal solution $(\mathcal{A}^*, \mathcal{R}^*)$.*

Proof. We first show that for all $X \subseteq \mathcal{R}$, $R(X) \leq P(X)$. Let $X \subseteq \mathcal{R}$ and assume for contradiction

$$R(X) > P(\mathcal{Q}, X) = P(X). \tag{2}$$

Let k be the last arriving customer in X. Since k was rejected by StablePair, it follows that $(\mathcal{Q}, X)$ is not a stable pair for customer k. Thus, by Eq. (1) there exists a set $S \subseteq X$ such that

$$R(S) < P(\mathcal{Q}, X) - P(\mathcal{Q}, X\backslash S) = P(X) - P(X\backslash S). \tag{3}$$

Note that $S \neq \emptyset$ or else $0 < 0$. Subtracting (3) from (2) yields

$$R(X\backslash S) > P(X\backslash S). \tag{4}$$

Now reset $X \leftarrow X\backslash S$ and repeat the analysis above until $X = \emptyset$. This will eventually give a contradiction that $0 > 0$. Now let $(\mathcal{A}^*, \mathcal{R}^*)$ be any optimal offline solution and let $X = \mathcal{R} \cap \mathcal{A}^*$. From the previous argument, it follows that

$$R(\mathcal{R} \cap \mathcal{A}^*) \leq P(\mathcal{R} \cap \mathcal{A}^*). \tag{5}$$

Adding $R(\mathcal{R} \cap \mathcal{R}^*)$ to both sides of (5) completes the proof. □

Copycat and StablePair are different in at least three ways. First, Example 4 in the Online Appendix demonstrates that the Copycat Algorithm is indeed strictly weaker with respect to bounding the rejection costs. Second, Example 5 in the Online Appendix shows that Copycat can "regret" accepting customers, where as StablePair never has regret since once a customer is accepted, there is always a stable pair (i.e., the original one) that would accept him later on. Finally, Example 6 in the Online Appendix shows that StablePair can accept customers that would never be accepted by $\mathcal{A}_k^*$ for any k.

Using Lemma 4 (which holds for any optimal solution), it is clear that if we can obtain a bound on the production costs of StablePair, then we can obtain strong competitive ratios that can be strictly better than Copycat. This is made precise in the following theorem.

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

Since Theorem 2 can be used by showing the bound for just one optimal solution, for the rest of the paper we will always assume that we are using a solution $\mathcal{A}_k^*$ that is maximal. This means that no optimal set of accepted customers to $\text{OPT}(U_k)$ is a strict superset of $\mathcal{A}_k^*$.

Although the online customer selection decisions can be done efficiently with StablePair, the decision maker still needs to calculate $P(\mathcal{A})$ in the production phase, which might be NP-hard to solve. In most practical settings, $P(\mathcal{A})$ can be calculated via integer programming or other methods. Indeed, if one desires to solve the production phase using a c-approximation algorithm (an algorithm that finds a solution in polynomial time that is no more than c times the optimal cost), then the theoretical competitive ratio would simply increase multiplicatively with c. The following theorem makes this statement precise and follows directly from the definition of an approximation algorithm and Lemma 4.

THEOREM 3. *Let $\mathcal{A}$ be the customers that StablePair accepted and let β, γ, and $c > 1$ be positive scalars. Assume there exists an optimal offline solution $(\mathcal{A}^*, \mathcal{R}^*)$ such that the optimal cost of serving $\mathcal{A}$ is $P(\mathcal{A}) \leq \beta P(\mathcal{A}^*) + \gamma R(\mathcal{R}^*) + R(\mathcal{A} \cap \mathcal{R}^*)$. Then the StablePair Algorithm is* $\max(c\beta + 1, c(\gamma+1))$*-competitive when a c-approximation algorithm for $P(\cdot)$ is used to serve $\mathcal{A}$.*

Note that the previous theorem also holds if we use a c-competitive online algorithm for solving $P(\mathcal{A})$. This may be useful if one wishes to merge the online customer selection and production phases together, which we do in Section 5.3. The remainder of this paper focuses on how to implement the StablePair Algorithm efficiently and how to bound the production costs of StablePair for several inventory and logistics problems. Since $\mathcal{A}^C \subseteq \mathcal{A}^S$ and $P(\cdot)$ is nondecreasing, then these production bounds will hold for the Copycat Algorithm as well. Not surprisingly, the production bounds we obtain are highly dependent on the combinatorial structure of the production problems. For example, the best bounds are obtained for the submodular cost production problem which is the simplest application we consider, and the highest bounds are for the JR production problem which is informally the most difficult problem we consider.

4. Submodular Cost Problems with Online Customer Selection

In this section, we consider the case where the production cost function is submodular and there is only one production option, i.e. $|\mathcal{Q}| = 1$. Furthermore, the rejection cost for each customer k, denoted by r_k, can be an arbitrary nonnegative number independent of his requirements, I_k. A function $P(\cdot)$ is *submodular* if for all $S \subseteq T \subseteq U$ and $i \notin T$, $P(S \cup \{i\}) - P(S) \geq P(T \cup \{i\}) - P(T)$. Equivalently, a function $P(\cdot)$ is submodular if for all $S, T \subseteq U$, $P(S) + P(T) \geq P(S \cap T) + P(S \cup T)$.

Submodular functions with online customer selection are simply online customer selection problems where the production cost function $P(\cdot)$ is submodular. Nondecreasing submodular functions arise naturally in many applications where there are economies of scale. We provide several examples below, all of which naturally have $|\mathcal{Q}| = 1$.

Example 1 (To Build or Not to Build). Consider the function $P(T) = K$ if $|T| > 0$, and $P(\emptyset) = 0$. This function essentially builds or implements a project if there is any customer that needs to be served. When a customer k arrives online, his willingness to pay, r_k, is revealed. Due to the simplicity of $P(\cdot)$, I_k has no particular meaning. $\square$

Example 2 (Multicast Routing). Consider a tree $\mathcal{T}$ with root v. Each edge $e_j \in \mathcal{T}$ has a cost $c_j \geq 0$. Let T be a subset of nodes and $P(T)$ be the cost of connecting the nodes in T to the root v using only edges in $\mathcal{T}$. Then $P(T)$ is clearly nondecreasing and submodular and is referred to as the multicast routing problem (Deering and Cheriton (1990)). The information I_k for customer k would be his node location, and r_k is his willingness to pay for service from that node. $\square$

Example 7 in the Online Appendix describes another application regarding polymatroid optimization and a continuous inventory replenishment problem.

4.1. Bounding Production Costs

The following theorem bounds the production costs incurred by StablePair for all submodular problems with online customer selection. From Lemma 3, this bound holds for Copycat as well.

Lemma 5. *If the StablePair (or Copycat) Algorithm is used for submodular problems with online customer selection, then $P(\mathcal{A}_k) \leq P(\mathcal{A}_k^*)$ for all k.*

Proof. By the monotonicity of $P(\cdot)$, it is sufficient to show that $\mathcal{A}_k \subseteq \mathcal{A}_k^*$, for each k. Therefore, it is now sufficient to prove that the StablePair Algorithm accepts k if and only if $k \in \mathcal{A}_k^*$.

Clearly if $k \in \mathcal{A}_k^*$, then $(\mathcal{Q}, \mathcal{A}_k^*)$ is a stable pair that StablePair can use to accept k. Now assume that k was accepted by the StablePair Algorithm, and let $(\mathcal{Q}, T)$ be the stable pair that accepted k. Then

$$R(T \backslash \mathcal{A}_k^*) \geq P(T) - P(T \cap \mathcal{A}_k^*) \geq P(T \cup \mathcal{A}_k^*) - P(\mathcal{A}_k^*).$$

The first inequality follows from (1) which is a property of a stable pair. The second inequality follows from the submodularity of $P(\cdot)$. This implies that accepting $\mathcal{A}_k^* \cup T$ is at least as cheap as only accepting $\mathcal{A}_k^*$ and rejecting $T \backslash \mathcal{A}_k^*$. Since we chose $\mathcal{A}_k^*$ to be a maximal optimal solution, this implies that $T \subseteq \mathcal{A}_k^*$ and thus $k \in \mathcal{A}_k^*$. □

Combining Theorems 1 and 2 and Lemma 5 obtains the following main result.

Theorem 4. *The Copycat and StablePair Algorithms for submodular problems with online customer selection is 2-competitive.*

Note that the proof of Lemma 5 implies that the Copycat and StablePair Algorithms are identical if Copycat always outputs the maximal optimal solution. This trivially happens if there is always a unique optimal solution, which can be achieved via a random perturbation. Interestingly enough, it turns out that the Copycat Algorithm can be implemented efficiently for submodular production problems with online customer selection. For any set of customers T,

$$\text{OPT}(T) = \min_{A \subseteq T} P(A) + R(T \backslash A) = \min_{A \subseteq T} P(A) + R(T) - R(A) = \min_{A \subseteq T} P(A) - R(A)$$

Since $R(\cdot)$ is modular, then $P(\cdot) - R(\cdot)$ is also submodular. Therefore, $\text{OPT}(T)$ is just the solution to a submodular minimization problem, which can be solved in polynomial time (McCormick (2006)). Thus Copycat (and StablePair) can be implemented efficiently. In Theorem 5 below, proved in Online Appendix, we show that the competitive ratio for any deterministic online algorithm is at least two, and thus Copycat and StablePair achieve the best possible result. This result also holds for the ELS, FL, and JR applications that we consider in later sections.

Theorem 5. *The competitive ratio for any deterministic algorithm used to solve submodular, ELS, FL, or JR problems with online customer selection is at least 2.*

5. Economic Lot Sizing Problem with Online Customer Selection

The *Economic Lot Sizing (ELS)* problem is a single item, discrete time inventory model. There is a set of customers, each with a due date and demand quantity, that need to be served by a sequence of production orders over a planning horizon of a fixed number of periods. Each order incurs a setup cost K. Each customer can only be served by an order prior to his due date. If a customer with due date t and demand d is satisfied by an order at time $s < t$, then a per unit holding cost $h > 0$ is incurred for every period and every unit of demand carried in inventory from period s to t. Without loss of generality we can assume that each customer is served from the latest order prior to his due date. The objective is to minimize the total setup ordering cost plus holding cost. The ELS problem can be solved efficiently via dynamic programming (Wagner and Whitin (1958)).

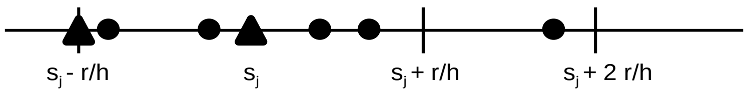
If the supplier has an option to not serve customers, or to subcontract customers to a third party, then we say the supplier can reject customers (or select only some of the customers). Let r be the per unit rejection cost. (For convenience, we assume r is an integer multiple of h.) In this full information model, the goal is to decide which customers to select and how to serve these customers. The objective is to minimize the rejection cost of the rejected customers plus the ELS setup and holding costs to satisfy the selected customers. This is called the *Economic Lot Sizing problem with Offline Customer Selection*, which can be solved efficiently (Geunes et al. (2006)).

We will focus on the *Economic Lot Sizing problem with Online Customer Selection.* Customers arrive in an online manner and the supplier needs to make an immediate selection decision before new customers arrive. When a customer k arrives, he specifies $I_k = (d_k, t_k)$, where d_k is the quantity and t_k is the due date. The rejection cost of customer k is then $r_k = rd_k$. The set of production options $\mathcal{Q}$ is the set of possible order dates. The production cost $P(Q,T)$ is then the optimal cost of the ELS problem on a subset of customers T using only the potential order dates $Q \subseteq \mathcal{Q}$. Note that we assume the per unit rejection cost is uniform among customers. In Elmachtoub and Levi (2014), it is shown that if there are customer-specific rejection costs per unit, then no deterministic nor randomized online algorithm can have a constant competitive ratio.

5.1. Bounding Production Costs

In this section, we provide a bound on the total production costs incurred by the StablePair Algorithm when applied to the ELS problem with online customer selection. It is well known that the optimal solution for the ELS problem has a zero inventory ordering (ZIO) property (Wagner and Whitin (1958)). This means that orders are placed only when the on-hand inventory level is zero. This implies that each order covers all demands with due dates between the order date and the due date of the last customer served by the order. We refer to this range of due dates as the respective *setup interval* of the order. We now prove two simple lemmas, proved in the Online Appendix, regarding the structure of an optimal offline solution.

Figure 1 Demands near an optimal order s_j.



Note. The triangles denote optimal orders and the circles denote due dates of customers. The first two customers will be served by the extra order placed at $s_j - r/h$. The next two customers are in $\mathcal{A}^*$ and are served by the order at s_j. The last customer will also be served by the order at s_j with holding cost at most $2r$ per unit.

Lemma 6. *Let $(Q,T) \subseteq (\mathcal{Q},U)$ be a stable pair. Each setup interval induced by the ELS solution to $P(Q,T)$ must have a length of at most r/h periods.*

Lemma 7. *Let $(Q,T) \subseteq (\mathcal{Q},U)$ be a stable pair and let $[a,b]$ be a setup interval from the ELS solution for $P(Q,T)$. Then the interval $[a,b]$ must contain the due date of a customer in $\mathcal{A}^*$. Therefore, $[a,b]$ must intersect a setup interval from the ELS solution for $P(\mathcal{A}^*)$.*

Using Lemmas 6 and 7, it is next shown that the StablePair Algorithm incurs production cost at most twice the optimal offline cost. From Lemma 3, this also holds for the Copycat Algorithm.

Lemma 8. *The production cost incurred by StablePair (Copycat) Algorithm for the ELS problem is within twice the optimal offline cost, specifically, $P(\mathcal{A}) \le 2P(\mathcal{A}^*) + 2R(\mathcal{A} \cap \mathcal{R}^*) \le 2C^*(U)$.*

Proof. We will explicitly construct a feasible production plan that serves all of the customers in $\mathcal{A}$ and show that its cost is at most twice the optimal offline cost $C^*(U)$. This will also clearly hold for the optimal (minimum cost) production plan to serve the customers in $\mathcal{A}$. Let $s_1,\dots,s_m$ denote the order periods in the ELS solution to $P(\mathcal{A}^*)$, i.e., the optimal production plan for customers accepted by the optimal offline solution. Let K^* and H^* represent the setup costs and holding costs, respectively, induced by $P(\mathcal{A}^*)$. Consider a production plan that places orders at times $s_1,\dots,s_m$ as well as $s_1 - r/h,\dots,s_m - r/h$. The total cost of these orders is exactly $2K^*$. Now serve all customers in $\mathcal{A} \cap \mathcal{A}^*$ by the same order that they were served in the ELS solution for $P(\mathcal{A}^*)$. The resulting total holding costs for these customers is at most H^*. The only customers left to be served are those in $\mathcal{A} \cap \mathcal{R}^*$.

Next, consider each customer $k \in \mathcal{A} \cap \mathcal{R}^*$, i.e., a customer accepted by StablePair but not by the optimal offline solution. The claim is that there exists some optimal order s_j such that customer k's due date $t_k \in [s_j - r/h, s_j + 2r/h]$. (See Figure 1 for an example of the accepted customers near an optimal order s_j.) Now consider the stable pair (Q,T) that made StablePair accept customer k. Lemma 7 implies that the setup interval within the production plan for $P(Q,T)$ that contained k also contains some customer $k' \in \mathcal{A}^*$. Let s_j be the order from which customer k' is served in the production plan for $P(\mathcal{Q},\mathcal{A}^*)$. By Lemma 6, the lengths of the setup intervals in any ELS solution are at most r/h. It then follows that $t_k \in [s_j - r/h, s_j + 2r/h]$. In the construction, customer k is served from the order $s_j - r/h$ if $t_k \in [s_j - r/h, s_j)$ and from s_j if $t_k \in [s_j, s_j + 2r/h]$. In either one of these cases, the per unit holding cost incurred by k is at most $2r$. However, customer k was rejected by the optimal offline solution and incurred a cost r. It follows that the total holding cost incurred by customers in $\mathcal{A}\cap\mathcal{R}^*$ is at most $2R(\mathcal{A}\cap\mathcal{R}^*)$. Summing up all the production costs yields an upper bound on $P(\mathcal{A})$ of $2K^* + H^* + 2R(\mathcal{A}\cap\mathcal{R}^*) \le 2P(\mathcal{A}^*) + 2R(\mathcal{A}\cap\mathcal{R}^*) \le 2C^*(U)$. $\square$

Combining Theorems 1 and 2 with Lemma 8, we obtain the following main result.

Theorem 6. *The Copycat and StablePair Algorithms for the Economic Lot Sizing Problem with Online Customer Selection are both 3-competitive.*

Example 8 in the Online Appendix demonstrates that the analysis above is tight for both algorithms. We note that the analysis of the Copycat and StablePair Algorithms can be extended easily for non-decreasing setup costs K_t and more general holding cost structure. Specifically, if the cost of holding a unit from period s to t is h_{st}, then the analysis holds if h_{st} is subadditive (for any $s \le u \le t$, $h_{st} \le h_{su} + h_{ut}$).

5.2. Including Soft Capacities

One interesting extension of the ELS problem is when there are soft capacities on the number of units produced in every order. Specifically, at most c units can be produced in an order, but there is no limit on the number orders placed in a given time period. Thus, capacity is soft in the sense that more customers can always be served if necessary, although no more than c units of demand can be served from a particular order. In this setting, every order placed still incurs a cost K, and the total ordering cost in a period is K times the number of orders. Theorem 7 below, proved in the Online Appendix, shows that our algorithms still perform well under this setting, albeit at an increased competitive ratio of 4.

Theorem 7. *The Copycat and StablePair Algorithms for the Economic Lot Sizing Problem with Soft Capacities and Online Customer Selection are both 4-competitive.*

5.3. Including Online Production

Another extension of the ELS problem with online customer selection is when the online customer selection phase and production phase overlap. In this scenario, the supplier learns about each customer k at their arrival time e_k, which may be *after* the due dates of other customers. The information for each customer k is now $I_k = (e_k, d_k, t_k)$, where d_k and t_k are the respective demand quantity and due date for customer k. If customer k is accepted, he must be satisfied by an order in the time window $[e_k, t_k]$. Naturally we assume that $e_k \le t_k$ and the arrival dates are chronological. The ELS problem with online customer selection and online production is more difficult, and in fact, Example 9 in the Online Appendix demonstrates that we cannot obtain a constant competitive ratio for this problem, even when the optimal offline solution must also respect the arrival times.

However, if the due dates of the customers are in chronological order, then we can obtain a positive result as stated in the following theorem which is proved in the Online Appendix.

Theorem 8. *Consider the Economic Lot Sizing Problem with Online Customer Selection and Online Production. Assume that the due dates for the customers are in chronological order, i.e.,* $t_1 \le t_2 \le \ldots \le t_N$. *Then the Copycat Algorithm combined with a 2-competitive heuristic of Axsater (1982) for the online ELS problem obtains an overall competitive ratio of 3.*

5.4. StablePair Implementation

Although the Copycat Algorithm for the ELS problem with online customer selection can be implemented in polynomial time by using dynamic programming, we describe the StablePair implementation since it is faster and also serves as a foundation for how to implement StablePair for the JR variant in Section 6.2. Lemma 9 below, proved in the Online Appendix, describes an exact method for implementing the StablePair Algorithm. The idea is that one can reduce the search space for stable pairs by only considering those with one production option (order date).

Lemma 9. *The StablePair Algorithm accepts customer k if and only if there exists a stable pair (Q,T) with $k \in T$ such that*

1. *Q consists of one order date, i.e. $Q = \{t\}$ for some time t.*
2. *T includes exactly all customers that can be served from t with at most r per unit in holding cost, i.e. $T = \{j \in U_k | t_j \in [t, t + r/h]\}$.*
3. *The rejection costs of T exceed the production costs of serving T from an order at t, i.e. $R(T) \ge K + \sum_{j \in T} h(t_j - t)d_j$.*

Note that without loss of generality, one only needs to consider the potential order dates that correspond to due dates of U_k (since ZIO policies are optimal). Thus, the maximum number of candidate stable pairs is at most N. Since checking stability of each pair takes $O(N)$ time, then this characterization of the StablePair Algorithm provides an efficient implementation.

6. Joint Replenishment Problem with Online Customer Selection

The *Joint Replenishment (JR)* problem is a natural extension of the ELS problem with multiple item types, indexed $1,\dots,M$. The goal is to serve a set of customers, each with a quantity, due date, and item type, by a sequence of production orders over a planning horizon of a fixed number of periods. Each order incurs a joint setup cost of K_0. In addition, for each item type i ordered, an item setup cost of K_i is incurred. Each customer can only be served by orders that contain his item type and are before his due date. As with the ELS problem, there is also a per unit holding cost h. The objective is to minimize the total setup ordering cost plus holding cost. This problem was shown to be NP-hard in Arkin et al. (1989), and admits a current best approximation guarantee of 1.80 due to Levi et al. (2008). More general JR problems are considered in Cheung et al. (2013).

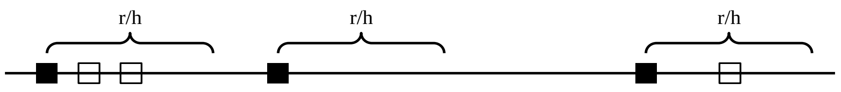
When the supplier does not have to serve all the customers, but can now reject customers at a per unit cost r, then the problem becomes even more complex. (For convenience, we assume r is an integer multiple of h.) The supplier must now decide on the optimal set of customers to select and decide how to serve them. Specifically, the goal is to minimize the total rejection costs plus the cost of the JR problem on the accepted customers. This is called the *Joint Replenishment Problem with Offline Customer Selection*. This problem was first studied in Geunes et al. (2011) who gave a 2.35-approximation for the market selection variant.

In this work, we focus on the *Joint Replenishment Problem with Online Customer Selection*. Like the previous models, customers arrive one after the other and must be either accepted or rejected immediately. Specifically, when a customer k arrives, his requirements $I_k = (d_k, t_k, i_k)$ are observed, where d_k specifies the quantity, t_k specifies the due date, and i_k specifies the item type. The rejection cost is then $r_k = rd_k$. We again define the set of production options $\mathcal{Q}$ as the set of potential order dates. The production cost function $P(Q,T)$ is now the optimal cost of the JR problem with production options Q and customers T.

6.1. Bounding Production Costs

In this subsection, we shall bound the production costs incurred by the StablePair Algorithm for the JR problem with online customer selection. For now, assume that there exists a black box to run the StablePair Algorithm. In Section 6.2 we describe an efficient implementation of StablePair. Again the setup interval of an order is defined as the range of due dates that the order serves, regardless of item type. Lemmas 6 and 7 both still hold with the same proofs for the JR problem with online customer selection. The following lemma describes the worst-case production costs that will be incurred.

Figure 2 Constructing the set T^i.



Note. The six squares represent the order dates that make up $\hat{T}^i$. The solid squares represent the order dates that make up T^i and the hollow squares represent the dates that were pruned. By default, the first square must be solid. Then all squares within r/h are pruned and the process is repeated.

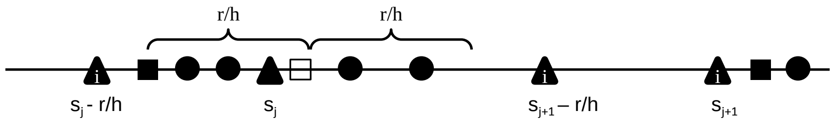
LEMMA 10. *If the StablePair (Copycat) Algorithm is used for the JR problem with online customer selection, then* $P(\mathcal{A}) \le 2P(\mathcal{A}^*) + R(R^*) + 2R(\mathcal{A} \cap \mathcal{R}^*) \le 3C^*(U)$.

Proof. The proof is similar in spirit to the proof of Lemma 8, but requires a more involved construction and analysis. Let $s_1, \ldots, s_m$ denote the times of the orders in the optimal production plan for $P(\mathcal{A}^*)$. The cost of the optimal production plan can be decomposed into the total setup costs, K^*, and the total holding costs, H^*. Let $\mathcal{A}^i$ simply denote the customers with item type i that are in $\mathcal{A}$. For each customer $k \in \mathcal{A}^i$, let a_k be the order date that serves customer k in the stable pair solution that StablePair used to accept customer k, and let $\hat{T}^i$ denote the set of these order dates sorted from earliest to latest. Construct $T^i \subseteq \hat{T}^i$ by processing $\hat{T}^i$ in order from earliest to latest, and remove any order date that is within r/h periods of the previous order date that was not removed. (See Figure 2 for an example of this construction.) Now consider the following two sets, X^i and Y^i. The set X^i is defined to be the set of all type i customers that have due date within $[t, t + r/h]$ for some $t \in T^i$. Note that X^i is not necessarily contained in the set $\mathcal{A}^i$. The set Y^i is then defined to be $\mathcal{A}^i \backslash X^i$.

To construct a solution that serves all the customers in $\mathcal{A}$, we first create the same sequence of orders as in $P(\mathcal{A}^*)$ in periods $s_1, \ldots, s_m$ and incur setups costs of K^*. Note that the item orders are replicated as well. All the customers in $\mathcal{A} \cap \mathcal{A}^*$ are then served in the same way as in the production plan for $P(\mathcal{A}^*)$, and thereby incur holding costs of at most H^*. Now we create a sequence of duplicate orders shifted back by time r/h, i.e. at periods $s_1 - r/h, \ldots, s_m - r/h$, and incur another K^*.

Next, consider each item type i separately. Assume for now that for each order date $t \in T^i$, there exists an order s_j in the production plan for $P(\mathcal{A}^*)$, such that either $s_j \in [t - r/h, t]$ or $s_j - r/h \in [t - r/h, t]$. Moreover, order s_j (or $s_j - r/h$) includes item i. Assuming this property holds, one can bound the holding costs for the customers in $\mathcal{A}^i \cap \mathcal{R}^*$. Specifically, the property ensures that all type i customers with due dates in $[t, t + r/h]$ can be served with holding cost at most $2r$ per unit. By definition of X^i, this means that the customers in $X^i \cap \mathcal{A}^i \cap \mathcal{R}^*$ will be served

Figure 3 Serving the customers in $\mathcal{A}^i$.



Note. In this figure, the triangles are orders, the circles are due dates for customers in $\mathcal{A}^i$, the solid squares are times in T^i, and the hollow squares are times in $\hat{T}^i$ that were pruned. In this picture, only the orders with an i written contain an order of type i. The first pair of orders is associated with Case 2 and the second pair of orders is associated with Case 1. The first two demands are in $X^i \cap \mathcal{A}^i \cap \mathcal{R}^*$. The next two demands are in $k \in Y^i \cap \mathcal{R}^*$. The last demand is in $\mathcal{A}^*$.

with total holding costs at most $2r$ per unit. The remaining customers left to be served are those in $Y^i \cap \mathcal{R}^*$. Consider customer $k \in Y^i \cap \mathcal{R}^*$. By construction of T^i, it follows that there exists a $t \in T^i$ such that $t \le a_k \le t + r/h \le a_k + r/h$. In addition, from the definition of a_k and Lemma 6, it follows that $t_k \in [a_k, a_k + r/h]$. By the property assumed above, there exists an order that includes item type i within r/h before t. The holding cost from that order to t, t to a_k, and a_k to t_k are each at most r per unit. Thus, customer k can be served with a holding cost of at most $3r$ per unit. (See Figure 3 for an example.)

It is now sufficient to ensure that indeed for each $t \in T^i$, there exists an order of type i within r/h time periods earlier than t. To achieve this, extra item orders will be added to the construction. From Lemma 7, it follows that the setup interval corresponding to t intersected some optimal setup interval starting at s_j. (If there is a choice of intersections, choose s_j that contains an item order of type i if one exists.) From Lemma 6, it follows that $s_j - r/h \le t \le s_j + r/h$. We now consider two cases and show how to enforce the property in each case.

Case 1: There is a type i customer in $\mathcal{A}^*$ with due date in $[t, t + r/h]$. By construction, this implies that there are type i orders at s_j and $s_j - r/h$, respectively.

Case 2: There is no type i customer in $\mathcal{A}^*$ with due date in $[t, t + r/h]$. If $s_j - r/h \le t < s_j$, then we place an *extra* item order of type i at the joint order located at time $s_j - r/h$. Otherwise, $s_j \le t \le s_j + r/h$ and we place the extra order of type i at s_j. Since t corresponds to a setup interval from a stable pair solution, it follows from (1) that there exists a set of customers with due dates in $[t, t + r/h]$ whose rejection costs are greater than K_i. Under the case assumption, the type i demands with due dates in $[t, t + r/h]$ are all in $\mathcal{R}^*$. Furthermore, all customers with due dates in $[t, t + r/h]$ are in X^i. Thus, the extra item orders have cost at most $R(X^i \cap \mathcal{R}^*)$. Each customer in $X^i \cap \mathcal{R}^*$ can be used at most once to pay for an extra item order by the spacing of the times we enforced in the construction of T^i.

The total cost incurred by the construction is $K^* + H^* + K^* + \sum_{i=1}^{M}(R(X^i \cap \mathcal{R}^*) + 2R(X^i \cap \mathcal{A}^i \cap \mathcal{R}^*) + 3R(Y^i \cap \mathcal{R}^*)) \leq 2P(\mathcal{A}^*) + R(\mathcal{R}^*) + 2R(\mathcal{A} \cap \mathcal{R}^*)$ which completes the proof. □

Combining Theorems 1 and 2 with Lemma 10 obtains the following main result.

Theorem 9. *The Copycat and StablePair Algorithms for the Joint Replenishment problem with Online Customer Selection are 4-competitive and 3-competitive, respectively.*

Note that the analysis is tight for the StablePair Algorithm by Example 8 in the Online Appendix since JR is a generalization of ELS. In Section 6.2, we describe how to implement StablePair efficiently. Theorem 10 below, proved in the Online Appendix, gives a slightly weaker result for the case when each item type i has a specific per unit rejection cost r^i and a specific per unit holding cost h^i.

Theorem 10. *The Copycat and StablePair Algorithms for the JR problem with Online Customer Selection and item-specific holding and rejection costs are* $(3 + \frac{max_i r^i/h^i}{min_i r^i/h^i})$*-competitive and* $(2 + \frac{max_i r^i/h^i}{min_i r^i/h^i})$*-competitive, respectively.*

6.2. StablePair Implementation

Implementing Copycat problem for the JR problem with online customer selection requires solving an NP-hard problem. However, StablePair can be implemented efficiently in time polynomial in the number of customers and items. Lemma 11 below, proved in the Online Appendix, shows that a stable pair exists if and only if there is a stable pair using only one order date that satisfies the properties below.

Lemma 11. *The StablePair Algorithm accepts customer k if and only if there exists a stable pair (Q,T) with $k \in T$ such that*

1. *Q consists of one order date, i.e. $Q = \{t\}$ for some time t.*
2. *If T contains customers of type i, then it contains exactly all type i customers with due dates in $[t, t + r/h]$. Let $T^i = \{j \in U_k | t_j \in [t, t + r/h] \text{ and } i_j = i\}$ and let $\mathcal{I}$ be a subset of item types. This property is equivalent to the property that $T = \cup_{i \in \mathcal{I}} T^i$.*
3. *The set of item types in T are those who can pay for their item ordering cost plus the holding costs, i.e. $\mathcal{I} = \{i | R(T^i) \geq K_i + \sum_{j \in T^i} h(t_j - t)d_j\}$.*
4. *The total rejection costs of T are at least the production costs of serving T from t, i.e. $R(T) \geq K_0 + \sum_{i \in \mathcal{I}} \left(K_i + \sum_{j \in T^i} h(t_j - t)d_j\right)$.*

This lemma provides an efficient implementation of StablePair since the set of potential stable pairs is at most N, and checking the four properties for each pair takes $O(MN)$ time.

7. Facility Location Problem with Online Customer Selection

The metric *Facility Location (FL)* problem is a well studied NP-hard problem. The goal is to serve a set of customers, each with a specified demand quantity and location, by opening a set of facilities. There are M potential facilities, indexed by $j = 1, \ldots, M$. The opening cost of facility j is f_j. Each customer k is served by the nearest open facility, and pays a service cost $c(j,k)$ per unit if served by facility j. The assumption is that $c(\cdot,\cdot)$ induces a metric (symmetric and satisfies triangle inequality) over the facilities and customers. The goal is to serve all the customers so as to minimize the total facility costs plus service costs. The first constant factor approximation algorithm was given by Shmoys et al. (1997), and the current best result is 1.488 due to Li (2011).

When the supplier does not have to serve all the customers, but can now reject customers at a per unit cost r, then the problem becomes even more complex. The supplier must now decide on the optimal set of customers to accept and decide how to serve them. Specifically, the goal is to minimize the total rejection costs plus the cost of the FL problem on the accepted customers. This is called the *Facility Location Problem with Offline Customer Selection.* The first approximation algorithm was given by Charikar et al. (2001) (who call this FL with outliers), and the current best approximation factor of 1.85 is due to Xu and Xu (2009).

Here we focus on the *Facility Location Problem with Online Customer Selection.* When a customer k arrives online, we observe $I_k = (d_k, l_k)$, where d_k specifies the quantity and l_k specifies the location. The rejection cost is then $r_k = rd_k$. Let $\mathcal{Q}$ denote the potential set of facilities. The production cost function $P(Q,T)$ is now the optimal cost of the FL problem for a given set of customers T that can only use the facilities in Q.

7.1. Bounding Production Costs

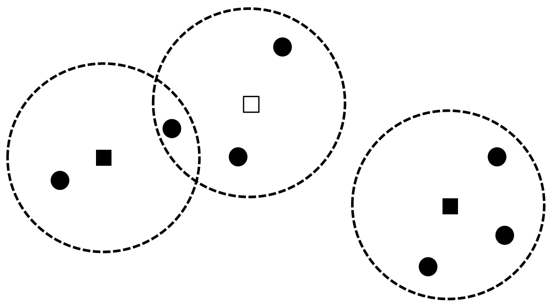
Now we will bound the production costs incurred by the StablePair Algorithm for the FL problem with online customer selection. This will require the two following lemmas which are proved in the Online Appendix and are similar in spirit to Lemmas 6 and 7.

Lemma 12. *Let (Q,T) be a stable pair. For each customer $k \in T$, the per unit service cost incurred by the FL solution to $P(Q,T)$ is at most r.*

For each customer $k \in \mathcal{A}$ accepted by the StablePair Algorithm, define (Q_k, T_k) to be the stable pair used by StablePair to accept k. Let $S_k \subseteq T_k$ be the set of customers served by the same facility as k in $P(Q_k, T_k)$ and let q_k denote this shared facility.

Lemma 13. *For each $k \in \mathcal{A}$, the set $S_k \cap \mathcal{A}^*$ is nonempty.*

Figure 4 Serving the customers in $\mathcal{A}$.



Note. In this figure, the squares represent facilities and the circles represent locations for customers in $\mathcal{A}$. The solid squares represent open facilities and the hollow squares are closed facilities. The circle around each facility represents the service area where the service cost per unit is at most r from the respective facility. The customers in the circles corresponding to open facilities are in $\mathcal{A}\cap\mathcal{A}^*$. The remaining customers in the circles corresponding to closed facilities are in $\mathcal{A}\cap\mathcal{R}^*$ and are no more than $3r$ away from an open facility.

We now proceed to bound the production costs of StablePair and therefore Copycat.

LEMMA 14. *If the StablePair (Copycat) Algorithm is used for the FL problem with online customer selection, then* $P(\mathcal{A}) \le P(\mathcal{A}^*) + 3R(\mathcal{A}\cap\mathcal{R}^*) \le 3C^*(U)$.

Proof. We will construct a feasible solution to serve the customers in $\mathcal{A}$ with cost at most three times the optimal offline cost $C^*(U)$. We simply open all the facilities in the FL solution to $P(\mathcal{A}^*)$ and serve all the customers in $\mathcal{A}$ by the nearest facility.

By construction, the facility costs plus the service costs for $\mathcal{A}\cap\mathcal{A}^*$ is bounded by $P(\mathcal{A}^*)$. Now consider a customer $k \in \mathcal{A}\cap\mathcal{R}^*$. From Lemma 13, it follows that there exists a customer $l \in S_k \cap \mathcal{A}^*$. Let q_l^* denote the facility from which customer l is served in $P(\mathcal{A}^*)$. Lemma 12 implies that $c(k,q_k)$, $c(q_k,l)$, and $c(l,q_l^*)$ are all at most r. Since c is a metric, it follows that $c(k,q_l^*) \le c(k,q_k) + c(q_k,l) + c(l,q_l^*) \le 3r$. Thus, the per unit service cost for customer k is at most $3r$. (See Figure 4 for an example.) This completes the proof since the construction has cost at most $P(\mathcal{A}) \le P(\mathcal{A}^*) + 3R(\mathcal{A}\cap\mathcal{R}^*) \le 3C^*(U)$. $\square$

Combining Theorems 1 and 2 with Lemma 14 obtains the following main result.

THEOREM 11. *The Copycat and StablePair Algorithms for the Facility Location problem with Online Customer Selection are 4-competitive and 3-competitive, respectively.*

7.2. Improving Bounds via Scaling

The results for FL with online customer selection can be improved via a scaling technique to get a competitive ratio of $1+\sqrt{2}$. In particular, the rejection costs are scaled by a factor $\alpha > 0$, and then the StablePair Algorithm is applied to the scaled input. The purpose of this idea is to "hedge" against the future by giving a different weighting to the rejection costs. Drop the superscript S and denote $\mathcal{A}_\alpha$ and $\mathcal{R}_\alpha$ as StablePair's decisions under the scaled input. In Lemma 15 below, we obtain a new bound for the rejection costs. The proof is in the Online Appendix.

LEMMA 15. *The StablePair Algorithm with the rejection costs scaled by a parameter α has total rejection cost $R(\mathcal{R}_\alpha) \le R(\mathcal{R}_\alpha \cap \mathcal{R}^*) + \frac{1}{\alpha} P(\mathcal{R}_\alpha \cap \mathcal{A}^*)$.*

Note that the bound on production costs in Lemma 14 is unbalanced. By scaling down the rejection costs by a factor $\alpha < 1$, we obtain Lemma 16 which we prove in the Online Appendix.

LEMMA 16. *The StablePair Algorithm with scaling for the FL problem with online customer selection has production costs $P(\mathcal{A}_\alpha) \le P(\mathcal{A}^*) + (2\alpha + 1) R(\mathcal{A}_\alpha \cap \mathcal{R}^*)$ when $\alpha < 1$.*

Combining Lemmas 15 and 16 obtains the following theorem, whose analysis is tight according Example 10 in the Online Appendix.

THEOREM 12. *The StablePair Algorithm with scaling for the Facility Location problem with online customer selection is $(1+\sqrt{2})$-competitive when $\alpha = \frac{1}{\sqrt{2}}$.*

7.3. StablePair Implementation

Using the Copycat Algorithm for the FL problem with online customer selection will be inefficient since solving the offline problem is NP-hard. Implementing the StablePair Algorithm, however, can be implemented efficiently in time polynomial in the number of customers and facilities. Lemma 17 below, proved in the Online Appendix, shows that a stable pair exists if and only if there is a stable pair with one facility with the properties below.

LEMMA 17. *The StablePair Algorithm accepts customer k if and only if there exists a stable pair (Q, T) containing k such that*

1. *Q consists of one facility location, i.e. $Q = \{j\}$ for some facility j.*
2. *T are all customers that can be served from j using at most r in service cost per unit, i.e. $T = \{i \in U_k | c(i,j) \le r\}$.*
3. *The rejection costs of T exceed the production costs of serving T from facility j, i.e. $R(T) \ge f_j + \sum_{i \in T} c(i,j) d_i$*

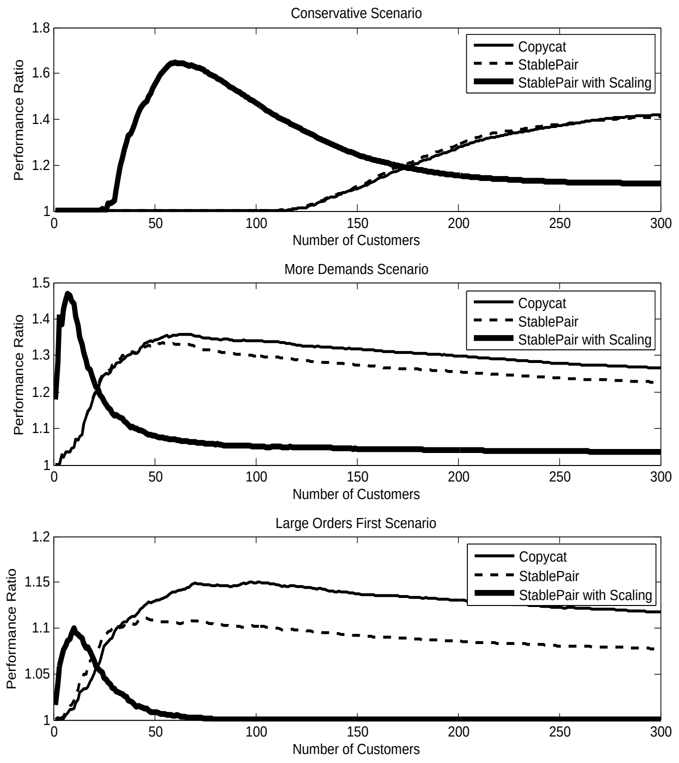
Since the number of possible stable pairs is at most M, and checking the conditions takes at most $O(N)$, then clearly this lemma provides an efficient way to implement the StablePair Algorithm.

8. Computational Results

We now present the results of computational experiments that test the typical empirical performance of our algorithms on the Economic Lot Sizing Problem with Online Customer Selection. We carried out three different scenarios of the same experiment. The parameters for each scenario were $K = 100, h = 1$, and $r = 5$. The time horizon was for one month with periods from 1 to 30. There were $N = 300$ customer arrivals simulated for each scenario. Copycat, StablePair, and StablePair with scaling factor of two (the rejection cost of each customer is weighted double the true value) were each tested for the same sequence of customer arrivals. After every customer k, the performance ratio, $C(U_k)/C^*(U_k)$, was calculated. Note that we are comparing ourselves to the optimal offline solution, which is a benchmark that is impossible to achieve. For all three scenarios, the due dates of the customers were drawn uniformly at random across the time horizon. We believe this distribution is the worst case for the supplier since she can never make a meaningful estimate of where the next customer's due date will be. In the first scenario, 'Conservative', the demands of each customer comprise of one unit ($d_k = 1$). In the second scenario, 'More Demands', the demand quantity is drawn uniformly at random from 1 to 10. Finally, in the third scenario, 'Large Orders First', the customers also have a random demand quantity between 1 and 10 but the first two orders are very large orders placed on the first and fifteenth, i.e. at times 1 and 15, representing the realistic situation where the supplier has at least one major routine customer. For each scenario, the results were averaged over 100 different customer sequences. The results are in Figure 5.

Based on these experiments, we see that even in the 'Conservative' scenario, the costs of Copycat and StablePair are at most 1.5 times larger than the optimal offline cost (which is not achievable in practice). The upper bound of 1.5 held up in all 25 runs of the scenario. Note that the performance ratios are much lower than the theoretical guarantee of 3. The 'Conservative' graph in Figure 5 indicates that Copycat and StablePair start out perfect since both OPT and the algorithms are rejecting everything because there are not enough customers to warrant production. Then around 125 customer arrivals OPT begins accepting some customers while Copycat and StablePair do the same and therefore begin incurring lots of fixed costs on top of the rejection costs. Eventually, as the number of customers gets very large, OPT, Copycat, and StablePair begin to accept everything and the performance ratios stabilize. (Note that $C^*(U_k)$ converges to $30K$ while the costs of the algorithms are $30K$ plus some initial rejection costs.) Since StablePair accepts a superset of what Copycat accepts, then StablePair outperforms Copycat as the number of customers gets large. StablePair with scaling does poorly in the beginning since it accepts customers before OPT does,

Figure 5 Experimental Results.



Note. Three experiments detailing the actual performance ratio of three algorithms on the ELS problem with online customer selection.

but later on it significantly outperforms the other algorithms. Moreover, StablePair is about 40 times faster than Copycat, making it computationally appealing as well.

Similar observations hold for the 'More Demands' and 'Large Orders First' graphs in Figures 5, except that the benefits of scaling are not as significant since more customers were accepted early on which makes it more difficult to make errors. Specifically, Copycat/StablePair accept more customers when they come in larger batches, so as the number of customers gets large this improves the performance ratio significantly. If there is some information known about the customer process, this can help decide on a reasonable choice for the scaling factor. The more customers one anticipates, the larger the scaling factor should be. Also note that in addition to StablePair being computationally much faster than Copycat, it typically outperforms Copycat. In the Online Appendix, Tables 1, 2, 3 describe the performance of our algorithms for the same scenarios but with different values of r. As r increases, all the algorithms perform better, while the previous insights still hold.

Finally, we tested our algorithms on the JR problem with online customer selection. Specifically, we chose a problem with three item types, and customer arrivals had due dates and item types chosen uniformly at random. The parameters were $N = 300$, $M = 3$, $K_0 = 100$, $K_1 = K_2 = K_3 = 20$, $r = 10$, and $h = 1$, with a time horizon of 30 days. We simulated the same three scenarios described previously and found that similar insights still hold. However, for this problem, StablePair was over 1,000 times faster than Copycat, since Copycat required solving an integer program. In the Online Appendix, Figure 6 and Table 4 summarize the results for the JR problem with Online Customer Selection under the different scenarios.

Acknowledgments
The authors thank the AE and the anonymous referees for many comments and suggestions that ultimately improved the quality of the manuscript. The research of the first author was conducted while at the Operations Research Center at Massachusetts Institute of Technology and was funded with government support under and awarded by DoD, Air Force Office of Scientific Research, National Defense Science and Engineering Graduate (NDSEG) Fellowship, 32 CFR 168a. The research of the second author was partially supported by NSF grant CMMI-0846554 (CAREER Award), AFOSR awards FA9550-11-1-0150 and FA9550-08-1-0369, an SMA grant and the Buschbaum Research Fund of MIT.

References

Albers, S. 2003. Online algorithms: A survey. *Mathematical Programming B* **97**(1) 3–26.

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

Axsater, S. 1982. Worst case performance for lot sizing heuristics. *EJOR* **9**(4) 339–343.

Ball, M.O., M. Queyranne. 2009. Toward robust revenue management: Competitive analysis of online booking. *Oper. Res.* **57**(4) 950–963.

Bhaskaran, S., K. Ramachandran, J. Semple. 2010. A dynamic inventory model with the right of refusal. *Management Science* **56**(12) 2265–2281.

Bienstock, D., M.X. Goemans, D. Simchi-Levi, D. Williamson. 1993. A note on the prize collecting traveling salesman problem. *Mathematical Programming* **59**(1) 413–420.

Buchbinder, N., T. Kimbrelt, R. Levi, K. Makarychev, M. Sviridenko. 2014. Online make-to-order joint replenishment model: primal dual competitive algorithms. *Oper. Res.* Forthcoming.

Caldentey, R., L. M. Wein. 2006. Revenue management of a make-to-stock queue. *Oper. Res.* **54**(5) 859–875.

Carr, S., I. Duenyas. 2000. Optimal admission control and sequencing in a make-to-stock/make-to-order production system. *Oper. Res.* **48**(5) 709–720.

Charikar, M., S. Khuller, D.M. Mount, G. Narasimhan. 2001. Algorithms for facility location problems with outliers. *Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics, 642–651.

Cheung, M., A.N. Elmachtoub, R. Levi, D.B. Shmoys. 2013. The submodular joint replenishment problem. *Unpublished Manuscript* .

Cornuéjols, G., G.L. Nemhauser, L.A. Wolsey. 1990. The uncapacitated facility location problem. P. Mirchandani, R. Francis, eds., *Discrete location theory*, vol. 23. Wiley-Interscience, 119–171.

Deering, S.E., D.R. Cheriton. 1990. Multicast routing in datagram internetworks and extended LANs. *ACM Transactions on Computer Systems (TOCS)* **8**(2) 85–110.

Elmachtoub, A.N., R. Levi. 2014. From cost sharing mechanisms to online selection problems. *Mathematics of Opeartions Research* Forthcoming.

Federgruen, A., Y.S. Zheng. 1992. The joint replenishment problem with general joint cost structures. *Oper. Res.* **40**(2) 384–403.

Fotakis, D. 2008. On the competitive ratio for online facility location. *Algorithmica* **50**(1) 1–57.

Gallien, J., Y. Le Tallec, T. Schoenmeyr. 2004. A model for make-to-order revenue management. Tech. rep., Citeseer.

Geunes, J., R. Levi, H.E. Romeijn, D.B. Shmoys. 2011. Approximation algorithms for supply chain planning and logistics problems with market choice. *Mathematical programming* **130**(1) 85–106.

Geunes, J., H.E. Romeijn, K. Taaffe. 2006. Requirements planning with pricing and order selection flexibility. *Oper. Res.* **54**(2) 394.

Goemans, M.X., M. Skutella. 2000. Cooperative facility location games. *Proceedings of the eleventh annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics, 76–85.

Goemans, M.X., D.P. Williamson. 1995. A general approximation technique for constrained forest problems. *SIAM Journal on Computing* **24**(2) 296–317.

Golrezaei, Negin, Hamid Nazerzadeh, Paat Rusmevichientong. 2014. Real-time optimization of personalized assortments. *Management Science* **60**(6) 1532–1551.

He, S., J. Zhang, S. Zhang. 2012. Polymatroid optimization, submodularity, and joint replenishment games. *Operations Research* **60**(1) 128–137.

Hintsches, A., T. Spengler, T. Volling, K.i Wittek, G. Priegnitz. 2010. Revenue management in make-to-order manufacturing: case study of capacity control at thyssenkrupp vdm. *BuR Business Research Journal* **3**(2).

Jaillet, P., X. Lu. 2013. Online traveling salesman problems with rejection options. Tech. rep., Working paper, Operations Research Center, Massachusetts Institute of Technology.

Keskinocak, P., R. Ravi, S. Tayur. 2001. Scheduling and reliable lead-time quotation for orders with availability intervals and lead-time sensitive revenues. *Management Science* **47**(2) 264–279.

Levi, R., R. Roundy, D. Shmoys, M. Sviridenko. 2008. A constant approximation algorithm for the one-warehouse multiretailer problem. *Management Science* **54**(4) 763–776.

Levi, R., R.O. Roundy, D.B. Shmoys. 2006. Primal-Dual Algorithms for Deterministic Inventory Problems. *Math. Oper. Res.* **31**(2) 267–284.

Li, S. 2011. A 1.488 approximation algorithm for the uncapacitated facility location problem. *Automata, Languages and Programming* 77–88.

Lieb, R., B. A. Bentz. 2005. The use of third-party logistics services by large american manufacturers: the 2004 survey. *Transportation Journal* 5–15.

Lovász, L. 1983. Submodular functions and convexity. *Mathematical Programming The State of the Art*. Springer, 235–257.

McCormick, S.T. 2006. Submodular function minimization. K. Aardal, G.L. Nemhauser, R. Weismantel, eds., *Handbooks in Operations Research and Management Science: Discrete Optimization*, vol. 12. Elsevier, 321–391.

Muckstadt, JA, A. Sapra. 2010. *Principles of Inventory Management: When You are Down to Four, Order More*. Springer.

Plambeck, E., S. Kumar, J. M.l Harrison. 2001. A multiclass queue in heavy traffic with throughput time constraints: Asymptotically optimal dynamic controls. *Queueing Systems* **39**(1) 23–54.

Shmoys, D.B., É. Tardos, K. Aardal. 1997. Approximation algorithms for facility location problems. *Proceedings of the twenty-ninth annual ACM Symposium on Theory of Computing*. ACM, 265–274.

Slotnick, S. A. 2011. Order acceptance and scheduling: a taxonomy and review. *European Journal of Operational Research* **212**(1) 1–11.

Van den Heuvel, W., O.E. Kundakcioglu, J. Geunes, H.E. Romeijn, T.C. Sharkey, A.P.M. Wagelmans. 2012. Integrated market selection and production planning: complexity and solution approaches. *Mathematical programming* **134**(2) 395–424.

Van den Heuvel, W., A.P.M. Wagelmans. 2010. Worst-case analysis for a general class of online lot-sizing heuristics. *Operations research* **58**(1) 59–67.

Wagner, H.M., T.M. Whitin. 1958. Dynamic Version of the Economic Lot Size Model. *Management Science* **5**(1) 89–96.

Wagner, M.R. 2010. Fully distribution-free profit maximization: the inventory management case. *Math. of Oper. Res.* **35**(4) 728–741.

Xu, G., J. Xu. 2009. An improved approximation algorithm for uncapacitated facility location problem with penalties. *Journal of combinatorial optimization* **17**(4) 424–436.

Xu, Y., A. Bisi, M. Dada. 2011. A periodic-review base-stock inventory system with sales rejection. *Operations research* **59**(3) 742–753.

Zeithaml, V. A., R. T. Rust, K. N. Lemon. 2001. The customer pyramid: creating and serving profitable customers. *California Management Review* **43**(4) 118–142.

Appendix. E-companion.

A. Examples

Example 3. Consider the online algorithm where we solve the offline optimal solution with respect to the first k customers, and the decisions for the first $k-1$ customers are fixed, i.e., they cannot be changed from what we have already decided. Then this will result in most customers being rejected if each customer on his own is not enough to warrant production.

For example, consider the To Build or Not To Build problem as explained in Example 1. If the rejection cost of each customer is less than K, then clearly the algorithm we just described will reject everyone in an online fashion. The competitive ratio will be $R(U)/K$ which can be arbitrarily poor. $\square$

Example 4. Consider the ELS problem with online customer selection from Section 5. Let $h = 1, K = 11$, and $r = 10$. Consider the input sequence $I_1 = (1,8), I_2 = (1,14), I_3 = (1,1)$, and $I_4 = (100,1)$, where each tuple represents demand quantity and due date, respectively. Let the set of potential order dates be $\mathcal{Q} = \{1,2,\ldots,15\}$. One can easily show that $\mathcal{A}^* = \{1,3,4\}$, $P(\mathcal{A}^*) = 19$, $\mathcal{R}^* = \{2\}$, and $R(\mathcal{R}^*) = 10$. The Copycat Algorithm will lead to $\mathcal{R}^C = \{1,3\}$ with $R(\mathcal{R}^C) = 20$. StablePair will only have $\mathcal{R}^S = \{1\}$ with $R(\mathcal{R}^S) = 10$. Note that $\mathcal{R}^C \cap \mathcal{R}^* = \mathcal{R}^S \cap \mathcal{R}^* = \emptyset$. Thus,

$$R(\mathcal{R}^S) < P(\mathcal{A}^*) + R(\mathcal{R}^C \cap \mathcal{R}^*) < R(\mathcal{R}^C) < P(\mathcal{A}^*) + R(\mathcal{R}^*)$$

This demonstrates that there are cases in which StablePair can be strictly less conservative than Copycat in that it accepts customers that were rejected by Copycat. $\square$

Example 5. Consider the ELS problem with online customer selection from Section 5. Let $h = 1, K = 11$, and $r = 10$. Let the set of potential order dates be $\mathcal{Q} = \{1,2,\ldots,15\}$. Consider the input sequence $I_1 = (2,4), I_2 = (1,11)$, and $I_3 = (100,1)$, where each tuple represents demand quantity and due date, respectively. One can easily see that $\mathcal{A}_1^* = \{1\}$, $\mathcal{A}_2^* = \{1,2\}$, and $\mathcal{A}_3^* = \{1,3\}$. This implies that $\mathcal{A}_3^C = \{1,2,3\}$. Here we see that Copycat "regrets" accepting customer 2 after stage 3 since the optimal offline solution in stage 3 rejects customer 2. $\square$

Example 6. Consider the ELS problem with online customer selection from Section 5. Let $h = 1, K = 11$, and $r = 10$. Let the set of potential order dates be $\mathcal{Q} = \{1,2,\ldots,10\}$. Consider the input sequence $I_1 = (2,10)$ and $I_2 = (1,3)$, where each tuple represents demand quantity and due date, respectively. Clearly $\mathcal{A}_1^* = \mathcal{A}_2^* = \{1\}$ and the optimal offline cost is $C^*(\{1,2\}) = 11 + 10 = 21$. However, $\mathcal{A}_2^S = \{1,2\}$. StablePair accepts customer 1 with stable pair $(\mathcal{Q}, \{1\})$, where $\mathcal{Q}$ represents all possible time periods to create a setup. However, unlike the optimal offline solution, StablePair also accepts customer 2 with stable pair $(\{3\},\{1,2\})$. If we can only create an order at time 3, then the optimal strategy is to accept both customers 1 and 2. If we can order at any time, then we would only order at time 10 and reject customer 2. $\square$

Example 7 (Polymatroid Optimization). Let U be a ground set and $f(\cdot)$ be a *rank function*, i.e., a nondecreasing submodular function with $f(\emptyset)=0$. Then the polyhedron $F(U,f)=\{\mathbf{x}\in\mathbb{R}^{|U|}: \sum_{i\in T} x_i \le f(T),\ T\subseteq U,\ \mathbf{x}\ge 0\}$ is referred to as a *polymatroid.* Now for each $i\in U$, let $g_i:\mathbb{R}\to\mathbb{R}$ be a concave function. Then for any set $T\subseteq U$ the production cost function is defined as

$$P(T)=\max\sum_{i\in T} g_i(x_i) \text{ s.t. } \mathbf{x}\in F(T,f).$$

He et al. (2012) showed that $P(T)$ is indeed nondecreasing and submodular.

An application of this result, as shown in He et al. (2012), is the *stationary joint replenishment problem.* In this problem the "customers" are actually products. Each product k arrives with $I_k=\{d_k,h_k\}$ where d_k is the stationary continuous demand rate and h_k is the holding cost rate. The rejection cost r_k is the revenue rate generated by producing product k. Demand for accepted products is satisfied by a sequence of orders, where an order for the item types in $T\subseteq U$ has fixed costs $K(T)$. $K(\cdot)$ is assumed to be a rank function. All demands are served by their nearest setup and also incur the appropriate holding costs. In Federgruen and Zheng (1992) it was shown that the optimal policy for the stationary JRP can be approximated well by only considering "power-of-two" (Po2) polices. Indeed, the problem of finding the best Po2 policy then reduces to a polymatroid maximization problem with a separable concave objective. The best of such Po2 policies is at most 2% suboptimal. He et al. (2012) also shows a similar result for the continuous one-warehouse multi-retailer problem. In our online customer selection model, products arrive online to a supplier, who decides whether to accept or reject them, and then computes the best Po2 policy to serve them. $\square$

Example 8. Let $h=1, r=2M$, and $K=2M^2+1$. Let the set of potential order dates be $\mathcal{Q}=\{1,2\}$. Let $S=\{1,2,3\}$ and $I_1=(M,2), I_2=(1,1)$, and $I_3=(100K,2)$, where the tuples represent quantity and due date, respectively. Observe that $\text{OPT}(U_1)$ will reject customer 1 since the setup cost is greater than the rejection cost of $2M^2$. Then $\text{OPT}(U_2)$ will accept both customers 1 and 2, and the total cost will be $2M^2+1+M$, which is less than rejecting both customers at cost $2M^2+2M$. Finally, $\text{OPT}(U_3)$ will clearly accept customers 1 and 3, while rejecting customer 2. The optimal offline cost will be $C^*(U)=P(\{1\cup 3\})+R(\{2\})=2M^2+1+2M$. The cost of Copycat and StablePair will both be $C(U) = P(\{2 \cup 3\}) + R(\{1\}) = 2(2M^2 + 1) + 2M^2 = 6M^2 + 2$. Thus, the competitive ratio is asymptotically close to 3 as M goes to ∞. $\square$

EXAMPLE 9. Let $h = 0$ and $d_k = 1$ for all customers. Let $I_1 = (0, 1, T)$. Clearly a good online algorithm must reject customer 1 or else the competitive ratio would be K/r after the first arrival. Now we repeat the following algorithm to generate a worst case adversary.

0. Initialize $s := 0$, $t := T/2$, $u = T$, and $k = 2$. Let $I_1 = (0, 1, T)$.
1. Let $I_k = (s, 1, t - \epsilon)$.
2. If online algorithm accepts k, then go to Step 3. Else, go to Step 4.
3. Set $s := t$, $t := s + (u - s)/2$, and $u := u$. Go to Step 1.
4. Set $s := s$, $t := s + (u - s)/4$, and $u := s + (u - s)/2$. Go to Step 1.

Let this adversary generate N arrivals and let n be the number of customers that the algorithm accepts. From the construction, it is clear that the algorithm must satisfy each customer with separate orders. Thus, the cost incurred by the algorithm is $nK + (N - n)r$. A good feasible solution to the sequence would be the exact opposite of what the algorithm did. By construction, all the customers that the algorithm rejected (at least one since it must reject customer 1) can be satisfied by one order. Thus the cost of this feasible solution is $K + nr$. The competitive ratio is then at least

$$\frac{nK + (N - n)r}{K + nr} \geq \min\{\frac{Nr}{K}, \frac{K - r}{r}\}.$$

For a given K and r, if we take N to be large enough, then the competitive ratio is at least $K/r - 1$ which is not a constant. One can extend the proof to discrete time by making T large enough.

EXAMPLE 10. Consider a single facility with cost f. Let $r = \sqrt{2}f - \epsilon$. Let the first customer that arrives request 1 unit at the facility, so the service cost would be 0. Then StablePair with scaling factor $\alpha = \frac{1}{\sqrt{2}}$ rejects customer 1. Let the second customer be identical to the first, and so StablePair with scaling accepts customer 2. Clearly the optimal solution is to serve both customers and incur cost f, while StablePair incurs a cost of $(1 + \sqrt{2})f - \epsilon$. Therefore, the competitive ratio can be arbitrarily close to $1 + \sqrt{2}$ as ϵ goes to 0. $\square$

B. Proofs

Proof of Lemma 3. Consider a customer $k \in \mathcal{A}^C$ and let $\mathcal{A}_k^*$ be the optimal solution induced by OPT$(\mathcal{Q}, U_k)$. Since k was accepted by the Copycat Algorithm, then $k \in \mathcal{A}_k^*$. This implies that $(\mathcal{Q}, \mathcal{A}_k^*)$ is a stable pair for k since there is a solution to OPT$(\mathcal{Q}, \mathcal{A}_k^*)$ that accepts all of $\mathcal{A}_k^*$. Therefore $k \in \mathcal{A}^S$ and $\mathcal{A}^C \subseteq \mathcal{A}^S$. $\square$

Proof of Theorem 5. Consider the To Build or Not to Build problem with online customer selection, as explained in Example 1. Let $K = 1$ and assume that the rejection cost for each customers k is $r_k = \epsilon$ for some $\epsilon > 0$. It can be seen that this is a special case of a submodular, ELS, JR, or FL problem with online customer selection. Any nontrivial deterministic algorithm for this problem simply specifies how many arrivals have to occur before it starts accepting customers. We denote ALG_q as the algorithm which rejects the first $q-1$ customers and then accepts all remaining customers starting with customer q. Thus, all algorithms can be parameterized by $q \in \mathbb{Z}_+ \cup \{\infty\}$. Now focus on the worst-case adversary strategy given a fixed q.

Case 1 If $q > 2/\epsilon$, then the adversary generates $2/\epsilon$ arrivals. Thus ALG_q will reject all q customers and incur a total cost of 2, while the optimal offline cost is clearly 1.

Case 2 If $1/\epsilon < q \le 2/\epsilon$, then the adversary generates exactly q arrivals. Thus ALG_q incurs a cost of $1 + (q-1)\epsilon$, while the optimal offline cost is just 1.

Case 3 If $q \le 1/\epsilon$, then the adversary generates exactly q arrivals. Thus ALG_q incurs a cost of $1 + (q-1)\epsilon$, while the optimal offline cost is $q\epsilon \le 1$.

The competitive ratio in all three cases is at least 2 or arbitrarily close to 2 as ϵ approaches 0. $\square$

Proof of Lemma 6. 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 $\mathrm{OPT}(Q, T)$ is better off rejecting k and keeping everything else the same, which is a contradiction to the stability of (Q, T). $\square$

Proof of Lemma 7. Assume that $[a, b]$ does not contain the due date of any customers in $\mathcal{A}^*$. 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 $\mathcal{A}^*$ could be augmented to include all customers having due dates in $[a, b]$ without increasing the overall cost. Since we chose $\mathcal{A}^*$ to be maximal, then this is a contradiction. $\square$

Proof of Theorem 7. We prove below that the production cost incurred by StablePair (Copycat) Algorithm is at most $P(\mathcal{A}) < 3P(\mathcal{A}^*) + 3R(\mathcal{A} \cap \mathcal{R}^*) <= 3C^*(U)$. Combining Theorems 1 and 2 with this bound yields the desired result. Note that Lemmas 6 and 7 still apply in this setting with soft capacities.

Similar to the analysis of the ELS case in Lemma 8, let $s_1 < s_2 < . < s_m$ denote the times where orders were placed in the ELS with soft capacities solution to $P(A^*)$. Based on the analysis of Lemma 8, we know that all customers in $\mathcal{A}$ have due dates within $\cup_{j=1}^m [s_j - r/h, s_j + 2r/h]$. We now focus on the customers in $\mathcal{A}$ with due dates in one particular interval, $[s_j - r/h, s_j + 2r/h]$, which we denote by $\mathcal{A}^j$. To serve the customers in $\mathcal{A}^j \cap \mathcal{A}^*$, we simply serve them from s_j using the same number of orders at s_j that were in the solution to $P(\mathcal{A}^*)$. Summing over all j, this implies that $P(\mathcal{A} \cap \mathcal{A}^*) \leq K^* + H^*$.

Let $\mathcal{A}^{j'}$ be the customers in $\mathcal{A}^j \cap \mathcal{R}^*$ with due dates in $[s_j - r/h, s_j)$. Now let $Lc + l$, with L being an integer, be the total demand for customers in $\mathcal{A}^{j'}$. We serve all the customers in $\mathcal{A}^{j'}$ from $L+1$ orders at $s_j - r/h$. This will result in a holding cost of at most $R(\mathcal{A}^{j'} \cap \mathcal{R}^*)$. We will pay for one order using the order at s_j. Finally, we will pay for the other L orders using $R(\mathcal{A}^{j'} \cap \mathcal{R}^*)$. This is due to the fact that, without loss of generality, $rc \geq K$. (If this were not the case, then no orders would ever be placed.) Now let $\mathcal{A}^{j''}$ be the customers in $\mathcal{A}^j \cap \mathcal{R}^*$ with due dates in $[s_j - r/h, s_j)$. Using a similar analysis, we can serve all the customers in $\mathcal{A}^{j''} \cap \mathcal{R}^*$ from additional orders at s_j. The additional orders will have a cost at most $K + R(\mathcal{A}^{j''} \cap \mathcal{R}^*)$, and the holding costs will be at most $2R(\mathcal{A}^{j''} \cap \mathcal{R}^*)$. Therefore, the total cost of serving $\mathcal{A}^j \cap \mathcal{R}^*$ is at most $2K + 3R(\mathcal{A}^j \cap \mathcal{R}^*)$. Summing over all j, $P(\mathcal{A} \cap \mathcal{R}^*) \leq 2K^* + 3R(\mathcal{A} \cap \mathcal{R}^*)$. Combining this with the previous bound we obtain that $P(\mathcal{A}) < 3P(\mathcal{A}^*) + 3R(\mathcal{A} \cap \mathcal{R}^*)$. $\square$

Proof of Theorem 8. We will show that $\mathcal{A}^C \subseteq \mathcal{A}^*$ and thus $P(\mathcal{A}^C) \leq P(\mathcal{A}^*)$. Since we will be using the 2-competitive heuristic of Axsater (1982) for the online ELS problem, then the production costs will be at most $2P(\mathcal{A}^C) \leq 2P(\mathcal{A}^*)$. From Theorem 3, this gives a 3-competitive algorithm.

In fact, we will show that there is a sequence of solutions such that $\mathcal{A}_1^* \subseteq \ldots \subseteq \mathcal{A}_N^*$ which implies $\mathcal{A}^C \subseteq \mathcal{A}^*$. Consider a customer k in $\mathcal{A}_k^*$. Let $k' \geq k$ be the first k' such that $k \notin \mathcal{A}_{k'}^*$. Then the our production horizon can be partitioned into $T_1 = [t_1, t_k]$ and $T_2 = (t_k, t_k']$. No customers with due dates in T_2 are served by an order in T_1 since customer k was rejected and the rejection cost per unit is the same for all customers. Therefore, the offline optimal solution can be decoupled to solving the respective problems for the customers with due dates in T_1 and T_2 and then merging the solutions together. Since the due dates of the customers are chronological, then the customers corresponding to T_1 are exactly U_k. Therefore, we know there is an optimal offline solution for U_k that accepts k, and thus we can use that to find an optimal solution to OPT$(U_{k'})$ that accepts k. $\square$

Proof of Lemma 9. Let (Q_k, T_k) be the stable pair that accepted customer k. Consider the order that serves customer k in the ELS production plan for $P(Q_k, T_k)$, and call this order date t. Define $Q = \{t\}$ and T' to be all the customers that are served by the order at time t in $P(Q_k, T_k)$. Clearly $(Q, T') \subseteq (Q_k, T_k)$ and $k \in T'$. The first property holds by construction. Note that (Q, T') is a stable pair since (Q_k, T_k) is stable. The stability property from (1) directly implies that $R(T') \geq K + \sum_{j \in T'} h(t_j - t)d_j$. Now let $T = \{j \in U_k | t_j \in [t, t + r/h]\}$, which is exactly the second property.

Lemma 6 implies that $T' \subseteq T$. It follows that for all $j \in T$, $rd_j \geq h(t_j - t)d_j$. Therefore, the rejection costs of T are also greater than the production costs $P(Q,T)$, i.e. $R(T) \geq K + \sum_{j \in T} h(t_j - t)d_j$, which is the third property. Properties 1, 2, and 3 are sufficient conditions for (Q,T) to be a stable pair. Thus (Q,T) is a stable pair with $k \in T$ that satisfies all three properties. The other direction is trivial. $\square$

Proof of Theorem 10. The proof is very similar to the proof of Lemma 10, but requires a more involved construction and analysis. Let $s_1, \ldots, s_m$ denote the times of the orders in the optimal production plan for $P(\mathcal{A}^*)$. The cost of the optimal production plan can be decomposed into the total setup costs, K^*, and the total holding costs, H^*. Let $\mathcal{A}^i$ simply denote the customers with item type i that are in $\mathcal{A}$. For each customer $k \in \mathcal{A}^i$, let a_k be the order date that serves customer k in the stable pair solution that StablePair used to accept customer k, and let $\hat{T}^i$ denote the set of these order dates sorted from earliest to latest. Construct $T^i \subseteq \hat{T}^i$ by processing $\hat{T}^i$ in order from earliest to latest, and remove any order date that is within r^i/h^i periods of the previous order date that was not removed. Now consider the following two sets, X^i and Y^i. The set X^i is defined to be the set of all type i customers that have due date within $[t, t + r^i/h^i]$ for some $t \in T^i$. Note that X^i is not necessarily contained in the set $\mathcal{A}^i$. The set Y^i is then defined to be $\mathcal{A}^i \backslash X^i$. For convenience, let us define $\Delta = \max_i r^i/h^i$ and $\delta = \min_i r^i/h^i$.

To construct a solution that serves all the customers in $\mathcal{A}$, we first create the same sequence of orders as in $P(\mathcal{A}^*)$ in periods $s_1, \ldots, s_m$ and incur setups costs of K^*. Note that the item orders are replicated as well. All the customers in $\mathcal{A} \cap \mathcal{A}^*$ are then served in the same way as in the production plan for $P(\mathcal{A}^*)$, and thereby incur holding costs of at most H^*. Now we create a sequence of duplicate orders shifted back by time Δ, i.e. at periods $s_1 - \Delta, \ldots, s_m - \Delta$, and incur another K^*.

Next, consider each item type i separately. Assume for now that for each order date $t \in T^i$, there exists an order s_j in the production plan for $P(\mathcal{A}^*)$, such that either $s_j \in [t - \Delta, t]$ or $s_j - \Delta \in [t - \Delta, t]$. Moreover, order s_j (or $s_j - \Delta$) includes item i. Assuming this property holds, one can bound the holding costs for the customers in $\mathcal{A}^i \cap \mathcal{R}^*$. Specifically, the property ensures that all type i customers with due dates in $[t, t + r^i/h^i]$ can be served with holding cost at most $(\Delta + r^i/h^i)h^i$ per unit. By definition of X^i, this means that the customers in $X^i \cap \mathcal{A}^i \cap \mathcal{R}^*$ will be served with total holding costs at most $(\Delta/\delta + 1)R(X^i \cap \mathcal{A}^i \cap \mathcal{R}^*)$. The remaining customers left to be served are those in $Y^i \cap \mathcal{R}^*$. Consider customer $k \in Y^i \cap \mathcal{R}^*$. By construction of T^i, it follows that there exists a $t \in T^i$ such that $t \leq a_k \leq t + r^i/h^i \leq a_k + r^i/h^i$. In addition, from the definition of a_k and Lemma 6, it follows that $t_k \in [a_k, a_k + r^i/h^i]$. By the property assumed above, there exists an order that includes item type i within Δ before t. The total holding cost from that order to t, t to a_k, and a_k to t_k is at most $(\Delta + 2r^i/h^i)h^i$ per unit.

It is now sufficient to ensure that indeed for each $t \in T^i$, there exists an order of type i within Δ time periods earlier than t. To achieve this, extra item orders will be added to the construction. From Lemma 7, it then follows that the setup interval corresponding to t intersected some optimal setup interval starting at s_j. (If there is a choice of intersections, choose s_j that contains an item order of type i if one exists.) From Lemma 6, it follows that $s_j - \Delta \le t \le s_j + \Delta$. We now consider two cases and show how to enforce the property in each case.

Case 1: There is a type i customer in $\mathcal{A}^*$ with due date in $[t, t + r^i/h^i]$**.** By construction, this implies that there are type i orders at s_j and $s_j - \Delta$, respectively.

Case 2: There is no type i customer in $\mathcal{A}^*$ with due date in $[t, t + r^i/h^i]$**.** If $s_j - \Delta \le t < s_j$, then we place an *extra* item order of type i at the joint order located at time $s_j - \Delta$. Otherwise, $s_j \le t \le s_j + \Delta$ and we place the extra order of type i at s_j. Since t corresponds to a setup interval from a stable pair solution, it follows from (1) that there exists a set of customers with due dates in $[t, t + r^i/h^i]$ whose rejection costs are greater than K_i. Under the case assumption, the type i demands with due dates in $[t, t + r^i/h^i]$ are all in $\mathcal{R}^*$. Furthermore, all customers with due dates in $[t, t + r^i/h^i]$ are in X^i. Thus, the extra item orders have cost at most $R(X^i \cap \mathcal{R}^*)$. Each customer in $X^i \cap \mathcal{R}^*$ can be used at most once to pay for an extra item order by the spacing of the times we enforced in the construction of T^i.

The total cost incurred by the construction is $K^* + H^* + K^* + \sum_{i=1}^{M}(R(X^i \cap \mathcal{R}^*) + (\Delta/\delta + 1)R(X^i \cap \mathcal{A}^i \cap \mathcal{R}^*) + (\Delta/\delta + 2)R(Y^i \cap \mathcal{R}^*)) \le 2P(\mathcal{A}^*) + R(\mathcal{R}^*) + (\Delta/\delta + 1)R(\mathcal{A} \cap \mathcal{R}^*)$. Combining Theorems 1 and 2 with this bound completes the proof. □

Proof of Lemma 11. The proof is analogous to the proof of Lemma 9. Let (Q_k, T_k) be the stable pair that the StablePair Algorithm used to accept customer k. Let t be the order that serves customer k in the JR solution of $P(Q_k, T_k)$. Define $Q = \{t\}$, T' to be all the customers served by the order at t, and $\mathcal{I}'$ to be the set of items ordered at t in the solution of $P(Q_k, T_k)$. Note that (Q, T') must be stable pair with $k \in T'$. Define T^i, $\mathcal{I}$, and T according to Properties 1, 2, and 3. Note that $T' \subseteq T$ by Lemma 6 and Property 2. Since $R(T') \ge K_0 + \sum_{i \in \mathcal{I}'} K_i + \sum_{j \in T'} h(t_j - t)d_j$ by (1), then it is easy to see that $R(T) \ge K_0 + \sum_{i \in \mathcal{I}} K_i + \sum_{j \in T} h(t_j - t)d_j$, satisfying Property 4. Therefore, (Q, T) is a stable pair with $k \in T$ that satisfies all four properties. The other direction is trivial. □

Proof of Lemma 12. If there is a customer paying more than r, then the solution $\mathrm{OPT}(Q, T)$ can be improved by rejecting that customer and keeping everything else the same. This would contradict the stability of (Q, T). □

Proof of Lemma 13. Assume that $S_k \cap \mathcal{A}^* = \emptyset$. This means that the solution to OPT(U) rejected the entire set of customers in S_k. However, by stability of (Q_k, T_k) and (1) it follows that $P(\{q_k\}, S_k) \leq R(S_k)$. Therefore, adding the customers in S_k to $\mathcal{A}^*$ will not increase the total cost. This is a contradiction since we chose $\mathcal{A}^*$ to be a maximal solution. $\square$

Proof of Lemma 15. As in Lemma 4, we can show that $\alpha R(\mathcal{R}_\alpha \cap \mathcal{A}^*) \leq P(\mathcal{R}_\alpha \cap \mathcal{A}^*)$. Adding $\alpha R(\mathcal{R}_\alpha \cap \mathcal{R}^*)$ and dividing by α completes the proof. $\square$

Proof of Lemma 16. We use the same notation defined in Section 7.1. Lemma 12 now holds for αr. Lemma 13 still holds because the rejection costs are scaled down, so (q_k, S_k) must be a stable pair under the original rejection costs as well.

Similar to Lemma 14, we will construct a solution with bounded costs. Specifically, open all of the facilities in the production plan for $P(\mathcal{A}^*)$ and serve each customer in $\mathcal{A}_\alpha$ by the nearest facility to its location. Consider customer $k \in \mathcal{A}_\alpha \cap \mathcal{R}^*$. By the stability of (q_k, S_k) under the scaling and Lemma 12, it follows that $c(k, q_k)$ and $c(q_k, l)$ are both at most αr. It also follows that $c(l, q_l^*) \leq r$ by stability of the optimal solution and using Lemma 12 with the unscaled costs. Thus, $c(k, q_l^*) \leq (2\alpha + 1)r$, which completes the proof. $\square$

Proof of Lemma 17. Assume k was accepted by StablePair, and let j be the facility that served k in the corresponding production plan. Now define Q and T as described in the lemma. The rest of the proof is almost identical to the proof of Lemma 9 and thus we omit it. $\square$

C. Figures and Tables

Conservative Scenario

	Maximum performance ratio			Final performance ratio		
r	Copycat	StablePair	StablePair(2)	Copycat	StablePair	StablePair(2)
1	1.00	1.00	1.00	1.00	1.00	1.00
5	1.51	1.47	1.85	1.44	1.41	1.11
10	1.53	1.49	2.29	1.29	1.25	1.06

Table 1 For each value of r, we do 100 experiments of the 'Conservative' scenario, each with $N = 500$ customer arrivals ($K = 100$ and $h = 1$). We report the maximum performance ratio observed over all experiments and customer arrivals. We also report the average final performance ratio after the final customer has arrived. StablePair(2) denotes the StablePair Algorithm with a scaling factor of 2.

More Demands Scenario

	Maximum performance ratio			Final performance ratio		
r	Copycat	StablePair	StablePair(2)	Copycat	StablePair	StablePair(2)
1	1.37	1.37	1.76	1.27	1.27	1.23
5	1.54	1.54	2.40	1.23	1.19	1.03
10	1.90	1.90	2.33	1.09	1.06	1.01

Table 2 For each value of r, we do 100 experiments of the 'More Demands' scenario, each with $N = 500$ customer arrivals ($K = 100$ and $h = 1$). We report the maximum performance ratio observed over all experiments and customer arrivals. We also report the average final performance ratio after the final customer has arrived. StablePair(2) denotes the StablePair Algorithm with a scaling factor of 2.

Large Orders First Scenario

	Maximum performance ratio			Final performance ratio		
r	Copycat	StablePair	StablePair(2)	Copycat	StablePair	StablePair(2)
1	1.29	1.29	1.56	1.20	1.20	1.15
5	1.26	1.24	1.22	1.11	1.07	1.00
10	1.09	1.05	1.05	1.01	1.00	1.00

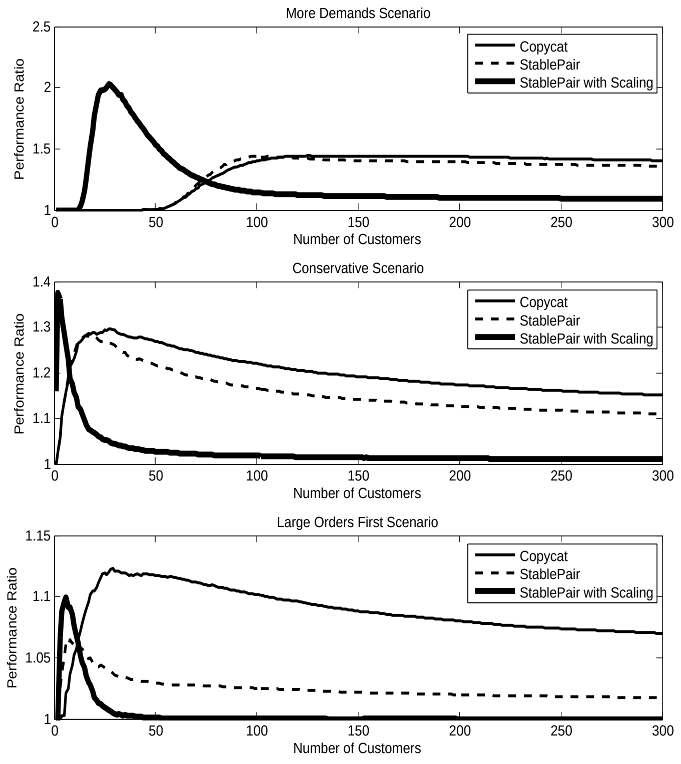
Table 3 For each value of r, we do 100 experiments of the 'Large Orders First' scenario, each with $N = 500$ customer arrivals ($K = 100$ and $h = 1$). We report the maximum performance ratio observed over all experiments and customer arrivals. We also report the average final performance ratio after the final customer has arrived. StablePair(2) denotes the StablePair Algorithm with a scaling factor of 2.

JR Problem Simulation Results

	Maximum performance ratio			Final performance ratio		
Scenario	Copycat	StablePair	StablePair(2)	Copycat	StablePair	StablePair(2)
Conservative	1.54	1.57	2.71	1.40	1.36	1.09
More Demands	1.75	1.75	3.00	1.15	1.11	1.01
Large Orders First	1.34	1.44	1.55	1.07	1.02	1.00

Table 4 We simulate the three scenarios over 100 experiments, with $N = 300$, $M = 3$, $K_0 = 100$, $K_1 = K_2 = K_3 = 20$, $r = 10$, and $h = 1$. We report the maximum performance ratio observed over all experiments and customer arrivals. We also report the average final performance ratio after the final customer has arrived. StablePair(2) denotes the StablePair Algorithm with a scaling factor of 2.

Figure 6 Experimental Results.



Note. Three experiments detailing the actual performance ratio of three algorithms on the JR problem with online customer selection.