Manufacturing & Service Operations Management

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

Postponable Acceptance and Assignment: A Stochastic Dynamic Programming Approach

Keumseok Kang, J. George Shanthikumar, Kemal Altinkemer

To cite this article:
Keumseok Kang, J. George Shanthikumar, Kemal Altinkemer (2016) Postponable Acceptance and Assignment: A Stochastic Dynamic Programming Approach. Manufacturing & Service Operations Management 18(4):493-508. http://dx.doi.org/10.1287/msom.2016.0581

M&SOM MANUFACTURING & SERVICE OPERATIONS MANAGEMENT
Vol. 18, No. 4, Fall 2016, pp. 493–508
ISSN 1523-4614 (print) | ISSN 1526-5498 (online)

informs

http://dx.doi.org/10.1287/msom.2016.0581

Postponable Acceptance and Assignment: A Stochastic Dynamic Programming Approach

Keumseok Kang
College of Business, Florida International University, Miami, Florida 33174, kskang@fiu.edu

J. George Shanthikumar, Kemal Altinkemer
Krannert School of Management, Purdue University, West Lafayette, Indiana 47907
{shanthikumar@purdue.edu, kemal@purdue.edu}

We study a new dynamic order acceptance and assignment problem of a make-to-order manufacturing or service system that produces heterogeneous products or services using heterogeneous servers. Unlike traditional dynamic order acceptance and assignment problems, in our settings the system can strategically postpone acceptance and assignment decisions for orders in hand. The system does this while waiting for more profitable orders to come and/or more productive servers to become available, with the risk of losing orders in hand, low server utilization, and high waiting penalties. We formulate this problem as a stochastic dynamic program, characterize the structure of the optimal policy, and discuss managerial insights gained from the optimal policy. The paper also provides a methodological contribution by finding general structural properties and their preservation conditions, which can be reused in related future operations management studies.

Keywords: postponable acceptance and assignment; structure of the optimal policy; stochastic dynamic program
History: Received: January 28, 2013; accepted: December 7, 2015. Published online in *Articles in Advance* August 18, 2016.

1. Introduction

We study a new problem category referred to as the *problem of manufacturing or service system with postponable acceptance and assignment in make-to-order settings* (hereafter MSSPAA). In MSSPAA, the system produces heterogeneous products or services using heterogeneous but cross-trained servers and tries to maximize its profit by accepting or rejecting orders from customers and assigning the accepted orders to available servers. One unique feature of MSSPAA is that the system can strategically postpone the acceptance and assignment decisions for some orders, while waiting for more profitable orders (defined as orders with larger revenue) to come and/or for more productive servers (defined as servers with higher service rates) to become available. Unlike traditional acceptance and assignment problems, such as the secretary, online-knapsack, and general queuing problems, in MSSPAA the acceptance decision for an order does not have to be made upon the arrival of the order. Similarly, the assignment decision does not have to be made upon acceptance. Instead, the acceptance decision can be deferred until the order is withdrawn by the customer, and the assignment decision can be delayed until an alternative server becomes available. There is a trade-off in MSSPAA. The system may improve the profit by waiting for more profitable orders and more productive servers to be available. However, it may earn less profit induced by losing orders in hand, low server utilization, and high waiting penalties. This is due to uncertainties on the arrivals of new orders, abandonment of orders in hand, and availability of servers in the future. We formulate MSSPAA as a stochastic dynamic program, characterize the structure of the optimal policy, and provide novel managerial insights obtained from the optimal policy.

This study is motivated from the order acceptance and resource assignment problem faced by an IT training/consulting company. The company sends its consultants with different expertise to client sites to offer different types of professional services such as mobile and web apps development training or consulting. After receiving a client's order to request a service, the company must decide to accept or reject the order and assign its resource if the order is accepted. Although clients prefer a prompt answer from the company on whether the company will provide the service, the clients are also willing to wait and negotiate for the starting date of the service in this industry (Akçay et al. 2010). The company may not make acceptance and assignment decisions for an order upon its arrival, but it may postpone making decisions while waiting for alternative orders and resources to become available. However, by delaying decisions, the company also takes the risk of losing the order and low utilization of its resources.

Using an example from the IT training/consulting company, we exemplify the optimal policy of MSSPAA and highlight the main insights provided by the optimal policy. The company has consultants for

mobile apps development (called *mobile consultants*). The mobile consultants provide on-site mobile apps training/consulting services to customers. The services include training, mentoring, troubleshooting, technical support, architecture design, and development. Suppose the company has received new orders for mobile apps training or consulting (called *mobile orders*) and needs to make the acceptance decision for them. If the company currently has available mobile consultants, the company must accept and assign the orders to these available consultants without any postponement. Even if there is no available consultant, the company still wants to accept and keep some orders in the queue (called *assignment queue*). They will be assigned later when consultants become available after completing their current jobs.

To better illustrate the optimal acceptance policy when the acceptance decision is postponable, first we show the optimal policy when the acceptance decision is not postponable. This means the company must accept or reject an order upon its arrival. In this case, the optimal acceptance policy forms a simple threshold policy where the threshold represents the optimal number of orders to keep in the assignment queue. Suppose the threshold is found to be 18. It will accept and keep up to 18 mobile orders in the assignment queue and will reject any surplus order after accumulating 18 orders.

Consider next the optimal policy when the acceptance decision is postponable. Now the company does not have to accept or reject an order upon its arrival. It may postpone its acceptance decision to some extent while the customer may withdraw the order maybe by finding another service provider. The company may keep some orders in the assignment queue, but it also may postpone the acceptance decisions of some orders and keep them in the queue for acceptance (called *acceptance queue*). Suppose that the optimal number of orders for the company to keep in the assignment and acceptance queues is 13 and 14, respectively. It is intuitive that the optimal number for the assignment queue (13) is smaller than 18 but the total number for both queues ($27 = 13 + 14$) is larger than 18.

Unlike the case if acceptance is not postponed, the optimal acceptance policy becomes more complicated. It is not a threshold type any more but a switching curve type consisting of several areas where the optimal decision varies by area. Suppose the company has fewer than 27 orders in both queues, including newly arrived orders. In this case, the company cannot have 13 orders in the assignment queue and 14 orders in the acceptance queue. The best acceptance decision for the company to make would be to have more orders in the assignment queue to increase the availability of order in subsequent periods. For example, if the company has only 21 available orders, the optimal number of orders to keep in the assignment queue is 15 and that in the acceptance queue is 6. The fewer orders in the queues, the more orders should be in the assignment queue. On the other hand, suppose the company already has more than 13 orders in the assignment queue. Since an accepted order cannot be rejected, the best decision for the company to make is to have fewer orders in the acceptance queue. For example, if the company already has 17 orders in the assignment queue, the optimal number of orders in the acceptance queue is 5. The more orders in the assignment queue, the fewer orders should be in the acceptance queue. One of the notable findings is that the company has to maintain the number of total orders in both queues to less than 27 ($= 13 + 14$) in any case (see Figures 4(c) and 4(d) for details of the optimal policy for this case).

Suppose the company also provides on-site consulting/training services for web apps development. The company has consultants dedicated for web development, but mobile consultants may be assigned to training or consulting orders for web apps development (called *web orders*) if it is necessary. However, the company prefers to assign mobile consultants to mobile orders since they are more efficient. Therefore, if there are no available mobile orders but only web orders, the company may postpone assigning a mobile consultant to a web order. The optimal number of mobile consultants to reserve for potential mobile orders is determined by the arrival rate of the mobile orders. However, it is also affected by the number of web orders in the assignment queue. The more web orders in the assignment queue, the fewer mobile consultants are reserved.

Although the paper is motivated from a particular problem in the IT training/consulting industry, MSSPAA is a fundamental dynamic order acceptance and resource assignment problem that can be applied to solve a subset of the complex real-world problems where acceptance and assignment decisions are separated and postponed. Related practical examples include make-to-order factories that accept and assign customers' orders to manufacturing facilities, healthcare service providers that accept and assign patients to healthcare resources, law companies that accept and assign clients' cases to lawyers, and field service companies that accept and assign customers' repair requests to field engineers.

The main contribution of this study is to introduce a new problem category, formulate the model, characterize the structure of the optimal policy, and provide new managerial insights. Similar concepts to assignment postponement have been considered in the inventory/resource rationing and the call center routing problems. To the best of our knowledge, the concept of acceptance postponement has not been

studied, especially when acceptance and assignment decisions are separated. In addition to the main contribution mentioned above, as a by-product of this study we provide a significant methodological contribution. We derive novel structural properties when a two-dimensional function has concavity, submodularity, and the *directional property* that we define as having subconcavity in one element and superconvexity in another element. We also find a condition where these new properties are preserved under profit-to-go expectation and a new condition where submodularity is preserved under maximization. Most known structural results of two-dimensional functions are based on diagonal dominance that means superconvexity in both elements (Ha 1997, Yang and Qin 2007, Zhao et al. 2008). Thus, we believe that these new structural properties and preservation conditions nicely complement the existing knowledge and may have great application in future operations management studies.

The paper continues in Section 2, discussing the uniqueness of this study with respect to prior literature. We formulate MSSPAA as a stochastic dynamic program in Section 3 and find the structural properties of the objective function, using concavity, submodularity, and the directional property in Section 4. We characterize the structure of the optimal policy of MSSPAA and discuss the managerial implications of the optimal policy in Section 5. In Section 6 we conduct a numerical study to show the benefits of acceptance postponement. The summary of our main results and the discussion of contributions and limitations are in Section 7.

2. Related Literature

To the best of our knowledge, MSSPAA is unique and has not previously been studied in the literature. Although similar concepts of assignment postponement have been considered in the literature, the concept of acceptance postponement has yet to be studied in the literature.

In the call center routing problem, customer calls are stored in queues and served by agents according to certain assignment rules. Some prior studies assume that heterogeneous customer calls are served by heterogeneous agents and customer calls may be abandoned before they are served (Bassamboo et al. 2005). However, MSSPAA is different from the call center routing problem. In call centers, acceptance and assignment decisions are made at the same time, while in MSSPAA both decisions need not be made at the same time. In call centers, calls are accepted and assigned only when there are available agents, and every accepted call is immediately served. However, in MSSPAA, high-priority orders may be accepted even when no server is available to secure them.

Similar problems have been studied in inventory management settings. Gupta and Wang (2007) study the acceptance and delivery problem for two customer types. Although they consider separate acceptance and assignment decisions in their extended make-to-stock model, they do not consider postponed decisions in both acceptance and assignment and do not assume order abandonment. Wang and Yan (2009) and Duran et al. (2008) consider one-cycle delayed delivery as one of the delivery options. However, it is not a postponed decision since the delivery decision is made together with the acceptance decision immediately upon the arrival of an order. Wang et al. (2014) examine a similar but extended model where some delivery decisions can be made separately from the acceptance decision. However, they do not consider acceptance postponement and order abandonment. The inventory (or resource) rationing problem reserves inventory (or resource) for high-priority orders while rejecting or postponing the fulfillment of low-priority orders (Deshpande et al. 2003, Gao et al. 2012). This is analogous to postponing order assignment in MSSPAA. Most inventory rationing studies do not consider the separate order acceptance process from assignment (e.g., Deshpande et al. 2003). A few others do consider it (e.g., Gao et al. 2012). None of them considers acceptance postponement and order abandonment.

Another similar problem is the stochastic (or online) knapsack problem (Kleywegt and Papastavrou 1998, Ross and Tsang 1989, Van Slyke and Young 2000). The discussion of Ross and Tsang (1989) seems to be most similar to MSSPAA. Unlike most stochastic knapsack problems that solve the one-time assignment problem where resources are assigned only once during a decision horizon, Ross and Tsang (1989) incorporate a random order completion process into their model. They assume that resources can be reassigned more than once as long as they complete assigned jobs like MSSPAA. However, Ross and Tsang (1989) do not consider postponable acceptance in their model. MSSPAA is also similar to the multiple secretary (or k-secretary) problems. The secretary problem is a special case of the online knapsack problem of which resource requirements of every upcoming order are identical (Ferguson 1989). Like the knapsack problem, in most secretary problems, acceptance decisions are assumed to be made upon the arrival of candidates, which is different from MSSPAA. Also, like most knapsack problems, traditional secretary problems deal with only one-time acceptance.

The dynamic order assignment problem with heterogeneous orders and servers is closely related to MSSPAA. Some propose simple heuristics such as the $c\mu$ rules (Van Mieghem 1995). Others provide the structural properties of the optimal policy (e.g., Ahn et al. 2005, Akçay et al. 2010, Xu et al. 1992). For example, Ahn et al. (2005) and Xu et al. (1992) provide some structural properties of the optimal

policy in the context of multiple orders and servers, and Akçay et al. (2010) study postponement in the one-time assignment decision. None of them has considered acceptance postponement.

Concavity (or convexity), submodularity (or supermodularity), superconvexity (or subconvexity), superconcavity (or subconcavity), and diagonal dominance are general mathematical properties that have been used to characterize the two-dimensional optimal policy in many operations management studies (e.g., Morton 2006, Zhuang and Li 2012, Li and Yu 2014). Zhuang and Li (2012) show generalizable structural results when a two-dimensional function $f(x_1, x_2)$ is supermodular and superconvex (or submodular and subconvex) in x_1 and x_2. Similarly, Morton (2006) examines the cases when a function is submodular and subconcave (or supermodular and superconvex) in its all elements. Some studies derive reusable properties of two-dimensional function $f(x_1, x_2)$ when $f(x_1, x_2)$ is diagonal dominant (i.e., superconvex), concave, and submodular (Ha 1997, Yang and Qin 2007, Zhao et al. 2008). Although the above studies provide important reusable results that can be applied to many other operations management problems, $f(x_1, x_2)$ may have a different combination of properties. This leads to different structural properties of $f(x_1, x_2)$. In this study we derive some novel structural properties for the case that $f(x_1, x_2)$, which is concave and submodular, has subconcavity in x_1 but superconvexity in x_2 (referred to as the *directional property*), and we apply these new properties to characterize the structure of the optimal policy for MSSPAA. To the best of our knowledge, there has not been any study investigating the combined properties of a two-dimensional objective function, which is concave, submodular, superconvex in one element, and subconcave in another element, in a stochastic dynamic maximization program setting. We believe the new properties we find have great potential to be applicable to related future operations management studies.

3. Problem Definition and Formulation

We define MSSPAA and formulate it as a discrete-time stochastic dynamic program. Since MSSPAA assumes postponable decisions for incoming orders do not require real-time response, a discrete-time model can be appropriately used for tractability and convenience (Akçay et al. 2010, Talluri and van Ryzin 2005). Consider a system where different types of orders arrive at discrete times $\{1, 2, \ldots, n\}$. Let period k be defined as the time interval $[k, k+1)$, $k = 1, 2, \ldots, n$. One period can be conceptually interpreted as a maximal time interval during which any order can be postponed without abandonment or any waiting penalty. There are I order types ($i = 1, 2, \ldots, I$), and each order belongs to one order type. There are J server types ($j = 1, 2, \ldots, J$), and each type has a predefined number of servers, $S^j > 0$. All servers are cross-trained so that they can process any type of order but with different service rates. Let Z_k^i be the total number of type-i orders that newly arrive at the end of period k. Z_k^i has mean $\bar{Z}^i$ and is independent over k and i. Let $x_{2,k}^i$ be the total number of type-i orders, including Z_{k-1}^i, in the acceptance queue at time k. Let $x_{1,k}^i$ be the number of type-i orders which have been accepted and kept in the assignment queue at time k.

At the beginning of each period, orders in the acceptance queue may be accepted or rejected (called *acceptance decision*), or their acceptance decisions may be postponed so that they stay in the acceptance queue (called *acceptance postponement*). Let a_k^i and r_k^i be the number of orders being accepted and rejected, respectively, at the beginning of period k. The number of type-i orders in the acceptance queue right after acceptance decisions becomes $x_{2,k}^i - a_k^i - r_k^i$. Similarly, at the beginning of each period, orders in the assignment queue may be assigned to servers (called *assignment decision*). Some orders in the acceptance queue are accepted and assigned to servers immediately (called *accept and immediate assign*), while some others are accepted and kept in the assignment queue (called just *accept*). Let u_k^{ij} be the number of type-i orders to be newly assigned to type-j servers, including orders being accepted and immediately assigned, at the beginning of period k. The number of type-i orders in the assignment queue right after acceptance and assignment decisions becomes $x_{1,k}^i + a_k^i - \sum_{j=1}^J u_k^{ij}$. In some cases, some orders may not be assigned even if there are available servers (called *assignment postponement*).

We assume that customers who place orders are impatient; therefore, some orders in the acceptance queue may be abandoned (called *order abandonment*). However, acceptance means a committed contract. Therefore, if an order is accepted, the order cannot be abandoned or rejected but has to be served. We use a multiplicative random variable W_k^i ($0 \le W_k^i \le 1$), which has mean $\bar{W}^i$ and is independent over k and i, to implement the order abandonment process. Only W_k^i fraction of type-i orders, waiting in the acceptance queue at the beginning of period k, remain in the same queue at the beginning of period $k+1$. The others are abandoned during period k:

$$x_{1,k+1}^i = x_{1,k}^i + a_k^i - \sum_{j=1}^J u_k^{ij};$$

$$x_{2,k+1}^i = W_k^i(x_{2,k}^i - a_k^i - r_k^i) + Z_k^i.$$

The number of type-j servers serving type-i orders at the beginning of period k is g_k^{ij}. Some servers may not be assigned and are idle during period k. These

servers are automatically available at the beginning of period $k+1$. Servers may complete assigned orders and become available to serve new orders (called *service completion*). To implement the service completion process, we use another multiplicative random variable L_k^{ij} $(0 \le L_k^{ij} \le 1)$, which has mean $\bar{L}^{ij}$ and is independent over k, i, and j. Only L_k^{ij} fraction of type-j servers, assigned to type-i orders at the beginning of period k, remain working on the same assigned orders. The other servers complete their jobs during period k and become available to work on new orders at the beginning of period $k+1$. Thus, we have

$$g_{k+1}^{ij} = L_k^{ij}(g_k^{ij} + u_k^{ij}).$$

The defined order abandonment and service completion processes using multivariable fractional random variables are consistent with the memoryless assumption widely used in the related literature (e.g., Bassamboo et al. 2005, Huh et al. 2013, Xu et al. 1992). The multiplicative variables have been widely used to represent random processes because of their simplicity and technical tractability. Since their forms do not guarantee the integrality, the continuous state approximation has been used together in many cases (e.g., Chao et al. 2009, Huh et al. 2013). Following these prior studies, we use the multiplicative variables and continuous state space. Thus, the state of the system at time k is denoted by $\mathbf{x}_k = (\mathbf{x}_{1,k}, \mathbf{x}_{2,k}, \mathbf{g}_k)$, where $\mathbf{x}_{1,k} = (x_{1,k}^1, x_{1,k}^2, \ldots, x_{1,k}^I) \in \mathbf{R}^I$, $\mathbf{x}_{2,k} = (x_{2,k}^1, x_{2,k}^2, \ldots, x_{2,k}^I) \in \mathbf{R}^I$, and $\mathbf{g}_k = (g_k^{11}, g_k^{12}, \ldots, g_k^{IJ}) \in \mathbf{R}^{IJ}$.

To represent decisions, we use the post-decision state variables (Powell 2007) which are defined as

$$y_{1,k}^i = x_{1,k}^i + a_k^i - \sum_{j=1}^J u_k^{ij};$$

$$y_{2,k}^i = x_{2,k}^i - a_k^i - r_k^i;$$

$$q_k^{ij} = g_k^{ij} + u_k^{ij};$$

where $\mathbf{y}_k = (\mathbf{y}_{1,k}, \mathbf{y}_{2,k}, \mathbf{q}_k)$, $\mathbf{y}_{1,k} = (y_{1,k}^1, y_{1,k}^2, \ldots, y_{1,k}^I) \in \mathbf{R}^I$, $\mathbf{y}_{2,k} = (y_{2,k}^1, y_{2,k}^2, \ldots, y_{2,k}^I) \in \mathbf{R}^I$, and $\mathbf{q}_k = (q_k^{11}, q_k^{12}, \ldots, q_k^{IJ}) \in \mathbf{R}^{IJ}$. Hence, $\mathbf{x}_k$ represents a state before making decisions, while $\mathbf{y}_k$ represents an instantaneous state right after decisions are made at time k. We define a function Π: $\mathbf{R}^{2I+IJ} \to \mathbf{R}^{2I+IJ}$,

$$\Pi(\mathbf{y}_k; \mathbf{Z}_k, \mathbf{W}_k, \mathbf{L}_k) = (y_{1,k}^1, y_{1,k}^2, \ldots, y_{1,k}^I, W_k^1 y_{2,k}^1 + Z_k^1, W_k^2 y_{2,k}^2 + Z_k^2, \ldots, W_k^I y_{2,k}^I + Z_k^I, L_k^{11} q_k^{11}, L_k^{12} q_k^{12}, \ldots, L_k^{IJ} q_k^{IJ}).$$

$\Pi(\mathbf{y}_k; \mathbf{Z}_k, \mathbf{W}_k, \mathbf{L}_k)$ represents the transition state space for $\mathbf{y}_k$ after random events occur, including the arrivals of new orders, abandonments of orders, and service completions of orders. We assume all these random events occur at the end of each period. For generalizability, we do not assume any particular functional form for the probability distribution of these random events as in Akçay et al. (2010).

Revenue p^i is earned when a type-i order is assigned to a server. The total waiting penalties are linear to waiting time. This linearity assumption for waiting penalties has been widely adopted in the related literature (e.g., Bassamboo et al. 2005, Xu et al. 1992). Waiting penalties, b_1^i and b_2^i, are incurred for letting a type-i order wait in the assignment and acceptance queue for a period, respectively. We assume $b_1^i \ge b_2^i$ for each i, and they accrue at the beginning of each period. For the sake of simplicity, we assume stationary revenues and waiting penalties. We also assume homogeneous service costs among different types of servers and exclude them from the model following the literature (e.g., Xu et al. 1992). Our model, however, can relax any of these assumptions without changing the main results.

The expected profit function for the remaining periods, at the beginning of period k, is defined as follows:

$$\psi_k(\mathbf{x}_k, \mathbf{y}_k) = \sum_{i=1}^I \left(p^i \sum_{j=1}^J (q_k^{ij} - g_k^{ij}) - b_1^i y_{1,k}^i - b_2^i y_{2,k}^i \right) + E[v_{k+1}(\Pi(\mathbf{y}_k; \mathbf{Z}_k, \mathbf{W}_k, \mathbf{L}_k))].$$

Then $\psi_k(\mathbf{x}_k, \mathbf{y}_k)$ can be represented by $\varphi_k(\mathbf{x}_k) + \phi_k(\mathbf{y}_k)$, where

$$\varphi_k(\mathbf{x}_k) = -\sum_{i=1}^I \sum_{j=1}^J p^i g_k^{ij} \quad \text{and}$$

$$\phi_k(\mathbf{y}_k) = \sum_{i=1}^I \sum_{j=1}^J p^i q_k^{ij} - \sum_{i=1}^I (b_1^i y_{1,k}^i + b_2^i y_{2,k}^i) + E[v_{k+1}(\Pi(\mathbf{y}_k; \mathbf{Z}_k, \mathbf{W}_k, \mathbf{L}_k))].$$

The maximum expected profit given state $\mathbf{x}_k$ at time k is defined and decomposed as follows:

$$v_k(\mathbf{x}_k) = \max \psi_k(\mathbf{x}_k, \mathbf{y}_k) = \max(\varphi_k(\mathbf{x}_k) + \phi_k(\mathbf{y}_k)) = \varphi_k(\mathbf{x}_k) + \max \phi_k(\mathbf{y}_k).$$

$v_k(\mathbf{x}_k)$ is obtained by solving the following maximization problem (P) for every $\mathbf{x}_k$and k:

$$\tau_k(\mathbf{x}_k) := \max \phi_k(\mathbf{y}_k)$$

$$\text{s.t.} \quad y_{1,k}^i + y_{2,k}^i + \sum_{j=1}^J q_k^{ij} \le x_{1,k}^i + x_{2,k}^i + \sum_{j=1}^J g_k^{ij}, \quad i = 1, 2, \ldots, I \quad (1)$$

$$y_{1,k}^i + \sum_{j=1}^J q_k^{ij} \ge x_{1,k}^i + \sum_{j=1}^J g_k^{ij}, \quad i = 1, 2, \ldots, I \quad (2)$$

$$\sum_{i=1}^I q_k^{ij} \le S^j, \quad j = 1, 2, \ldots, J \quad (3)$$

$$q_k^{ij} \geq g_k^{ij} \tag{4}$$

$$y_{1,k}^i, y_{2,k}^i, q_k^{ij} \geq 0. \tag{5}$$

The optimal policy is defined as $\mathbf{y}_k^*(\mathbf{x}_k) = \arg\max_{\mathbf{y}_k} \phi_k(\mathbf{y}_k)$ s.t. (1)–(5) for all $\mathbf{x}_k$ and k. Constraint set (1) represents the number of orders in the system must not increase after making decisions. Constraint set (2) shows that the number of accepted orders, including those being served, must not decrease after making decisions. Constraint set (3) enforces that the number of assigned orders to each server type must not exceed the number of servers of that type. Constraint set (4) shows assignments cannot be negative. Finally, constraint set (5) restricts all variables to be nonnegative. Without loss of generality, we assume that $\mathbf{x}_1 = 0$ and $v_{n+1}(\mathbf{y}_{n+1}) = 0$. A notation table is provided in the appendix. Finding the optimal policy for MSSPAA requires solving (P) for every state at every period. This is computationally intractable because the state space grows exponentially with the numbers of orders and servers (Powell 2007). Therefore, we are interested in characterizing the structure of the optimal policy.

4. Structural Properties

We analyze the structural properties of the objective function of maximization problem (P) using concavity (Section 4.1), submodularity (Section 4.2), and the directional property (Section 4.3). The structural properties derived here are used to characterize the structure of the optimal policy of MSSPAA in Section 5. The proofs of theorems, lemmas, and propositions are provided in the online appendix (available as supplemental material at http://dx.doi.org/10.1287/msom.2016.0581).

4.1. Concavity

First of all, we are interested in the existence of the optimal policy. Note that the constraint set (1)–(5) of (P) forms a closed convex set. Therefore, if $\phi_k(\mathbf{y}_k)$ is jointly concave in $\mathbf{y}_k$, $\mathbf{y}_k^*(\mathbf{x}_k)$ always exists for any $\mathbf{x}_k$ (Chao et al. 2009, Topkis 1998). $\phi_k(\mathbf{y}_k)$ can be decomposed into $\phi_k^1(\mathbf{y}_k) = \sum_{i=1}^I \sum_{j=1}^J p^i q_k^{ij} - \sum_{i=1}^I (b_1^i y_{1,k}^i + b_2^i y_{2,k}^i)$, representing profit incurred during period k, and $\mathbb{E}[v_{k+1}(\Pi(\mathbf{y}_k; \mathbf{Z}_k, \mathbf{W}_k, \mathbf{L}_k))]$, representing the expected maximum profit realized during the remaining periods (called *profit-to-go*). $\phi_k^1(\mathbf{y}_k)$ is linear and jointly concave in $\mathbf{y}_k$. This means the concavity of $\phi_k(\mathbf{y}_k)$ depends on if $\mathbb{E}[v_{k+1}(\Pi(\mathbf{y}_k; \mathbf{Z}_k, \mathbf{W}_k, \mathbf{L}_k))]$ is concave in $\mathbf{y}_k$. Note that given any $\mathbf{z}_k^i \in \mathbf{Z}_k^i$, $\mathbf{w}_k^i \in \mathbf{W}_k^i$, and $\mathbf{l}_k^{ij} \in \mathbf{L}_k^{ij}$ for any i and j, $\Pi(\mathbf{y}_k; \mathbf{z}_k, \mathbf{w}_k, \mathbf{l}_k)$ is a linear transformation of $\mathbf{y}_k$. This means if $v_{k+1}(\mathbf{y}_k)$ is concave, $\mathbb{E}[v_{k+1}(\Pi(\mathbf{y}_k; \mathbf{Z}_k, \mathbf{W}_k, \mathbf{L}_k))]$ is concave in $\mathbf{y}_k$, too. As a result, $\phi_k(\mathbf{y}_k)$ is concave if $v_k(\mathbf{y}_k)$ is concave in $\mathbf{y}_k$.

Theorem 1. *$v_k(\mathbf{y}_k)$ is jointly concave in $\mathbf{y}_k$, for $k = 1, 2, \ldots, n$.*

Theorem 1 shows the existence of the optimal policy: $\mathbf{y}_k^*(\mathbf{x}_k)$ for all $\mathbf{x}_k$ and k.

4.2. Submodularity

We now investigate whether $\phi_k(\mathbf{y}_k)$ is submodular in $\mathbf{y}_k$. Let $f(x_i, x_j)$ be a real function defined in $\mathbf{R}^2$. $f(x_i, x_j)$ is submodular if $f(x_i, x_j) + f(x_i + \varepsilon_i, x_j + \varepsilon_j) \leq f(x_i + \varepsilon_i, x_j) + f(x_i, x_j + \varepsilon_j)$ for any (x_i, x_j), $\varepsilon_i \geq 0$, and $\varepsilon_j \geq 0$ (Topkis 1998, Yang and Qin 2007). Note that $\phi_k^1(\mathbf{y}_k)$ is linear in $\mathbf{y}_k$; therefore, $\phi_k^1(\mathbf{y}_k)$ is submodular in $\mathbf{y}_k$. The sum of two submodular functions is submodular (Powell 2007) which means that $\phi_k(\mathbf{y}_k)$ is submodular in $\mathbf{y}_k$ if $\mathbb{E}[v_{k+1}(\Pi(\mathbf{y}_k; \mathbf{Z}_k, \mathbf{W}_k, \mathbf{L}_k))]$ is submodular in $\mathbf{y}_k$. Note that $\Pi(\mathbf{y}_k; \mathbf{Z}_k, \mathbf{W}_k, \mathbf{L}_k)$ is stochastically increasing in $\mathbf{Z}_k, \mathbf{W}_k$, and $\mathbf{L}_k$ (Shaked and Shanthikumar 1994). Therefore, if $v_{k+1}(\mathbf{y}_k)$ is submodular, $\mathbb{E}[v_{k+1}(\Pi(\mathbf{y}_k; \mathbf{Z}_k, \mathbf{W}_k, \mathbf{L}_k))]$ becomes submodular, too (Chao et al. 2009). This implies that the submodularity of $\phi_k(\mathbf{y}_k)$ eventually depends on the submodularity of $v_k(\mathbf{y}_k)$.

Lemma 2. *If $f(x_i, x_j)$ is submodular and concave in (x_i, x_j), then for any $x_i \geq 0$ and $x_j \geq 0$, all*
(1) *$g(x_i, x_j) = \max_{y_i \geq 0, y_j \geq 0} f(y_i, y_j)$ s.t. $y_i + y_j \leq x_i + x_j$ and $y_i \geq x_i$,*
(2) *$g(x_i, x_j) = \max_{y_i \geq 0, y_j \geq 0} f(y_i, y_j)$ s.t. $y_i + y_j = x_i + x_j$ and $y_j \geq x_j$,*
(3) *$g(x_i, x_j) = \max_{y_i \geq 0, y_j \geq 0} f(y_i, y_j)$ s.t. $y_i \leq x_i$ and $y_j \leq x_j$, and*
(4) *$g(x_i, x_j) = \max_{y_i \geq 0, y_j \geq 0} f(y_i, y_j)$ s.t. $y_i = x_i$ and $y_j \leq x_j$*
are submodular in (x_i, x_j).

Maximization does not preserve submodularity in general (Chao et al. 2009). Some studies find submodularity is preserved under certain types of maximization (e.g., Chao et al. 2009). Our problem (P) does not belong to any of these known types. In Lemma 2, we find new types of maximization where submodularity is preserved. This proves the submodularity of $v_k(\mathbf{y}_k)$ in Theorem 3.

Theorem 3. *$v_k(\mathbf{y}_k)$ is submodular in $\mathbf{y}_k$, for $k = 1, 2, \ldots, n$.*

Submodularity provides important structural information of a function. Let $x_i^*(\hat{x}_j) = \arg\max_{x_i} f(x_i, x_j)$ s.t. $x_j = \hat{x}_j$ and $x_j^*(\hat{x}_i) = \arg\max_{x_j} f(x_i, x_j)$ s.t. $x_i = \hat{x}_i$. Then if $f(x_i, x_j)$ is submodular, $x_i^*(\hat{x}_j)$ decreases in $\hat{x}_j$, and $x_j^*(\hat{x}_i)$ decreases in $\hat{x}_i$. This structural information has been used to characterize the monotonic optimal policy in the literature (e.g., Powell 2007, Ross 1983). Some decisions in MSSPAA are represented as vertical or horizontal movements in a two-dimensional space. For example, the rejection decision of type-i order can be represented as a vertical movement in $(x_{1,k}^i, x_{2,k}^i)$.

Theorem 3 reveals the monotonicity of the optimal rejection decision w.r.t. $x^i_{1,k}$. Some decisions in MSSPAA, however, are not represented by this vertical or horizontal movement in a plane. We discuss them in the following section.

4.3. The Directional Property

Diagonal movements along a line of -45 degrees (i.e., $(1,-1)$ or $(-1,1)$ direction) in a two-dimensional space, consisting of two state variables, represent some important decisions in MSSPAA. For example, acceptance is represented as a movement of $(1,-1)$ direction in the two-dimensional plane of $(x^i_{1,k}, x^i_{2,k})$. Assignment is represented as the same movement in $(g^{ij}_k, x^i_{1,k})$ or $(g^{ij}_k, x^i_{2,k})$.

We say a real function $f(x_1,x_2)$ in $\mathbf{R}^2$ is subconcave (superconvex) in x_1 if $f(x_1,x_2)+f(x_1+\delta+\delta_1,x_2-\delta) \le (\ge) f(x_1+\delta,x_2-\delta)+f(x_1+\delta_1,x_2)$, any (x_1,x_2), $\delta \ge 0$, and $\delta_1 \ge 0$ (Zhuang and Li 2012). When $f(x_1,x_2)$ is superconvex in both elements, x_1 and x_2, we say $f(x_1,x_2)$ has diagonal dominance. This diagonal dominance has been used to examine the structural properties of objective functions with respect to these types of diagonal movements in the operations management literature (Altman et al. 2000, Zhuang and Li 2012, Ha 1997, Yang and Qin 2007, Zhao et al. 2008).

In this study, however, we are interested in a different condition from diagonal dominance that $f(x_1,x_2)$ is subconcave in x_1 but superconvex in x_2, which we refer to as the *directional property* (see Definition 4). In particular, we investigate the combined structural properties of a two-dimensional objective function when it has concavity, submodularity, and the directional property (see Lemmas 5 and 6). We also find their preservation conditions in stochastic dynamic optimization (see Lemmas 7 and 8).

Definition 4 (The Directional Property). Function $f(x_i,x_j)$ has the directional property if $f(x_i,x_j)$ is subconcave in x_i but superconvex in x_j.

Let $(\bar{x}^*_i(\hat{x}_i,\hat{x}_j), \bar{x}^*_j(\hat{x}_i,\hat{x}_j)) = \arg\max_{(x_i,x_j)} f(x_i,x_j)$ s.t. $x_i+x_j=\hat{x}_i+\hat{x}_j$ which represents a point maximizing $f(x_i,x_j)$ on the -45° line passing $(\hat{x}_i,\hat{x}_j)$. Concavity, submodularity, and the directional property all together provide unique structural properties of $(\bar{x}^*_i(\hat{x}_i,\hat{x}_j), \bar{x}^*_j(\hat{x}_i,\hat{x}_j))$, $x^*_i(\hat{x}_j)$, and $x^*_j(\hat{x}_i)$.

Lemma 5. *If $f(x_i,x_j)$ has concavity, submodularity, and the directional property, then for any $\varepsilon>0$, (1) when $\hat{x}_i+\hat{x}_j = x_i+x_j+\varepsilon$, $\bar{x}^*_i(x_i,x_j)-\bar{x}^*_i(\hat{x}_i,\hat{x}_j) \le \varepsilon$ and $\bar{x}^*_j(\hat{x}_i,\hat{x}_j)-\bar{x}^*_j(x_i,x_j) \ge \varepsilon$; (2) when $\hat{x}_j = x_j+\varepsilon$, $0 \le x^*_i(\hat{x}_j)-x^*_i(x_j) \le \varepsilon$; and (3) when $\hat{x}_i = x_i+\varepsilon$, $x^*_j(\hat{x}_i)-x^*_j(x_i) \ge \varepsilon$.*

The directional property shows how the optimal balance between two elements $(\bar{x}^*_i(x_i,x_j), \bar{x}^*_j(x_i,x_j))$ changes as the sum of these two elements (x_i+x_j) increases or decreases. Lemma 5(1) implies that as the sum increases (or decreases), the second element in the optimal balance increases (decreases) more than the increase (decrease) of the sum, while the first element in the optimal balance decreases (increases) less than the increase (decrease) of the sum. For example, suppose that $(3,3)$ is the optimal balance given the sum is 6. When the sum is 7, the second element in the optimal balance becomes more than $4(=3+1)$, while the first element becomes less than 3. The directional property also provides more information about the monotonicity of $x^*_i(x_j)$ and $x^*_j(x_i)$. Lemmas 5(2) and 5(3) imply the optimal value of the first (second) element decreases less (more) than the change of the other element. For example, suppose the optimal level of the first (second) element is 5 given the second (first) element is 1. When the second (first) element is 2, the optimal level of the first (second) element is more (less) than $4(=5-1)$.

Lemma 6. *If $f(x_i,x_j)$ has concavity, submodularity, and the directional property and $f(x_i,x_j)$ has an internal global optimum $(x^*_i,x^*_j)=\arg\max f(x_i,x_j)$, then*

(1) *(x^*_i,x^*_j) is unique; $x^*_i(x_j)$, $x^*_j(x_i)$, and $(\bar{x}^*_i(x_i,x_j), \bar{x}^*_j(x_i,x_j))$ intersect at (x^*_i,x^*_j) only once;*

(2) *for any (x_i,x_j) on $(\bar{x}^*_i(x_i,x_j), \bar{x}^*_j(x_i,x_j))$, $x_i \le x^*_i(x_j)$ if $x_j < x^*_j$, $x_i > x^*_i(x_j)$ otherwise;*

(3) *for any (x_i,x_j) on $(x^*_i(x_j),x_j)$, $x_j \le x^*_j(x_i)$ if $x_i > x^*_i$, $x_j > x^*_j(x_i)$ otherwise.*

Thus, Lemmas 5 and 6 provide useful information to characterize the structure of a two-dimensional optimal policy when the objective function has concavity, submodularity, and the directional property.

Lemma 7. *Let $h(x_i,x_j)=f(x_i,x_j)+g(x_i,x_j)$, where $f(x_i,x_j)$ and $g(x_i,x_j)$ are two distinct real functions in $\mathbf{R}^2$. If both $f(x_i,x_j)$ and $g(x_i,x_j)$ have concavity, submodularity, the directional property, then $h(x_i,x_j)$ has concavity, submodularity, and the directional property.*

Lemma 7 is intuitive given that concavity, submodularity, subconcavity, and superconvexity are preserved under addition. To examine the directional property of $\phi_k(\mathbf{y}_k)$, we use two-dimensional $\phi_k(\mathbf{y}_k)$ defined as $\phi_k(\mathbf{y}_k)$ projected into $\mathbf{R}^2$ having only two elements in the two-dimensional plane as variables and the other variables in the original state space as fixed constants. For simplicity, we use the same notations of $\phi_k(\mathbf{y}_k)$, $v_k(\mathbf{y}_k)$, $\mathbf{x}_k$, and $\mathbf{y}_k$ to represent two-dimensional functions and corresponding spaces, respectively. Recall $\phi^1_k(\mathbf{y}_k)$ is linear in $\mathbf{y}_k$. Therefore, it satisfies the directional property in every possible two-dimensional subspace of $\mathbf{y}_k$. Hence, by Lemma 7, whether $\phi_k(\mathbf{y}_k)$ has the directional property depends on if $\mathbb{E}[v_{k+1}(\Pi(\mathbf{y}_k;\mathbf{Z}_k,\mathbf{W}_k,\mathbf{L}_k))]$ has the directional property.

Subconcavity and superconvexity are not preserved under profit-to-go expectation in general. Given that the directional property consists of two different properties, one may expect that it is more difficult for

the directional property to be preserved under profit-to-go expectation. However, we find that the directional property is preserved in the following type of profit-to-go expectation.

Lemma 8. *Let* π: $\mathbf{R}^2 \to \mathbf{R}^2$ *be a function defined as* $\pi(x_i, x_j) = (x_i + a, cx_j + b)$, *where* a, b, *and* c *are random variables,* $a, b, c \in \mathbf{R}$, *and* $0 \le c \le 1$. *If* $f(x_i, x_j)$ *has concavity, submodularity and the directional property, then* $E[f(\pi(x_i, x_j))]$ *has concavity, submodularity, and the directional property.*

Lemmas 5–8 provide a significant methodological contribution to the related literature. Lemmas 5 and 6 are novel general properties that can be applied to any maximization problem of which the objective function has concavity, submodularity, and the directional property. Lemmas 7 and 8 provide some preservation conditions. In particular, Lemma 8 shows the type of profit-to-go expectation where the directional property is preserved. We expect that these four lemmas would have great application potential in dealing with various operations management problems.

In the next section the optimal policy of MSSPAA is characterized by the structure of the optimal policy in two-dimensional spaces. Lemma 8 provides a condition where the directional property is preserved; because of the limited preservation under profit-to-go expectation, the objective function of MSSPAA does not have the directional property in every possible two-dimensional space. The directional property is preserved in $(x^i_{1,k}, x^i_{2,k})$, where diagonal movements represent *accept*, and $(x^i_{1,k}, g^{ij}_k)$, where diagonal movements represent *assign*. Lemmas 7 and 8 imply that, in these two subspaces, $\phi_k(\mathbf{y}_k)$ has the directional property if $v_k(\mathbf{y}_k)$ has it. Accept and immediate assign can be represented as diagonal movements in $(g^{ij}_k, x^i_{2,k})$ as well. However, the preservation of the directional property under profit-to-go expectation is not guaranteed in $(g^{ij}_k, x^i_{2,k})$. However, concavity and submodularity still provide useful structural properties such as a switching curve structure and monotonicity.

Substitute-type decisions such as which type of server to assign, which type of order to assign, and which type of order to accept, between two alternatives, can be represented as diagonal movements in other two-dimensional spaces, including (g^{ij}_k, g^{il}_k), (g^{ij}_k, g^{lj}_k), $(x^i_{1,k}, x^l_{1,k})$, $(x^i_{2,k}, x^l_{2,k})$, and $(x^i_{1,k}, x^l_{2,k})$. In these two-dimensional spaces, $\phi_k(\mathbf{y}_k)$ does not have the directional property. However, concavity and submodularity still provide useful information for these decisions for a given state. For instance, $\phi_k(q^{ij}_k, q^{il}_k) < \phi_k(q^{ij}_k + 1, q^{il}_k - 1)$ implies that type-j server is better than type-l server to assign a type-i order for a given state. The state space is divided by $(\tilde{q}^{ij*}_k(g^{ij}_k, g^{il}_k), \tilde{q}^{il*}_k(g^{ij}_k, g^{il}_k))$ into two areas: one area where type-j is better than type-l server and another area where type-l is better than type-j server.

5. Structure of the Optimal Policy

Using the structural properties of the objective function of MSSPAA presented in Section 4, we characterize the structure of the optimal policy for MSSPAA. MSSPAA is multidimensional. Although we find some multidimensional structural properties for MSSPAA, including concavity and submodularity, it is still challenging to describe the multidimensional optimal policy (Zhuang and Li 2012). Therefore, we discuss the structure of the optimal policy in two-dimensional spaces and expand our discussion to the special cases of MSSPAA where the entire optimal policy is relatively easily visualized. For simplicity, we just use "order" to refer to "type-i order" and "server" to refer to "type-j server" if they are not confusing.

5.1. Structure of the Optimal Policy in Two-Dimensional Spaces

The optimal policy in the two-dimensional spaces forms a switching curve policy where several curves (functions) separate the space into multiple adjacent areas and different actions are optimal for states in each area. We discuss the structure of the optimal policy in $(x^i_{1,k}, x^i_{2,k})$, $(x^i_{1,k}, g^{ij}_k)$, and $(g^{ij}_k, x^i_{2,k})$.

5.1.1. Optimal Policy in $(\mathbf{x}^i_{1,k}, \mathbf{x}^i_{2,k})$. Lemmas 6–8 lead to the $(y^i_{1,k}, y^i_{2,k})$-policy, specified in Proposition 9. The policy shows how to optimally accept, reject, or postpone accepting type-i order given that everything else is fixed.

Proposition 9. *Let the* $(y^i_{1,k}, y^i_{2,k})$*-policy be as follows*:

$$(y^i_{1,k}, y^i_{2,k}) = \begin{cases} (\tilde{y}^{i*}_{1,k}(x^i_{1,k}, x^i_{2,k}), \tilde{y}^{i*}_{2,k}(x^i_{1,k}, x^i_{2,k})) \\ \quad \textit{when } x^i_{1,k} + x^i_{2,k} < y^{i*}_{1,k} + y^{i*}_{2,k} \textit{ and} \\ \quad\quad x^i_{1,k} < \tilde{y}^{i*}_{1,k}(x^i_{1,k}, x^i_{2,k}) (\textit{Area } 1) \\ (y^{i*}_{1,k}, y^{i*}_{2,k}) \\ \quad \textit{when } x^i_{1,k} + x^i_{2,k} \ge y^{i*}_{1,k} + y^{i*}_{2,k} \textit{ and} \\ \quad\quad x^i_{1,k} < y^{i*}_{1,k} (\textit{Area } 2) \\ (x^i_{1,k}, y^{i*}_{2,k}(x^i_{1,k})) \\ \quad \textit{when } x^i_{1,k} \ge y^{i*}_{1,k} \textit{ and} \\ \quad\quad x^i_{2,k} \ge y^{i*}_{2,k}(x^i_{1,k}) (\textit{Area } 3) \\ (x^i_{1,k}, x^i_{2,k}) \\ \quad \textit{when } x^i_{1,k} \ge \tilde{y}^{i*}_{1,k}(x^i_{1,k}, x^i_{2,k}) \textit{ and} \\ \quad\quad x^i_{2,k} < y^{i*}_{2,k}(x^i_{1,k}), (\textit{Area } 4) \end{cases}$$

where

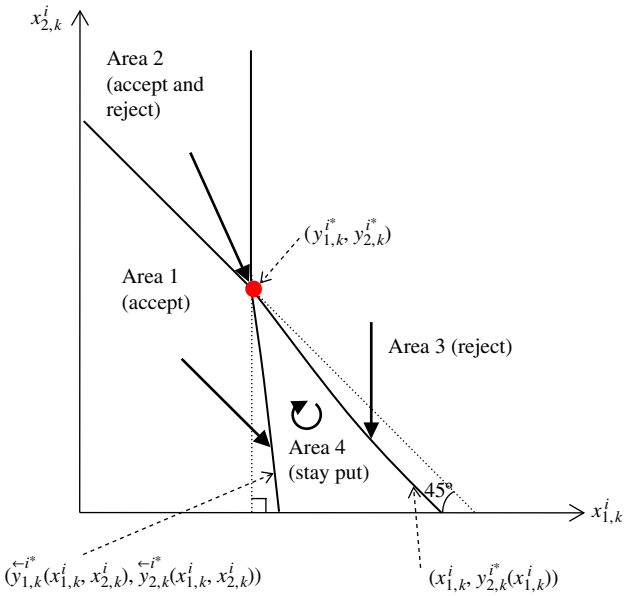
$$(\tilde{y}^{i*}_{1,k}(x^i_{1,k}, x^i_{2,k}), \tilde{y}^{i*}_{2,k}(x^i_{1,k}, x^i_{2,k})) = \arg\max_{(y^i_{1,k}, y^i_{2,k})} \phi(y^i_{1,k}, y^i_{2,k}) \quad s.t.\ y^i_{1,k} + y^i_{2,k} = x^i_{1,k} + x^i_{2,k};$$

$$(y^{i*}_{1,k}, y^{i*}_{2,k}) = \arg\max_{(y^i_{1,k}, y^i_{2,k})} \phi(y^i_{1,k}, y^i_{2,k});$$

$$y^{i*}_{2,k}(x^i_{1,k}) = \arg\max_{y^i_{2,k}} \phi(y^i_{1,k}, y^i_{2,k}) \quad s.t.\ y^i_{1,k} = x^i_{1,k}.$$

Then the $(y^i_{1,k}, y^i_{2,k})$*-policy is optimal.*

Figure 1 (Color online) Structure of the $(y^i_{1,k}, q^{ij}_k)$-Policy



The structure of the $(y^i_{1,k}, y^i_{2,k})$-policy is shown in Figure 1. The solid lines represent switching curves that divide the state space into multiple adjacent areas, and the solid arrows represent the optimal decision in each area. The dashed arrows and lines are used to provide additional information for switching curves or points. $(y^{i*}_{1,k}, y^{i*}_{2,k})$, the intersection point of all four areas, represent the optimal numbers of orders to keep in the assignment and acceptance queues, respectively, at which the two-dimensional $\phi_k(y^i_{1,k}, y^i_{2,k})$ is maximized. $(\tilde{y}^{i*}_{1,k}(x^i_{1,k}, x^i_{2,k}), \tilde{y}^{i*}_{2,k}(x^i_{1,k}, x^i_{2,k}))$, the borderline between areas 1 and 4, is the optimal balance between the numbers of orders to keep in the assignment and acceptance queues when the total number of orders in both queues is $x^i_{1,k} + x^i_{2,k}$. $y^{i*}_{2,k}(x^i_{1,k})$, the borderline between areas 3 and 4, denotes the optimal number of orders to keep in the acceptance queue when $x^i_{1,k}$ orders are in the assignment queue.

The $(y^i_{1,k}, y^i_{2,k})$-policy consists of four areas based on whether or not the total number of orders currently waiting in both queues $(x^i_{1,k} + x^i_{2,k})$ is greater than the optimal number of orders to be in the two queues $(y^{i*}_{1,k} + y^{i*}_{2,k})$ and whether or not the number of orders currently in the assignment queue $(x^i_{1,k})$ is greater than the optimal number of orders to be in the assignment queue $(y^{i*}_{1,k})$. When $x^i_{1,k} + x^i_{2,k} \ge y^{i*}_{1,k} + y^{i*}_{2,k}$ and $x^i_{1,k} < y^{i*}_{1,k}$ (area 2), there are enough orders as a whole but not in the assignment queue so that $(y^{i*}_{1,k}, y^{i*}_{2,k})$ can be reached by accepting and rejecting orders. However, when $x^i_{1,k} + x^i_{2,k} \ge y^{i*}_{1,k} + y^{i*}_{2,k}$ and $x^i_{1,k} \ge y^{i*}_{1,k}$ (area 3), more than necessary orders are waiting in the queues, especially in the assignment queue. So no more orders are to be accepted. Instead, $y^{i*}_{2,k}(x^i_{1,k})$, fewer than the optimum $(\le y^{i*}_{2,k})$, orders should be in the acceptance queue to decrease the waiting penalty. When $x^i_{1,k} + x^i_{2,k} < y^{i*}_{1,k} + y^{i*}_{2,k}$ and $x^i_{1,k} < y^{i*}_{1,k}$ (area 1), overall there is a shortage of orders. No order is to be rejected. Instead, orders are to be accepted so that $\tilde{y}^{i*}_{1,k}(x^i_{1,k}, x^i_{2,k})(\ge y^{i*}_{1,k})$ and $\tilde{y}^{i*}_{2,k}(x^i_{1,k}, x^i_{2,k})(\le y^{i*}_{2,k})$ orders remain in the assignment queue and the acceptance queue, respectively. However, when $x^i_{1,k} + x^i_{2,k} < y^{i*}_{1,k} + y^{i*}_{2,k}$ and $x^i_{1,k} \ge y^{i*}_{1,k}$, three different cases exist. First, when $x^i_{2,k} \ge y^{i*}_{2,k}(x^i_{1,k})$ (area 3), some orders are to be rejected. Second, when $x^i_{2,k} < y^{i*}_{2,k}(x^i_{1,k})$ and $x^i_{2,k} < \tilde{y}^{i*}_{2,k}(x^i_{1,k}, x^i_{2,k})$ (area 4), no order is either rejected or accepted because rejecting orders increases the risk of shortage of orders in subsequent periods and accepting orders increases the risk of having too many committed orders in hand. Finally, when $\tilde{y}^{i*}_{2,k}(x^i_{1,k}, x^i_{2,k}) \le x^i_{2,k} < y^{i*}_{2,k}(x^i_{1,k})$ (area 1), too few orders are available as a whole. More orders are accepted until $\tilde{y}^{i*}_{2,k}(x^i_{1,k}, x^i_{2,k})$ orders remain in the acceptance queue to increase the availability of them in subsequent periods.

Lemmas 5 provide the following additional properties of the $(y^i_{1,k}, y^i_{2,k})$-policy.

(1) *For any* $\mathbf{x}_k$, $y^{i*}_{1,k}(\mathbf{x}_k) \ge y^{i*}_{1,k}$, $y^{i*}_{2,k}(\mathbf{x}_k) \le y^{i*}_{2,k}$, *and* $y^{i*}_{1,k}(\mathbf{x}_k) + y^{i*}_{2,k}(\mathbf{x}_k) \le y^{i*}_{1,k} + y^{i*}_{2,k}$.

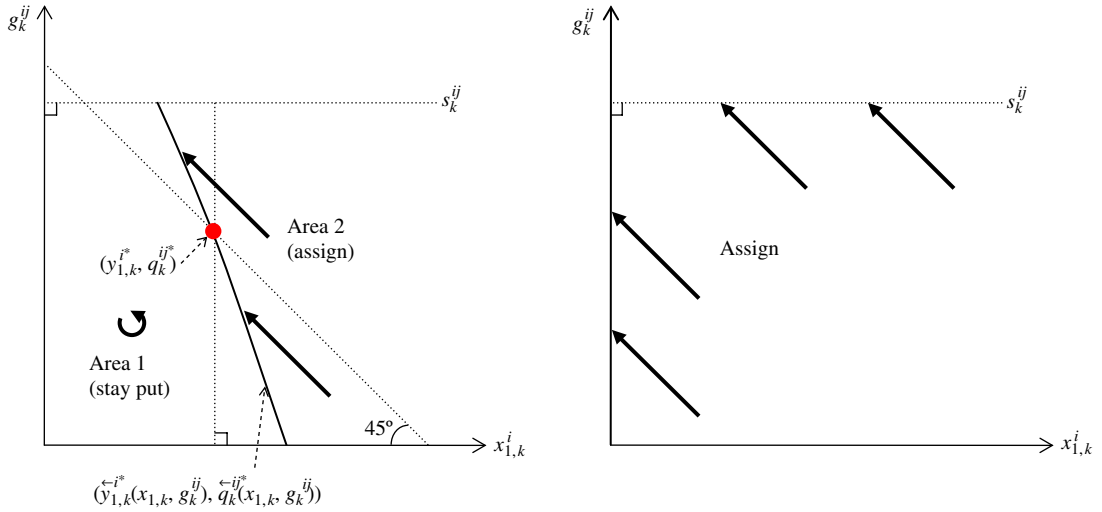
(2) *For any* $\varepsilon \ge 0$, *there exists some* $\gamma \ge 0$ *such that if* $\mathbf{x}_k = (y^{i*}_{1,k} + \varepsilon, y^{i*}_{2,k})$, *then* $\mathbf{y}^*_k(\mathbf{x}_k) = (y^{i*}_{1,k} + \varepsilon, y^{i*}_{2,k} - \varepsilon - \gamma)$ *and* γ *increases in* ε.

(3) *For any* $\varepsilon \ge 0$ *and* $\varepsilon' \ge 0$, *there exists some* $\beta \ge 0$ *such that if* $\mathbf{x}_k = (y^{i*}_{1,k} - \varepsilon, y^{i*}_{2,k} - \varepsilon')$, *then* $\mathbf{y}^*_k(\mathbf{x}_k) = (y^{i*}_{1,k} + \beta, y^{i*}_{2,k} - \varepsilon - \varepsilon' - \beta)$ *and* β *increases in* $\varepsilon + \varepsilon'$.

The above three properties provide insights on how to minimize the waiting penalty and to maintain a certain level of availability of orders for subsequent periods by balancing the number of orders in the assignment and acceptance queues. The most interesting property is that in any situation, the decision maker should keep at least $y^{i*}_{1,k}$ orders in the assignment queue, at most $y^{i*}_{2,k}$ orders in the acceptance queue, and at most $y^{i*}_{1,k} + y^{i*}_{2,k}$ orders in the both queues (1). When there are ε more orders than the optimal level $(y^{i*}_{1,k})$ in the assignment queue, the decision maker should have $\varepsilon + \gamma$ fewer orders than the optimal level $(y^{i*}_{2,k})$ in the acceptance queue and γ increases in ε (according to Lemma 5(3))(2). By doing so, she decreases the waiting penalty while keeping the similar level of availability of orders for subsequent periods. When there are fewer orders than the optimal levels in both queues, the decision maker should keep β more orders than $y^{i*}_{1,k}$ in the assignment queue to increase the availability of orders for subsequent periods. β increases as the orders in both queues become fewer (according to Lemma 5(1))(3).

5.1.2. Optimal Policy in $(\mathbf{x}^i_{1,k}, \mathbf{g}^{ij}_k)$. Lemmas 6–8 lead to the $(y^i_{1,k}, q^{ij}_k)$-policy, specified in Proposition 10, which explains how to optimally assign type-i orders to type-j servers or postpone assignment given that everything else is fixed.

Figure 2 **(Color online) Structure of the $(y^i_{1,k}, q^{ij}_k)$-Policy**



Proposition 10. *Let the $(y^i_{1,k}, q^{ij}_k)$-policy be as follows*:

$$(y^i_{1,k}, q^{ij}_k) = \begin{cases} (x^i_{1,k}, g^{ij}_k) & \text{when } g^{ij}_k \geq \tilde{q}^{ij*}_k(x^i_{1,k}, g^{ij}_k) \text{ (Area 1)} \\ (\tilde{y}^{i*}_{1,k}(x^i_{1,k}, g^{ij}_k), \tilde{q}^{ij*}_k(x^i_{1,k}, g^{ij}_k)) & \\ \qquad \text{when } g^{ij}_k < \tilde{q}^{ij*}_k(x^i_{1,k}, g^{ij}_k), \text{ (Area 2)} & \end{cases}$$

where

$$(\tilde{y}^{i*}_{1,k}(x^i_{1,k}, g^{ij}_k), \tilde{q}^{ij*}_k(x^i_{1,k}, g^{ij}_k)) = \arg\max_{(y^i_{1,k}, q^{ij}_k)} \phi(y^i_{1,k}, q^{ij}_k) \quad s.t.\ y^i_{1,k} + q^{ij}_k = x^i_{1,k} + g^{ij}_k,$$
$$q^{ij}_k \leq s^{ij}_k, s^{ij}_k = S^j - \sum_{l \neq i} q^{lj}_k.$$

Then the $(y^i_{1,k}, q^{ij}_k)$-policy is optimal.

The structure of the $(y^i_{1,k}, q^{ij}_k)$-policy is represented in Figure 2. Figure 2(b) represents a special case when type-i order is the best fit for type-j server, which is in turn the best fit for type-i order. In this special case there is no need for assignment postponement. Orders are assigned to servers until no server or order is available. Figure 2(a) depicts the other general cases. Since type-i order is not the best fit for type-j server or type-j server is not the best fit for type-i order, an internal point $(y^{i*}_{1,k}, q^{ij*}_k)$ may maximize $\phi_k(y^i_{1,k}, q^{ij}_k)$. The policy is first characterized by the number of accepted but not completed orders $(x^i_{1,k} + g^{ij}_k)$ and the number of orders in the assignment queue $(x^i_{1,k})$. Orders always need to be assigned if $x^i_{1,k} + g^{ij}_k \geq y^{i*}_{1,k} + q^{ij*}_k$ and $x^i_{1,k} \geq y^{i*}_{1,k}$, while assignment always needs to be postponed if $x^i_{1,k} + g^{ij}_k \leq y^{i*}_{1,k} + q^{ij*}_k$ and $x^i_{1,k} \leq y^{i*}_{1,k}$. This implies that when there are more orders than the optimal level, especially in the assignment queue, they need to be assigned to avoid an unnecessary waiting penalty. On the other hand, when there are fewer orders than the optimal level, especially in the assignment queue, order assignment needs to be postponed. More specifically, the optimal balance between orders in the assignment queue and orders assigned to servers $(\tilde{y}^{i*}_{1,k}(x^i_{1,k}, g^{ij}_k), \tilde{q}^{ij*}_k(x^i_{1,k}, g^{ij}_k))$, the borderline between areas 1 and 2, depends on the total number of accepted but not completed orders $(x^i_{1,k} + g^{ij}_k)$. Thus, the optimal policy consists of two areas based on whether or not the number of orders currently served (g^{ij}_k) is greater than the optimal number of orders to be served $(\tilde{q}^{ij*}_k(x^i_{1,k}, g^{ij}_k))$ given $x^i_{1,k} + g^{ij}_k$. When $g^{ij}_k < \tilde{q}^{ij*}_k(x^i_{1,k}, g^{ij}_k)$ (area 2), orders need to be assigned more up to $\min(\tilde{q}^{ij*}_k(x^i_{1,k}, g^{ij}_k), s^{ij}_k)$ where s^{ij}_k represents the maximum number of type-j servers that can be assigned to type-i orders at time k. On the other hand, when $g^{ij}_k \geq \tilde{q}^{ij*}_k(x^i_{1,k}, g^{ij}_k)$ (area 1), too many orders are already assigned to servers so that no more orders are to be assigned.

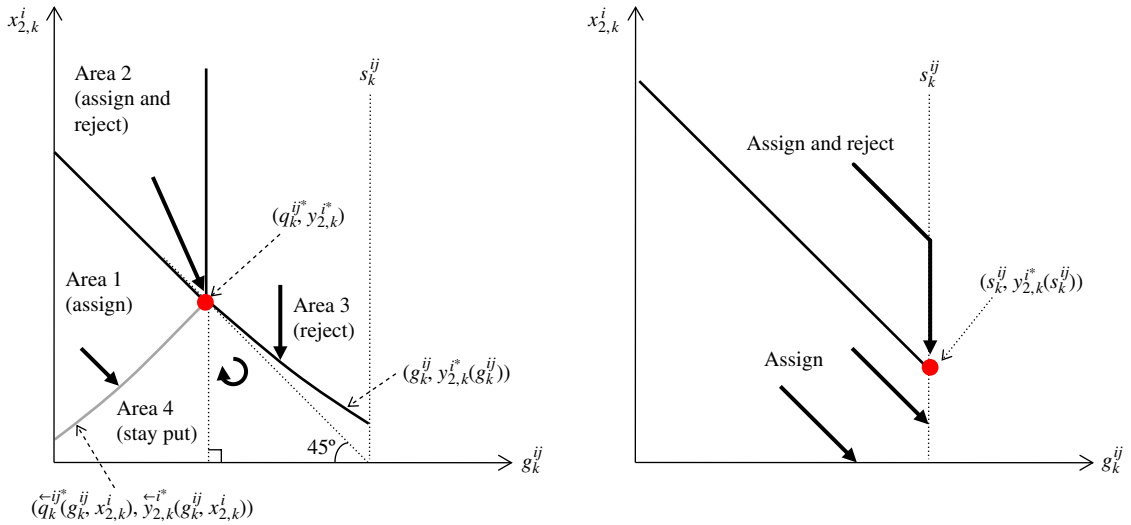
Lemma 5 provides the following additional properties of the $(y^i_{1,k}, q^{ij}_k)$-policy.

(1) *For any $\varepsilon \geq 0$, there exists some $\alpha \geq 0$ such that if $x^i_{1,k} + g^{ij}_k = y^{i*}_{1,k} + q^{ij*}_k + \varepsilon$, then $\mathbf{y}^*_k(\mathbf{x}_k) = (y^{i*}_{1,k} - \alpha, q^{ij*}_k + \varepsilon + \alpha)$ and α increases in ε.*

(2) *For any $\varepsilon \geq 0$, there exists some $\theta \geq 0$ such that if $x^i_{1,k} + g^{ij}_k = y^{i*}_{1,k} + q^{ij*}_k - \varepsilon$, then $\mathbf{y}^*_k(\mathbf{x}_k) = (y^{i*}_{1,k} + \theta q^{ij*}_k - \varepsilon - \theta)$ and θ increases in ε.*

The above properties show how to control the level of committed but incomplete orders in hand. When there are ε more committed orders than the optimal level $(y^{i*}_{1,k} + q^{ij*}_k)$ in hand, the decision maker should push out orders from the system by assigning $\varepsilon + \alpha$ more orders than the optimal level (q^{ij*}_k) to servers (according to Lemma 5(1)) (1). However, when there are ε fewer committed orders than $y^{i*}_{1,k} + q^{ij*}_k$ in hand,

Figure 3 (Color online) Structure of the $(q_k^{ij}, y_{2,k}^i)$-Policy



she should postpone θ more orders than the optimal level $(y_{1,k}^{i*})$ so that she may assign them to more productive servers later (according to Lemma 5(1)) (2).

5.1.3. Optimal Policy in $(\mathbf{g}_k^{ij}, \mathbf{x}_{2,k}^i)$. Theorems 1 and 3 lead to the $(q_k^{ij}, y_{2,k}^i)$-policy, specified in Proposition 11. The $(q_k^{ij}, y_{2,k}^i)$-policy explains how to optimally accept and immediately assign type-i orders to type-j servers, reject, or postpone this decision given that everything else is fixed.

Proposition 11. *Let the $(q_k^{ij}, y_{2,k}^i)$-policy be as follows*:

$$(q_k^{ij}, y_{2,k}^i) = \begin{cases} (\tilde{q}_k^{ij*}(g_k^{ij}, x_{2,k}^i), \tilde{y}_{2,k}^{i*}(g_k^{ij}, x_{2,k}^i)) \\ \quad \textit{when } g_k^{ij} + x_{2,k}^i < q_k^{ij*} + y_{2,k}^{i*} \textit{ and} \\ \quad g_k^{ij} < \tilde{q}_k^{ij*}(g_k^{ij}, x_{2,k}^i) & (\textit{Area } 1) \\ (q_k^{ij*}, y_{2,k}^{i*}) \\ \quad \textit{when } g_k^{ij} + x_{2,k}^i \geq q_k^{ij*} + y_{2,k}^{i*} \textit{ and} \\ \quad g_k^{ij} < q_k^{ij*} & (\textit{Area } 2) \\ (g_k^{ij}, y_{2,k}^{i*}(g_k^{ij})) \\ \quad \textit{when } g_k^{ij} \geq q_k^{ij*} \textit{ and } x_{2,k}^i \geq y_{2,k}^{i*}(g_k^{ij}) & (\textit{Area } 3) \\ (g_k^{ij}, x_{2,k}^i) \\ \quad \textit{when } g_k^{ij} \geq \tilde{q}_k^{ij*}(g_k^{ij}, x_{2,k}^i) \textit{ and} \\ \quad x_{2,k}^i < y_{2,k}^{i*}(g_k^{ij}), & (\textit{Area } 4) \end{cases}$$

where

$$(\tilde{q}_k^{ij*}(g_k^{ij}, x_{2,k}^i), \tilde{y}_{2,k}^{i*}(g_k^{ij}, x_{2,k}^i)) = \arg\max_{(q_k^{ij}, y_{2,k}^i)} \phi(q_k^{ij}, y_{2,k}^i) s.t.\ q_k^{ij} + y_{2,k}^i = g_k^{ij} + x_{2,k}^i;$$

$$(q_k^{ij*}, y_{2,k}^{i*}) = \arg\max_{(q_k^{ij}, y_{2,k}^i)} \phi(q_k^{ij}, y_{2,k}^i);$$

$$y_{2,k}^{i*}(g_k^{ij}) = \arg\max_{y_{2,k}^i} \phi(q_k^{ij}, y_{2,k}^i) s.t.\ q_k^{ij} = g_k^{ij}.$$

Then the $(q_k^{ij}, y_{2,k}^i)$-policy is optimal.

The structure of the $(q_k^{ij}, y_{2,k}^i)$-policy is shown in Figure 3. Figure 3(b) represents a special case when type-i order is the best fit for type-j server which is in turn the best fit for type-i order. In this case, orders in the acceptance queue must be accepted and immediately assigned to servers until no server or order is available and after all available servers are assigned. If there are more orders than necessary left in the acceptance queue $(y_{2,k}^{i*}(s^{ij}))$, they are rejected. Figure 3(a) represents the other general cases. q_k^{ij*} and $y_{2,k}^{i*}$ denote the optimal numbers of orders to be assigned to servers and in the acceptance queue, respectively. As shown in Figure 3(a), this optimum may be an internal point. $(\tilde{q}_k^{ij*}(g_k^{ij}, x_{2,k}^i), \tilde{y}_{2,k}^{i*}(g_k^{ij}, x_{2,k}^i))$, the borderline between areas 1 and 2, represents the optimal numbers of orders to be assigned to servers and kept in the acceptance queue, respectively, given the number of orders in the assignment queue and assigned to servers is $g_k^{ij} + y_{2,k}^i$. The optimal policy is characterized as follows. When $g_k^{ij} < q_k^{ij*}$, there are two cases: If $x_{2,k}^i + g_k^{ij} \geq y_{2,k}^{i*} + q_k^{ij*}$ (area 2), there are sufficient orders to reach the optimum $(q_k^{ij*}, y_{2,k}^{i*})$; otherwise, the optimal decision is to stay put (area 4) or accept and immediately assign more orders (area 1) depending on $(\tilde{q}_k^{ij*}(g_k^{ij}, x_{2,k}^i), \tilde{y}_{2,k}^{i*}(g_k^{ij}, x_{2,k}^i))$. When $g_k^{ij} \geq q_k^{ij*}$, there are three cases depending on $x_{2,k}^i$. First, if $x_{2,k}^i \geq y_{2,k}^{i*}(g_k^{ij})$ (area 3), there are more than necessary orders waiting. Orders are to be rejected. Second, if $x_{2,k}^i < y_{2,k}^{i*}(g_k^{ij})$ and $x_{2,k}^i < \tilde{y}_{2,k}^{i*}(g_k^{ij}, x_{2,k}^i)$ (area 4), having more orders in the acceptance queue, even rather than having them assigned to servers, is better; therefore, stay put is the best. Third, if $\tilde{y}_{2,k}^{i*}(g_k^{ij}, x_{2,k}^i) \leq x_{2,k}^i < y_{2,k}^{i*}(g_k^{ij})$ (area 1), more orders are to be accepted and immediately assigned until $\tilde{q}_k^{ij*}(g_k^{ij}, x_{2,k}^i)$ orders are processed by servers. This implies type-j server may be the best

or better server for type-i order (although type-i order is not the best fit for type-j server) so that the decision maker preempts remaining type-j servers for type-i orders.

Theorems 1 and 3 provide the following properties of the $(q_k^{ij}, y_{2,k}^i)$-policy.

(1) *For any* $\varepsilon \geq 0$, *there exists some* $\delta \geq 0$ *such that if* $\mathbf{x}_k = (q_k^{ij*} + \varepsilon, y_{2,k}^{i*})$, *then* $\mathbf{y}_k^*(\mathbf{x}_k) = (q_k^{ij*} + \varepsilon, y_{2,k}^{i*} - \delta)$ *and* δ *increases in* ε.

(2) *For any* $\varepsilon \geq 0$ *and* $\varepsilon' \geq 0$, *there exists some* $\vartheta \geq 0$ *such that if* $\mathbf{x}_k = (q_k^{ij*} - \varepsilon, y_{2,k}^{i*} - \varepsilon')$, *then* $\mathbf{y}_k^*(\mathbf{x}_k) = (q_k^{ij*} - \varepsilon + \vartheta, y_{2,k}^{i*} - \varepsilon' - \vartheta)$.

When there are ε more orders than the optimal level being assigned to servers (q_k^{ij*}), the decision maker should have δ fewer orders than the optimal level in the acceptance queue $(y_{2,k}^{i*})$ to decrease the waiting penalty (1). Submodularity (Theorem 3) ensures that δ increases in ε. However, because $\phi_k(q_k^{ij}, y_{2,k}^i)$ does not have the directional property, Lemma 5 is not applicable to the $(q_k^{ij}, y_{2,k}^i)$-policy. Thus, unlike the $(y_{1,k}^i, y_{2,k}^i)$-policy, $\delta \geq \varepsilon$ is not guaranteed in the $(q_k^{ij}, y_{2,k}^i)$-policy. δ may be greater or smaller than ε depending on the problem parameters. For example, if the service rate is high and the waiting penalty is low, δ may be smaller than ε. When there are fewer orders than the optimal levels both in the acceptance queue and servers, orders are not rejected for sure, and in some cases orders should be assigned immediately after being accepted. Thus, $\vartheta \geq 0$. However, again unlike the $(y_{1,k}^i, y_{2,k}^i)$-policy, because of a lack of the directional property, increasing monotonicity of ϑ in $\varepsilon + \varepsilon'$ is not guaranteed here (2).

5.2. Structure of the Overall Optimal Policy for Special Cases

Now we discuss the structure of the overall optimal policy in the following two special cases where the entire optimal policy is fully characterized using the two-dimensional optimal policies we find in Section 5.1.

5.2.1. Single Type of Order and Server $(I = J = 1)$**.** The optimal policy is specified in Proposition 12. For simplicity, we do not use superscripts i and j.

Proposition 12. *Let the* $(y_{1,k}, y_{2,k}, q_k)$*-policy be as follows*:

$$(y_{1,k}, y_{2,k}, q_k) = \begin{cases} (0, 0, x_{2,k} + x_{1,k} + g_k) \\ \quad \textit{when } x_{1,k} + x_{2,k} \leq \hat{S} \\ (P_{y_1,y_2,S}(0, x_{2,k} - S + x_{1,k} + g_k), S) \\ \quad \textit{when } x_{1,k} + x_{2,k} > \hat{S} \textit{ and } x_{1,k} < \hat{S} \\ (P_{y_1,y_2,S}(x_{1,k} - S + g_k, x_{2,k}), S) \\ \quad \textit{when } x_{1,k} + x_{2,k} > \hat{S} \textit{ and } x_{1,k} \geq \hat{S}, \end{cases}$$

where $\hat{S} = S - g_k$; $P_{y_1,y_2,S}(x_{1,k}, x_{2,k})$ *represents the post-decision state of* $(x_{1,k}, x_{2,k})$ *made by the* $(y_{1,k}, y_{2,k})$*-policy at* $q_k = S$. *Then the* $(y_{1,k}, y_{2,k}, g_k)$*-policy is optimal for MSSPAA with* $I = J = 1$.

Since there is only one type of server, there is no reason to postpone assigning any order. However, acceptance postponement is necessary to hedge the risk of having too many committed orders or too few orders in the queues to assign in subsequent periods. Since there is no assignment postponement, orders are assigned to servers until no server remains. Orders are assigned from the assignment queue first and then from the acceptance queue after all orders in the assignment queue are assigned. If any orders remain in the queues after all servers are assigned, some may be accepted or rejected, or their acceptance decisions may be postponed, according to the $(y_{1,k}, y_{2,k})$-policy at $g_k = S$. Thus, the overall optimal policy for MSSPAA with $I = J = 1$ is collapsed into one two-dimensional $(y_{1,k}, y_{2,k})$-policy.

5.2.2. Multiple Types of Server and Single Type of Order $(I = 1$ **and** $J > 1)$**.** Now we expand our discussion to the case when $I = 1$ and $J > 1$. The overall optimal policy still has a simple form. For simplicity, we do not use superscript i and assume $\bar{L}^j \leq \bar{L}^{j+1}$, $j = 1, 2, \ldots, J-1$.

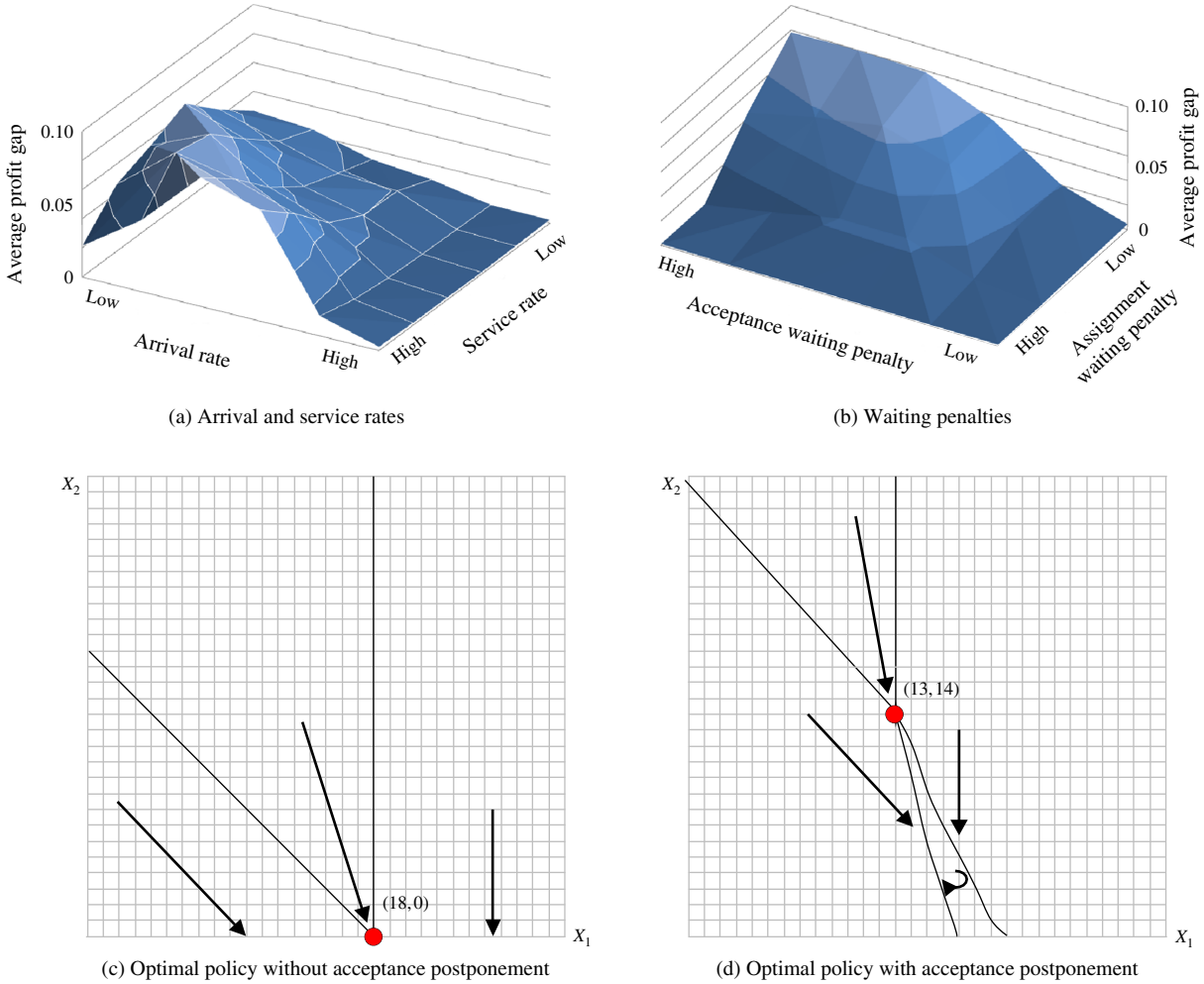
Proposition 13. *Let the* $(y_{1,k}, y_{2,k}, q_k^1, q_k^2, \ldots, q_k^J)$*-policy be as follows*:

$$(y_{1,k}, y_{2,k}, q_k^1, q_k^2, \ldots, q_k^J) = \begin{cases} (0, 0, S^1, S^2, \ldots, S^l, (g_k^{l+1} + x_{1,k} + x_{2,k} - \hat{s}^l), \\ \quad g_k^{l+2}, \ldots, g^J) \\ \quad \textit{when } \hat{s}^l \leq x_{1,k} + x_{2,k} < \hat{s}^{l+1} \textit{ and } l < J \\ (P_{y_1,y_2,S}(0, x_{2,k} - \hat{s}^J + x_{1,k}), S^1, S^2, \ldots, S^J) \\ \quad \textit{when } x_{1,k} + x_{2,k} > \hat{s}^J \textit{ and } x_{1,k} < \hat{s}^J \\ (P_{y_1,y_2,S}(x_{1,k} - \hat{s}^J, x_{2,k}), S^1, S^2, \ldots, S^J) \\ \quad \textit{when } x_{1,k} + x_{2,k} > \hat{s}^J \textit{ and } x_{1,k} \geq \hat{s}^J, \end{cases}$$

where $\hat{s}^l = \sum_{j=1}^{l} (S^j - g_k^j)$; $P_{y_1,y_2,S}(x_{1,k}, x_{2,k})$ *represents the post-decision state of* $(x_{1,k}, x_{2,k})$ *made by the* $(y_{1,k}, y_{2,k})$*-policy at* $q_k^1 = S^1, q_k^2 = S^2, \ldots, q_k^J = S^J$. *Then the* $(y_{1,k}, y_{2,k}, q_k^1, q_k^2, \ldots, q_k^J)$*-policy is optimal for MSSPAA with* $I = 1$ *and* $J > 1$.

The intuition of the $(y_{1,k}, y_{2,k}, q_k^1, q_k^2, \ldots, q_k^J)$-policy is similar to that of the $(y_{1,k}, y_{2,k}, q_k)$-policy. Since there is only one type of order, every order is the best fit for every type of server. Therefore, there is no reason to postpone assignment. Servers are assigned in an ascending rank of i (in other words, a descending rank of the server's average service rates, $1 - \bar{L}^j$) until no server remains. Since there is no assignment postponement, orders are assigned sequentially from the assignment queue and the acceptance queue. If any order remains in the queues after all servers are

Figure 4 (Color online) Advantages of Acceptance Postponement



assigned ($x_{1,k} + x_{2,k} > \hat{s}^j$), they may be accepted or rejected, or their acceptance decisions may be postponed, according to the $(y_{1,k}, y_{2,k})$-policy at $q_k^j = S^j$ for all j. Again, the overall optimal policy is reduced to one two-dimensional $(y_{1,k}, y_{2,k})$-policy.

6. Advantages of Acceptance Postponement: A Numerical Study

Similar concepts to assignment postponement are considered in related studies, including call center routing problems and inventory rationing problems, although these studies do not consider a separate acceptance process. What is totally new to the literature is acceptance postponement. We show the advantages of acceptance postponement and how they are affected by the problem parameters, by comparing the profits generated optimally with acceptance postponement and without it. We use the special case of single type order and server ($I = J = 1$) for this comparison. This case highlights the effects of acceptance postponement since it has no postponement assignment decisions.

In MSSPAA, some random processes are affected by decisions being made dynamically. For example, the service completion process is affected by the assignment decision. Thus, it is difficult to test the same problem instance across multiple policies. Similar to Akçay et al. (2010), we define the problem scenarios, each of which has the same values for the problem parameters, generate one million problem instances for each policy per scenario, compute the average profit that the policy generates over the instances, and compare the average profit across different policies. In the experiment, we use discrete values to represent the states and decisions. Although some random transitions, including service completion and order abandonment, generate real values, we use the closest integers by rounding the real values. To implement random transitions, we randomly generate the actual values each time (e.g., per period, type, and instance) according to certain probability distributions of which the parameter varies across the problem scenarios. We test a wide range of 5,158 problem scenarios by varying the values for the parameters such as the average

arrival, abandonment, and service rates as well as the waiting penalty.

The profit gap for each scenario is defined as (average profit generated by the optimal policy with acceptance postponement—average profit generated by the optimal policy without acceptance postponement)/average profit generated by the optimal policy without acceptance postponement. The maximal profit gap among the tested scenarios is 34.95%, and the average profit gap is 4.45%, which implies that postponement acceptance generates more profits.

To further investigate when acceptance postponement matters more, we conduct a sensitivity analysis. Figure 4(a) depicts how acceptance postponement creates the most benefit when traffic is moderate (the arrival rate and the service rate are moderate). In both high (the arrival rate is high, while the service rate is low) and low traffics (the arrival rate is low while the service rate is high), the benefit of acceptance postponement is diminished. In high traffic, there are enough new orders to accept and assign at every period so that all unaccepted orders simply can be rejected. In low traffic, there is a lack of orders so that every order that arrives is likely to be accepted and assigned. In both cases, the acceptance decision becomes less complicated because there is almost no need to postpone the acceptance decision. Figure 4(b) illustrates how acceptance postponement matters more as the ratio of the waiting penalty in the acceptance queue to that in the assignment queue decreases. As the ratio is lower, orders may be kept in the acceptance queue instead of being accepted in order to avoid paying more penalties. Figure 4(b) shows how acceptance postponement matters more when the ratio of waiting penalty to the revenue is moderate. If the waiting penalty is high, we simply do not store orders in the queues. If the waiting penalty is low, we accept and keep orders so as not to lose them. Since it is intuitive, the following is not shown in Figure 4: when the abandonment rate gets lower, the benefit of acceptance postponement is maximized. As customers are more willing to wait, we can postpone more orders and get more benefits.

Figures 4(c) and 4(d) depict the real optimal policies for the acceptance decision for a single order and server type problem ($p = 100; b_1 = 10; b_2 = 2; \bar{Z} = 0.1; \bar{W} = 0.5; \bar{L} = 0.15; n = 40; k = 1$) when acceptance postponement is ignored and considered, respectively. This example shows the optimal policy for the illustrative example used to motivate the problem in Section 1. They illustrate by postponing acceptance, the decision maker can have a higher availability of orders in subsequent periods while paying less waiting penalty. For example, the expected number of remaining orders in a subsequent period of state $(13, 14)$, the optimum when acceptance postponement is considered, is 20 $(= 13 + 14 \times 0.5)$, while state $(18, 0)$, the optimum when acceptance postponement is ignored, is 18. The waiting penalty for the focal period of $(13, 14)$ is 158 $(= 13 \times 10 + 14 \times 2)$ while that of $(18, 0)$ is 180 $(= 18 \times 10)$. If acceptance postponement is ignored, $(13, 14)$ is moved to $(18, 0)$ by accepting and rejecting orders losing profit.

7. Concluding Remarks

We introduce a new dynamic order acceptance and resource assignment problem of a make-to-order manufacturing or service system with heterogeneous orders and resources that can strategically postpone acceptance and assignment decisions (MSSPAA). We formulate MSSPAA as a stochastic dynamic program and characterize the structure of the optimal policy in a number of two-dimensional spaces by analyzing the concavity, submodularity, and directional property. We also characterize the structure of the optimal policy in the original multidimensional space by discussing some special cases where the entire optimal policy is collapsed into a few two-dimensional optimal policies. We also identify when acceptance postponement creates benefits by conducting an extensive numerical experiment.

This study contributes to the literature by introducing and formulating a new problem category, characterizing the optimal policy, and providing new managerial insights. Similar problems, such as call center routing, inventory rationing, stochastic knapsack, and stochastic secretary problems, have been studied. They differ, however, in that postponable acceptance and assignment are not allowed (stochastic knapsack/secretary) and acceptance and assignment decisions are not separated (call center routing and inventory rationing). To the best of our knowledge, there has not been any prior study that has considered acceptance postponement when acceptance and assignment decisions are separated.

The paper provides managerial insights on how to accept, reject, assign orders, or postpone them. Our results suggest that the decision maker should increase the number of orders in the acceptance queue up to a certain point (say α) and decrease the number of orders in the assignment queue down to a certain point (say β) as the total number of available orders in both queues increase. The decision maker should always keep the number of orders in both queues no more than $\alpha + \beta$ by rejecting some orders if necessary. By doing so, the decision maker can keep a certain level of availability of orders while keeping down the waiting penalty. The results also suggest that the decision maker should increase the number of orders being assigned to servers and decrease the

number of orders waiting in the assignment queue as the total number of orders assigned to servers and in the assignment queue increases. The decision maker can better match orders and servers while keeping a certain level of utilization and waiting penalty. The experimental results suggest that acceptance postponement creates more benefits when the traffic is moderate, the waiting penalty in the assignment queue is large compared to that in the acceptance queue but moderate compared to the revenue, and the abandonment rate is low.

This study provides a methodological contribution to the literature, too. We derive some novel general properties that may be reused in other future operations management studies. We find unique structural properties (Lemmas 5 and 6) when a two-dimensional function has concavity, submodularity, and the directional property (i.e., superconvexity in its second element and subconcavity in its first element). Concavity (or convexity), submodularity (or supermodularity), superconvexity (or subconvexity), superconcavity (or subconcavity), and diagonal dominance (or multimodularity) have been used in prior studies to find the structural properties of optimal policies. Each of them has been well studied, and some of their combined effects have been examined, too (Altman et al. 2000, Zhuang and Li 2012, Ha 1997, Yang and Qin 2007, Zhao et al. 2008, Morton 2006). However, to the best of our knowledge, no prior study has examined the combined case that we examine in this paper. General structural properties, such as submodularity, subconcavity, and superconvexity, are not preserved under maximization and profit-to-go expectation in general (Chao et al. 2009, Li and Yu 2014). This study finds new preservation conditions of submodularity under maximization (Lemma 2) and a preservation condition of the new structural results under a certain type of profit-to-go expectation (Lemma 8).

One of the limitations of the paper is the use of multiplicative random variables for the order abandonment and service completion processes. We chose this method because it is appropriate to our problem, provides technical convenience, and has been widely used to represent random processes in the related literature (Chao et al. 2009, Huh et al. 2013, Petruzzi and Dada 1999, Talluri and van Ryzin 2005). However, based on the results of the preservation of submodularity and the directional property under our random transition, we believe alternative random processes, including the binomial process and additive random variables, do not change our main technical results and the insight extracted from them. Another related limitation is that we use a continuous state space in the analysis although many practical MSSPAA problems would naturally be represented in a discrete state space. However, using a discrete state space usually makes problems intractable to analyze; therefore, prior studies have used the continuous space approximation for complex problems (Chao et al. 2009, Huh et al. 2013, Talluri and van Ryzin 2005). The complex state space of MSSPAA necessitates the continuous space approximation. Next, we characterize the optimal policy for some special cases in the multidimensional original state space; however, most of our results about the structure of the optimal policy are characterized in two-dimensional spaces. Although the two-dimensional optimal policies that we find provide significant managerial implications as well as practical benefits, we, as in similar studies (e.g., Zhuang and Li 2012), acknowledge the difficulty in characterizing an optimal policy in multidimensional space. We note that the structural properties found for the optimal policy often lead to efficient methods to solve the problem. Further research on an efficient solution is necessary.

Supplemental Material

Supplemental material to this paper is available at http://dx.doi.org/10.1287/msom.2016.0581.

Appendix: Summary of Notations

Notation	Description
I, i	Number of order types, order type.
J, j	Number of server types, server type.
n, k	Number of periods, period (or time at which period k begins).
S^j	Number of type-j servers.
s_k^{ij}	Maximum number of type-j servers which can be assigned to type-i orders at time k.
Q	Maximum number of orders per queue.
$x_{1,k}^i, x_{2,k}^i$	Number of type-i orders in the assignment queue and the acceptance queue at time k before making decisions.
g_k^{ij}	Number of type-j servers serving type-i orders at time k before making decisions.
a_k^i, r_k^i	Number of orders being accepted and rejected at time k.
u_k^{ij}	Number of type-i orders being newly assigned to type-j servers at time k.
$y_{1,k}^i, y_{2,k}^i$	Number of type-i orders in the assignment queue and the acceptance queue at time k after making decisions.
q_k^{ij}	Number of type-j servers serving type-i orders at time k after making decisions.
L_k^{ij}	Radom fraction variable with mean $\bar{L}^{ij}$. $1 - L_k^{ij}$ is the service rate.
W_k^i	Radom fraction variable with mean $\bar{W}^i$. $1 - W_k^i$ is the abandonment rate.
Z_k^i	Random variable with mean $\bar{Z}^i$. Number of type-i orders that newly arrive at the end of period k.
p^i	Revenue earned when a type-i order is assigned to a server.
b_1^i, b_2^i	Waiting penalty for a type-i order in the assignment and acceptance queue for a period.

References

Ahn H, Righter R, Shanthikumar JG (2005) Staffing decisions for heterogeneous workers with turnover. *Math. Methods Oper. Res.* 62(3):499–514.

Akçay Y, Balakrishnan A, Xu SH (2010) Dynamic assignment of flexible service resources. *Production Oper. Management* 19(3): 279–304.

Altman E, Gaujal B, Hordijk A (2000) Multimodularity, convexity, and optimization properties. *Math. Oper. Res.* 25(2):324–347.

Bassamboo A, Harrison JM, Zeevi A (2005) Dynamic routing and admission control in high-volume service systems: Asymptotic analysis via multi-scale fluid limits. *Queuing Systems* 51(3–4): 249–285.

Chao X, Chen H, and Zheng S (2009) Dynamic capacity expansion for a service firm with capacity deterioration and supply uncertainty. *Oper. Res.* 57(1):82–93.

Deshpande V, Cohen MA, and Donohue K (2003) A threshold inventory rationing policy for service-differentiated demand classes. *Management Sci.* 49(6):683–703.

Duran S, Liu T, Simchi-Levi D, Swann JL (2008) Policies utilizing tactical inventory for service-differentiated customers. *Oper. Res. Lett.* 36(2):259–264.

Ferguson TS (1989) Who solved the secretary problem? *Statist. Sci.* 4(3):282–289.

Gao L, Xu SH, Ball MO (2012) Managing an available-to-promise assembly system with dynamic short-term pseudo-order forecast. *Management Sci.* 58(4):770–790.

Gupta D, Wang L (2007) Capacity management for contract manufacturing. *Oper. Res.* 55(2):367–377.

Ha AY (1997) Optimal dynamic scheduling policy for a make-to-stock production system. *Oper. Res.* 45(1):42–53.

Huh WT, Liu N, Truong V (2013) Multiresource allocation scheduling in dynamic environments. *Manufacturing Service Oper. Management* 15(2):280–291.

Kleywegt AJ, Papastavrou JD (1998) The dynamic and stochastic knapsack problem. *Oper. Res.* 46(1):17–35.

Li Q, and Yu P (2014) Multimodualrity and its applications in three stochastic dynamic inventory problems. *Manufacturing Service Oper. Management* 16(3):455–463.

Morton A (2006) Structural properties of network revenue management models: An economic perspective. *Naval Res. Logist.* 53(8):748–760.

Petruzzi NC, Dada M (1999) Pricing and the newsvendor problem: A review with extensions. *Oper. Res.* 47(2):183–194.

Powell WB (2007) *Approximate Dynamic Programming: Solving the Curses of Dimensionality* (Wiley-Interscience, Hoboken, NJ).

Ross KW, Tsang DHK (1989) The stochastic knapsack problem. *IEEE Tran. Comm.* 37(7):740–747.

Ross SM (1983) *Introduction to Stochastic Dynamic Programming* (Academic Press, New York).

Shaked M, Shanthikumar JG (1994) *Stochastic Orders and Their Applications* (Academic Press, San Diego).

Talluri KT, van Ryzin GJ (2005) *The Theory and Practice of Revenue Management* (Kluwer, New York).

Topkis DM (1998) *Supermodularity and Complementarity* (Princeton University Press, Princeton, NJ).

Van Mieghem JA (1995) Dynamic scheduling with convex delay costs: The generalized $c\mu$ rule. *Ann. Appl. Probab.* 5(3):809–833.

Van Slyke R, Young Y (2000) Finite horizon stochastic knapsacks with applications to yield management. *Oper. Res.* 48(1): 155–172.

Wang H, Yan H (2009) Inventory management for customers with alternative lead times. *Production Oper. Management* 18(6): 705–720.

Wang H, Liang X, Sethi S, and Yan H (2014) Inventory commitment and prioritized backlogging clearance with alternative delivery lead times. *Production Oper. Management* 23(7):1227–1242.

Xu SH, Righter R, and Shanthikumar JG (1992) Optimal dynamic assignment of customers to heterogeneous servers in parallel. *Oper. Res.* 40(6):1126–1138.

Yang J, Qin Z (2007) Capacitated production control with virtual lateral transshipments. *Oper. Res.* 55(6):1104–1119.

Zhao H, Ryan JK, Deshpande V (2008) Optimal dynamic production and inventory transshipment policies for a two-location make-to-stock system. *Oper. Res.* 56(2):400–410.

Zhuang W, Li MZF (2012) Monotone optimal control for a class of Markov decision processes. *Eur. J. Oper. Res.* 217(2):342–350.