DBS-25-02

「Why and How Seru Production Systems Are Responsive and Efficient in Volatile Markets」

Dongni Li / Co-corresponding Author, Beijing Institute of Technology
Kathryn E. Stecke / University of Texas at Dallas
Yong Yin † / † Corresponding Author, Doshisha University
Kan Fang / Tianjin University
Hongbo Jin / Beijing Institute of Technology
Ikou Kaku / Tokyo City University

June, 2025

Dongni Li
Co-corresponding Author, School of Computer Science, Beijing Institute of Technology, Beijing 100081, China, ldn@bit.edu.cn
Kathryn E. Stecke
Naveen Jindal School of Management, University of Texas at Dallas, P.O. Box 830688, SM 30, Richardson, Texas 75083, U.S., kstecke@utdallas.edu
Yong Yin †
† Corresponding Author. Graduate School of Business, Doshisha University, Karasuma Imadegawa, Kamigyo-ku, Kyoto 602-8580, Japan, yyin@mail.doshisha.ac.jp
Kan Fang
College of Management and Economics, Tianjin University; Laboratory of Computation and Analytics of Complex Management Systems (CACMS), Tianjin University, Tianjin 300072, China, kfang@tju.edu.cn
Hongbo Jin
School of Computer Science, Beijing Institute of Technology, Beijing 100081, China, hb@bit.edu.cn
Ikou Kaku
Faculty of Environmental and Information Studies, Tokyo City University, Japan, kakuikou@tcu.ac.jp

Abstract

Seru production systems have demonstrated excellent rapid response capabilities in both stable and uncertain environments. This study reveals that when compared to the Toyota Production System, the seru production method improves rapid response capabilities by 20% and 50% in stable and uncertain environments, respectively. The underlying reasons for this improvement were unclear, so this became a focus of this paper. Static and dynamic Just-In-Time Organization Systems are used to investigate both the flexibility and efficiency of seru production systems under stable and uncertain conditions. Findings show that the rapid response capability of a seru system is driven by the substitution effect of parallel serus. Efficiency is relatively easy to achieve in stable environments but is more challenging in unstable conditions. Therefore, this study explored methods to achieve high efficiency in seru systems under uncertain environments. A stochastic gradient algorithm and a dynamic allocation algorithm are proposed. Experimental results demonstrate that the proposed methods outperform traditional newsvendor models and can achieve near-optimal performance.

本論文は、4 回にわたる厳しい査読を経て、ようやく採択が決まりました。採択されるのは、経営学分野のトップジャーナルである Production and Operations Management（UTD 24、FT 50）です。今回、同志社大学より学術サバティカルをいただいたおかげで、三名の査読者からの四度にわたるコメントに集中して対応することができました。このような機会をいただいたことに、心より感謝申し上げます。

(This paper was supported by KAKEN 25K08175.)

Why and How *Seru* Production Systems Are Responsive and Efficient in Volatile Markets

Dongni Li
Co-corresponding Author, School of Computer Science, Beijing Institute of Technology, Beijing 100081, China, ldn@bit.edu.cn

Kathryn E. Stecke
Naveen Jindal School of Management, University of Texas at Dallas, P.O. Box 830688, SM 30, Richardson, Texas 75083, U.S., kstecke@utdallas.edu

Yong Yin†
† Corresponding Author. Graduate School of Business, Doshisha University, Karasuma Imadegawa, Kamigyo-ku, Kyoto 602-8580, Japan, yyin@mail.doshisha.ac.jp

Kan Fang
College of Management and Economics, Tianjin University; Laboratory of Computation and Analytics of Complex Management Systems (CACMS), Tianjin University, Tianjin 300072, China, kfang@tju.edu.cn

Hongbo Jin
School of Computer Science, Beijing Institute of Technology, Beijing 100081, China, hb@bit.edu.cn

Ikou Kaku
Faculty of Environmental and Information Studies, Tokyo City University, Japan, kakuikou@tcu.ac.jp

Seru production systems have demonstrated excellent rapid response capabilities in both stable and uncertain environments. This study reveals that when compared to the Toyota Production System, the *seru* production method improves rapid response capabilities by 20% and 50% in stable and uncertain environments, respectively. The underlying reasons for this improvement were unclear, so this became a focus of this paper. Static and dynamic Just-In-Time Organization Systems are used to investigate both the flexibility and efficiency of *seru* production systems under stable and uncertain conditions. Findings show that the rapid response capability of a *seru* system is driven by the substitution effect of parallel *serus*. Efficiency is relatively easy to achieve in stable environments but is more challenging in unstable conditions. Therefore, this study explored methods to achieve high efficiency in *seru* systems under uncertain environments. A stochastic gradient algorithm and a dynamic allocation algorithm are proposed. Experimental results demonstrate that the proposed methods outperform traditional newsvendor models and can achieve near-optimal performance.

Key words: just-in-time, flexibility, assembly, Toyota Production System (TPS), lean.
History: This paper was first submitted in June 2023, revised in May and December 2024 and April 2025, and accepted on 5 June, 2025 by Panos Kouvelis.

1. Introduction

Taiichi Ohno, the architect of the TPS, identified two pillars underpinning TPS: (1) autonomation and (2) Just-In-Time Material System (JIT-MS). JIT-MS encompasses two components: the *kanban* system and *heijunka* (level production). For assembly lines focused on a single product type, a *kanban* system suffices to manage inventory and optimize throughput (Mitra and Mitrani 1990, 1991, Tayur 1993). Conversely, mixed-model assembly lines, which produce various product types on the same line, necessitate the integration of both a *kanban* system and level production to efficiently control inventory (Boysen et al. 2009, 2022).

Achieving efficiency on a mixed-model line using *kanban* requires meticulous line balancing attempts across different models. This balancing process can entail the strategic development of a

joint precedence graph to represent an "average" model, effectively simplifying the mixed-model challenge into a single-model with uniform processing routes and average processing times. On the other hand, *heijunka* aims to regulate production rates over time, ensuring they match with the market's consumption patterns. *Heijunka* facilitates JIT material supply, reducing inventory levels and attempting to align production closely with actual market demand. However, volatile markets, characterized by frequent model changes, large fluctuations in demand, and short product life cycles (with frequent launch of new models), can challenge the principles of balancing and *heijunka*, making their implementation difficult. Let's delve into scenarios that highlight these challenges.

Balancing a mixed-model assembly line aims for each workstation having an equal workload to promote efficient flow across models. However, introducing a new model with unique or more complex assembly requirements disrupts this balance. For instance, a line adjusted for three models is unsettled by a new model needing specialized tasks, altering workstation capabilities, and extending assembly times, leading to bottlenecks. This change disrupts the line's balance, as existing setups cannot handle the new tasks without modifications, highlighting the challenge of maintaining equilibrium with new model introductions.

Level production tries to match production rates to demand, a challenge during demand fluctuations for different models. Consider a car plant producing a range of models on one line, with a schedule aligning closely with demand. An unexpected demand surge for compact cars, driven by fuel price increases, disrupts this balance. The line, designed for a diverse model mix, struggles to shift focus to compact cars, showing the operational limits in adapting to rapid demand changes. This leads to inefficiencies, such as surplus SUVs and insufficient compact cars, underlining the difficulty in sustaining level production amid changing demands.

These examples illustrate how a volatile market disrupts the functionality of a mixed-model production line, leading to delays in meeting market demands. Real-world instances have shown Toyota grappling with this issue, notably in 2016 when customers faced a six to eight-month wait for a new Prius, a situation that deteriorated with the 2021 Land Cruiser model, pushing wait times to over two years. These cases highlight the challenges Toyota's mixed-model lines face in achieving both efficiency and flexibility in volatile markets.

Market variability requires buffering strategies that include inventory, time, and capacity buffers (Hopp and Spearman 2011). Mixed-model systems, ideal for low variability environments, use *kanban* methods for WIP inventory control, aim for maximum capacity utilization through line balancing, and use level production to ensure consistent production rates for finished product inventory control. In essence, these buffers (inventory, time, and capacity) are manageable results of strategic decisions. However, as variability increases, the need for these buffers increases. If inventory levels are kept constant (by not changing the number of *kanbans*), the capacity buffer increases because of the reduction in capacity utilization in an unbalanced system. The reduction in capacity utilization leads to increased time buffers, which adversely affects customer service.

High variability makes it difficult for mixed-model systems to effectively manage all three buffers simultaneously. This sequence of effects demonstrates the interrelated nature of these buffering strategies in response to market variability.

The introduction of the *seru* production system was a response to these challenges of high variability, allowing the production of multiple product types on parallel assembly units, known as *serus*. A *seru* is a small, compact assembly line consisting of simple equipment (e.g., manual tools) and one or more cross-trained workers to assemble one or more product types (many *serus* are dedicated to a single product type). *Serus* can be constructed, modified, dismantled, and reconstructed quickly and frequently, mirroring the adaptability biological cells. This similarity inspired Japanese managers to coin the term *seru*, the Japanese word for a cellular organism, to denote its reconfigurable property.

The principal aim of employing *seru* systems is the rapid response to volatile markets. In contrast to the two-pillar foundation of the TPS, *seru* systems rest on two pillars: Reconfigurability and Just-In-Time Organization System (JIT-OS). Reconfigurability represents the system's physical adaptability, while JIT-OS embodies the managerial flexibility required for *seru* systems to address variability swiftly. JIT-OS ensures the availability of the necessary *serus*, at the right place and time, with adequate capacity (contrasting with TPS's JIT-MS, which focuses on material requirements). To accommodate new models or model changes, *serus* can be newly created or modified. Demand fluctuations are managed by altering the number of *serus* and/or the number of workers within *serus*, thereby minimally impacting the existing system. For example, when a *seru* is created/dismantled/modified, it does not influence other *serus*. In contrast, any change on a mixed-model line has evident influence to the line.

Considering the three types of buffers, TPS's JIT-MS emphasizes the "inventory of materials", with *kanban* and level production designed to regulate WIP and finished goods inventories. In contrast, inventory management is less problematic for *seru* systems, which usually practice one-piece-flow, eliminating the need for *kanban* and minimizing WIP. Level production is also less of a concern because the dedicated nature of most *serus* negates the need for level production. However, capacity becomes crucial in *seru* implementations. With minimal inventory levels and rapid delivery expectations (small time buffer), capacity should be strategically managed to buffer against variability, shifting the focus from inventory in TPS to capacity in *seru* systems.

The first English *seru* paper was Yin et al. (2008). Subsequent *seru* research attention includes Stecke et al. (2012) and Yin et al. (2017), which provide case analyses of *seru* practices. A recent *seru* tutorial can be found in Yin et al. (2025). de Treville et al. (2017) describe *seru* as a strategic response to rapid product turnover and high demand volatility, enabling competitiveness in Japan by prioritizing proximity to markets and R&D and shifting from automated machinery to smaller, versatile equipment. Roth et al. (2016) highlight *seru* as a more flexible advancement over TPS, signifying the next generation of lean manufacturing. Hopp and Spearman (2021) indicate that investigating leanness and agility (e.g., as exemplified by *seru* production) necessitates an

integrated approach, grounded in strategic intent and practical utility. Using statistical analysis, Bendoly et al. (2021) find that *seru* responsiveness has a significant positive impact on organizational performance, particularly in contexts where traditional lean thinking is less emphasized. Cohen et al. (2022) consider *seru* production as an exemplary model that demonstrates the potential for firms to significantly enhance their supply chain resilience by rethinking and challenging conventional constraints.

This paper contributes to the understanding of *seru* systems by addressing two gaps in the existing literature. First, while prior research has empirically demonstrated that *seru* systems offer flexibility and efficiency, the underlying reasons why this is the case have not been thoroughly explored. By employing analytical methods to examine the structural properties of *seru* systems, this study uncovers mechanisms that show their responsiveness and efficiency. Second, concerning how to effectively manage a *seru* system, previous studies, e.g., Zhang et al. (2022), Li et al. (2023), have predominantly focused on reducing lead times to enhance responsiveness with fixed *seru* capacity. To our knowledge, this is an initial effort that conceptualizes *seru* capacity as a decision variable, marking a shift in focus towards capacity buffers from the inventory buffers predominantly underscored in traditional production systems.

2. *Seru* Applications and Research Questions

The applicability and research questions of *seru* production systems are presented in this section. The electronics industry, in contrast to the automotive sector, experiences high volatility. During the 1990s, to manage such unpredictable conditions, Canon and Sony implemented the mixed-model approach of the TPS. These efforts did not meet the expected level of success, as documented in Stecke et al. (2012). Since mixed-model systems are suited to environments with low variability, they struggle to adapt to highly variable conditions. The shift to *seru* production systems endowed electronics companies with flexibility and rapid responsiveness to prosper.

2.1. *Seru* Applications

Following the success of *seru* in the electronics industry, other Japanese sectors, including those in low variability business environments, have also used *seru* systems, with reports of successful implementations. *Food Plant Manager*, a Japanese monthly magazine, dedicated a special issue (Editor 2011) showcasing the application of *seru* production in various food factories. Denso incorporates *seru* systems to produce maintenance parts for cars (Stecke et al. 2012). Nissan utilizes *seru* systems to assemble front-end modules, components in automobile assembly (Gai et al. 2023). Yamaha (2024) employs *seru* systems to manufacture engines and motorcycles. Wimo (2020) applies *seru* systems to bicycle production. Notably, these sectors are characterized by non-innovative products with long life cycles. For example, food items and maintenance parts typically sustain life cycles spanning decades, with relatively stable and predictable market conditions.

In high variability environments, *seru* systems demonstrate three distinct operational modes: regular, seasonal, and emergent. For regular production, the focus is on fulfilling the day-to-day demand for both off-the-shelf and customized products, with the primary goal being profitability.

While a high degree of responsiveness is necessary, it is not as critical as it is during seasonal and emergent periods. *Seru* responsiveness allows companies to selectively respond to customer demands as well as strategically relocate less profitable product lines. For example, since 2003, Canon has moved many low-profit production lines from China back to Japan (Takano 2005).

Seasonal production addresses the less predictable and significant fluctuations in demand tied to cultural and seasonal events. Products such as Daikin's air conditioners, which see heightened sales before and during summer, or items in high demand during the Golden Week, China's Double 11, or festive periods like Christmas, require a preemptive scaling of production. A survey (Wu et al. 2024) noting that seasonal demand can constitute the majority of annual sales for some products underscores the potential use of *seru* systems to balance profitability with responsiveness.

Emergent production is triggered by unexpected occurrences such as pandemics, earthquakes, and other disasters. The primary objective of emergent production is responsiveness and resilience, so that production is not disrupted by emergencies. Profit is less important but higher responsiveness is required. Most customers are local governments and non-profit organizations. Production is related to the survival of disaster-affected residents. An example is Nihon Kohden, a Japanese medical electronics company that used *serus* to produce ventilators for COVID-19.

The above production practices are summarized in Table 1.

Table 1 ***Seru* Practice in the Manufacturing Environments.**

	Low variability	**High variability**
A single line	TPS: mixed-model	N/A
Parallel *serus*	Variation	Regular, Seasonal, Emergent

Seru production principles have been applied successfully beyond the manufacturing domain. For instance, in the context of logistics, *seru* has enhanced the efficiency of order picking in warehouses (Gai et al. 2023). In the healthcare industry, the "*seru* nursing provision method" is used to streamline nursing tasks, improving efficiency and job satisfaction. This method reduces unnecessary practices, allowing nurses to devote more time to patient care and derive greater fulfillment from their work (Sudo 2019). In product development, Iris-Ohyama uses a *seru*-inspired method to assign a single developer the responsibility for the entire process from planning and market research to procurement, prototyping, design, cost calculation, pricing, and sales planning. This comprehensive approach has proven to yield short lead times (Mitamura 2012). Although such applications in non-manufacturing sectors are compelling, they fall outside the scope of this paper.

2.2. Managerial Decisions and Research Questions

Responsiveness is defined as the degree to which a production system can speedily and effectively fulfill customer demands. *Flexibility* refers to the capability of a system to adapt its operations in ways that enhance responsiveness. *Maximum Cost Efficiency* is the ability to maximize output while minimizing input, thereby maximizing overall profitability.

To evaluate responsiveness, completion time in this section and cycle time in Section 4 are used to assess a production system's speed. Response rate is defined in Section 4 to evaluate a system's

effectiveness in meeting customer demands. *Cost efficiency* is used in TPS and lean (Hopp and Spearman 2021). Cost efficiency focuses on maintaining constant output while minimizing input.

Seru practices outlined in Table 1 employ static (offline) and/or dynamic (online) JIT-OS for production planning. Static JIT-OS is used when orders are known in advance. Dynamic JIT-OS is applied when orders are unknown, requiring a *seru* system schedule incoming orders during the production period. Formal definitions of static and dynamic JIT-OS are provided below.

- **Static JIT-OS Problem:** *Given a set of serus and a roster of customer orders at the start of a production period, the objective is to determine the optimal capacity for each seru and to assign customer orders to maximize total profit.*
- **Dynamic JIT-OS Problem:** *Given a set of serus at the start of a production period, the capacity of each seru is determined to accommodate a stream of unknown incoming orders of various product types and volumes. When an order arrives, it will be assembled on one or more serus to maximize expected profit. If remaining capacity is insufficient to complete an order, the order is assembled to the greatest extent possible within the capacity limits, which implies that customers are willing to accept partial fulfillment of their orders. Remaining unsatisfied orders can be produced in subsequent periods with known orders using the static JIT-OS mode that scheduled the next period. Or they may become lost sales if customers no longer want them.* For example, seasonal and emergent customer orders are usually lost if they cannot be produced before a deadline such as the first day of the Golden Week.

The use of static or dynamic scheduling varies across different *seru* practices. In low variability contexts, where offline orders dominate, a static JIT-OS approach should be used. For example, delicatessen products sold in retail stores have short product lifecycles (one or two days). Food plants such as Ishii Foods use a static approach to plan tomorrow's production based on confirmed orders received from retailers today.

In high variability contexts, regular *seru* production involves a mix of offline and online orders. For instance, Omron (a Japanese electronics *seru* practitioner producing a variety of electronic products such as healthcare devices) provides a "within 24 hours delivery" service. For products to be delivered on Friday, Omron produces them on Thursday. Many customers place orders before Thursday (for Friday), which are known in advance, allowing Omron to use a static JIT-OS approach to plan Thursday's production. However, some customers place orders on Thursday, which are unknown in advance. Omron employs a dynamic JIT-OS approach to adapt to these incoming orders during Thursday's production.

Seasonal *seru* production is largely driven by online orders. For example, a *seru* practitioner producing cameras can organize a 3-day production period (say, April 19 to 21) before Golden Week (April 22 to 30) for holiday sales. Proactive retailers place orders before the 19th. These offline orders are managed using a static JIT-OS approach. Many retailers prefer to to place orders during the production period, as they have more accurate forecasts of customer demand closer to Golden Week. These online orders are managed using a dynamic JIT-OS approach.

Finally, emergent *seru* production operates almost exclusively with online orders. For example, a *seru* practitioner producing ventilators for COVID-19 faces a situation where both producers and customers can hardly make any forecasts. Therefore, almost all customer orders are online and managed using a dynamic JIT-OS approach.

So, sometimes both static and dynamic JIT-OS approaches are required. Regular and seasonal *seru* production involves a mix of static and dynamic JIT-OS problems. These problems can be addressed separately and sequentially. First, the static JIT-OS problem is solved. Then the dynamic JIT-OS problem is solved. Finally, the solutions are combined to achieve an integrated solution.

The following Theorem 1 states that the static JIT-OS problem can be optimized within polynomial time, thus enabling any standard optimization software to solve it efficiently. All proofs for Theorems are in Section EC.3 of the online Appendix.

THEOREM 1. *A general static JIT-OS problem can be optimized in polynomial-time to maximize output while minimizing input, maximizing overall profitability.*

Under the static JIT-OS mode, Theorem 1 shows that ***efficiency*** of a *seru* system can be achieved. This efficiency is useful because *seru* production periods are typically short (one day for Ishii and Omron). Delays in computation could lead to missed delivery windows, adversely affecting customer satisfaction and company reputation. In contrast, traditional production systems such as TPS are designed for longer production periods. Toyota's production period is around two months. Toyota can afford to spend several days on computation.

Flexibility can be achieved by using different *serus* to produce customer orders. If a *seru* cannot produce an order, another *seru* can substitute to fulfill it. Details are in the proof. As we show in Section 4.1, this substitution mechanism also operates in the dynamic JIT-OS mode. However, achieving efficiency in dynamic JIT-OS is not automatic.

Theorem 1 demonstrates that a *seru* can achieve cost efficiency. Next, we compare the responsiveness of a *seru* system with a TPS line under static mode using an example.

- **Static Version of Penny Fab One**

An example is used from **Factory Physics** (page 232), Penny Fab One, which operates as a balanced TPS line, for this comparison. The line consists of four sequential stations, each operated by a worker, with an assembly time of two hours allowed per station. We compare this TPS line with a *seru* system, assuming both are perfectly designed (i.e., a balanced TPS line and an optimized *seru* system, as outlined in Theorem 1).

The *seru* system has four *yatais*, each with one worker. A single worker assembles the entire product in each *yatai*, taking eight hours per product. Both the TPS line and the *seru* system have the same production rate of 1/2 product per hour. Consider a static JIT-OS scenario where a customer orders 12 products, known at the start of the production period.

With this complete information, the TPS line produces the 12 products in 30 hours. The first product is finished in 8 hours, the second and third in 10 and 12 hours, respectively, with the

final product completed in 30 hours. In contrast, the *seru* system produces 12 products in 24 hours. Four products are completed in 8 hours, and 8 and 12 products are finished in 16 and 24 hours, respectively. Thus, the *seru* system is more responsive than the TPS line, reducing batch completion time by 6 hours, 20% faster than the TPS line. We summarize these findings in Managerial Insight 1.

Managerial Insight 1: *In static stable environments, optimizing the efficiency of a seru system is straightforward. The enhanced responsiveness of a seru system is automatic because of its flexible structure, where multiple parallel serus (e.g., i, j, and others) can substitute for one another to handle orders. In terms of responsiveness, a seru system outperforms a TPS line by achieving shorter completion times for batch production.*

Sections 3, 4, 5, 6, and 7 discuss dynamic JIT-OS.

3. Dynamic JIT-OS

A dynamic JIT-OS problem incorporates two stages. Determine capacity for each *seru* at the beginning of a period and assign each order to one or more *serus* after it arrives.

Notation is in Table 2. Consider a *seru* system in which there are I *serus* with index $i = 1, ..., I$. The set of *serus* is defined as $\mathcal{I} = \{1, 2, \ldots, I\}$, $i \in \mathcal{I}$. A skill structure is a set that contains all types and ranges of worker skills that the *seru* system can use to perform tasks. Each worker has achieved a particular skill level that defines what tasks he/she can efficiently perform. Within a *seru* system, each *seru* has its own skill set, which is a subset of the skill structure. A skill vector is used to represent a skill set. The *seru* system has a set $\mathcal{J} = \{1, 2, \ldots, J\}$ of assembly skills. Each *seru* $i \in \mathcal{I}$ has its own skill vector $s_i = (s_i^1, \ldots, s_i^j, \ldots, s_i^J)$, where $s_i^j = 1$ if *seru* i has skill j, and 0 otherwise. $j \in \mathcal{J}$ and $j = 1, ..., J$ is the index of skills.

Products, especially those with modular architectures, can be described by their components, some of which satisfy customers' preferences (Swaminathan and Tayur 1998, Jiang et al. 2006, Mendelson and Parlaktürk 2008). Some studies assume that the relationship between a skill and a component is one-to-one (Swaminathan and Tayur 1998, Baldwin and Clark 2000). This means that any component, say j, can only be produced or assembled by using its associated skill j. Then j is the index for both skills and components. This paper follows the above assumption.

$\mathcal{O}$ is the set of orders. Each order $o \in \mathcal{O}$ is defined by a component vector, $v_o = (v_o^1, \ldots, v_o^j, \ldots, v_o^J)$, where $v_o^j = 1$ if order o requires component j, and 0 otherwise. The component vector has the same cardinality as the skill vector. It is easy to see that when *seru* i can assemble order o, then for any skill $j \in \mathcal{J}$, if $v_o^j = 1$, then $s_i^j = 1$. For simplicity, define $\alpha_{io} = 1$ if *seru* i can assemble order o, and 0 otherwise. $\mathcal{G}_o$ is the set of *serus* whose skills can assemble order o, $\mathcal{G}_o = \{i \in \mathcal{I} : \alpha_{io} = 1\}$. The demand for order o is d_o.

3.1. Profit Function

The objective is to maximize the profit of a *seru* system. Let $x = (x^1, \ldots, x^i, \ldots, x^I)$ be the initial *seru* system capacity at the beginning of a period. Capacity is a continuous variable (hours or

Table 2 Notation

Sets and Parameters	
$(\Omega, \mathcal{F}, P)$	Probability space of orders
$\mathcal{I}$	Set of *serus*, indexed by $i = 1, ..., I$
$\mathcal{J}$	Set of assembly skills/components, indexed by $j = 1, ..., J$
$\mathcal{O}$	Set of orders, indexed by o
s_i^j	Indicator parameter, which equals 1 if *seru* i has skill j, and 0 otherwise
v_o^j	Indicator parameter, which equals 1 if order o requires component j, and 0 otherwise
α_{io}	Indicator parameter, which equals 1 if *seru* i can assemble order o, and 0 otherwise
$\mathcal{G}_o$	Set of *serus* whose skills can assemble order o, $\mathcal{G}_o = \{i \in \mathcal{I} : \alpha_{io} = 1\}$
λ_o	Cardinality of set $\mathcal{G}_o$, $\lambda_o = \lvert\mathcal{G}_o\rvert$
d_o	Number of products required for order o
τ_o	Total processing time for one product of order o
a_o	Time interval between the arrival times of orders o and $o+1$
c_j	Labor cost of assembly skill j
p_j	Revenue from component j
s_i	Skill vector of *seru* i, $s_i = (s_i^1, \ldots, s_i^j, \ldots, s_i^J)$, where $s_i^j = 1$ if *seru* i has skill j, and 0 otherwise
v_o	Component vector of order o, $v_o = (v_o^1, \ldots, v_o^j, \ldots, v_o^J)$, where $v_o^j = 1$ if order o requires component j, and 0 otherwise
e_i	Labor cost of *seru* i, $e_i = \sum_{j=1}^{J} c_j s_i^j$
b_o	Revenue from one product of order o, $b_o = \sum_{j=1}^{J} v_o^j p_j$
$b_{\max}$	Highest revenue from one product of any order in set $\mathcal{O}$, $b_{\max} = \max\{b_o : o \in \mathcal{O}\}$.
$b_{\min}$	Lowest revenue from one product of any order in set $\mathcal{O}$, $b_{\min} = \min\{b_o : o \in \mathcal{O}\}$.
$m_{\max}$	Maximum profit margin achievable by any *seru* i, $m_{\max} = (b_{\max} - e_i)/b_{\max}$ where $i \in \mathcal{I}$.
$m_{\min}$	Minimum profit margin achievable by any *seru* i, $m_{\min} = (b_{\min} - e_i)/b_{\min}$ where $i \in \mathcal{I}$.
r_i	Rank assigned to *seru* i, which prioritizes the *serus* for an order
ω	Sample path of $(\Omega, \mathcal{F}, P)$, $\omega = \{v_o \in \Omega \lvert o \in \mathcal{O}\}$
$\mathbb{E}$	Expectation operator on the probability space $(\Omega, \mathcal{F}, P)$
$\mathbb{I}_M$	Indicator of mathematical statement M, which equals 1 if M is true, and 0 otherwise
$\widetilde{m}$	For any order o, the remaining capacity of *serus* whose ranks are higher than *seru* i, $\widetilde{m} = x_o^{[1]} + x_o^{[2]} + \cdots + x_o^{[r_i - 1]}$
m	For any order o, the remaining capacity of *seru* i and *serus* whose ranks are higher than *seru* i, $m = x_o^{[1]} + x_o^{[2]} + \cdots + x_o^{[r_i]} = \widetilde{m} + x_o^{[r_i]}$
β_j	Increase of the initial capacity of *seru* j, $\beta_j = \widehat{x}^j - x^j$ and $\widehat{x}^j \geq x^j$.
Decision Variables	
x^i	Initial capacity of *seru* i
x	Initial capacity of a *seru* system, $x = (x^1, x^2, \ldots, x^I)$
x_o	Remaining capacity of a *seru* system before the arrival of order o, $x_o = (x_o^1, x_o^2, \ldots, x_o^I)$, where x_o^i is the remaining capacity of *seru* i before the arrival of order o
Functions	
$q_o(x, \omega)$	Capacity consumed by order o on sample path ω, when initial capacity is x, $q_o(x, \omega) = (q_o^1(x, \omega), \ldots, q_o^i(x, \omega), \ldots, q_o^I(x, \omega))$, where $q_o^i(x, \omega)$ is the capacity of *seru* i that is consumed to assemble order o
$R(x, \omega)$	Profit produced by a *seru* system by sample path ω, when initial capacity is x
U_o	Remaining production requirements of order o
W_i	Remaining capacity of *seru* i
$P(o, x)$	Profit obtained by assembling order o under initial capacity x
$A(o, x)$	Set of *serus* with which order o is assembled under initial capacity x
cr	Competitive ratio, the ratio of the possible minimum profit to the possible maximum profit, $cr = \frac{m_{\min}}{m_{\max}} \times \frac{1 - m_{\max}}{1 - m_{\min}}$.

minutes). For example, for a 3-day period, if x^1=8 hours and x^2=72 hours, then *seru* 1 is closed after 8 hours and *seru* 2 operates during the entire period.

A *seru* system's cost is calculated using a skill-based labor cost system. Let c_j denote the labor cost for skill j (i.e., hourly wage). Then the labor cost of *seru* i is $e_i = \sum_{j=1}^{J} c_j s_i^j$. Let p_j denote the revenue from component j. Then the revenue from one product for order o is $b_o = \sum_{j=1}^{J} v_o^j p_j$.

Let $(\Omega, \mathcal{F}, P)$ denote a probability space of orders, where Ω is a sample space of orders. $\mathcal{F}$ and P are the σ-algebra and probability measure of this probability space, respectively. Let ω be a sample path of $(\Omega, \mathcal{F}, P)$. The number of orders on sample path ω is denoted as O. That is, there is a set $\mathcal{O} = \{1, 2, \ldots, O\}$ of orders and $\omega = \{v_o \in \Omega \lvert \forall o \in \mathcal{O}\}$. During a production period, orders with stochastic demands arrive over time. When there is not enough remaining capacity to entirely assemble a newly arrived order o within the production period, this order is assembled as much as possible until the remaining capacity of *serus* in $\mathcal{G}_o$ becomes 0.

Let $x_o = (x_o^1, \ldots, x_o^i, \ldots, x_o^I)$ denote the remaining capacity of the *seru* system before the arrival of order o. Obviously, $x_1^i = x^i$, i.e., the remaining capacity before the arrival of the first order is

the initial capacity. Define $q_o(x,\omega)=(q_o^1(x,\omega),\ldots,q_o^i(x,\omega),\ldots,q_o^I(x,\omega))$ as the capacity consumed by order o on sample path ω when the initial capacity of the *seru* system is x, where $q_o^i(x,\omega)$ is the capacity of *seru* i that is consumed to assemble order o. $R(x^i,\omega)=\sum_{o\in\mathcal{O}} b_o[q_o^i(x,\omega)/\tau_o]-e_i x^i$ is the profit function of *seru* i, where τ_o is the processing time for one product of order o. The profit function of the *seru* system is as follows.

$$R(x,\omega)=\sum_{i\in\mathcal{I}} R(x^i,\omega)=\sum_{i\in\mathcal{I}}\sum_{o\in\mathcal{O}} b_o[q_o^i(x,\omega)/\tau_o]-\sum_{i\in\mathcal{I}} e_i x^i. \tag{1}$$

Let $\mathbb{E}$ be the expectation operator defined on the probability space $(\Omega,\mathcal{F},P)$. Then the objective of the dynamic JIT-OS problem is to maximize expected profit: $\max_{x\geq 0}\mathbb{E}[R(x,\omega)]$.

3.2. Evolution of the Dynamic JIT-OS

A ranking system is used to assign orders to *serus*. Upon the arrival of a new order o, each *seru* i is given a rank r_i, arranging the *serus* in order of priority for handling order o. The *seru* capable of assembling order o with the lowest labor cost receives the top rank, 1, followed by the others in ascending order. Specifically, the *serus* in the set $\mathcal{G}_o$ are ordered from the lowest to the highest labor costs, producing a sequence $e_{[1]}\leq e_{[2]}\leq\cdots\leq e_{[\lambda_o]}$. Here, *seru* $[1]\in\mathcal{G}_o$ incurs the least cost and is thus prioritized with the highest rank. Conversely, *seru* $[\lambda_o]$ is ranked last with the rank number λ_o, which is the cardinality of set $\mathcal{G}_o$, $\lambda_o=|\mathcal{G}_o|$. The rank r_i is then defined as follows.

$$r_i=\begin{cases} k, & \text{if } \alpha_{io}=1, \text{ and } e_i=e_{[k]},\\ \lambda_o+1, & \text{if } \alpha_{io}=0.\end{cases} \tag{2}$$

On a sample path ω, a total of O orders arrive over time. As the period progresses, the evolution of the JIT-OS undergoes two related directions: a monotonic increase in the quantity of assembled products, $\sum_{o\in\mathcal{O}} q_o(x,\omega)$, and a monotonic decrease of *seru* capacity, x. In detail, the capacity of a *seru* system diminishes incrementally, from $x=x_1$ (the initial capacity prior to the first order's arrival) to x_2 (the capacity before the second order's arrival), and so on. This evolution continues until all orders are scheduled and the capacity of the *seru* system is x_{O+1} (the capacity after assembling order O). This evolution of capacity is calculated using x_{o+1}^i, the capacity of *seru* i before the arrival of order $o+1$, from x_o^i as follows.

$$x_{o+1}^i=\begin{cases}\mathbb{I}_{d_o\tau_o\leq a_o}(x_o^i-a_o)^+ +\mathbb{I}_{d_o\tau_o>a_o}(x_o^i-d_o\tau_o)^+, & r_i=1;\\ \mathbb{I}_{d_o\tau_o<\widetilde{m}}(x_o^i-a_o)^+ +\mathbb{I}_{\widetilde{m}\leq d_o\tau_o<m}(x_o^i-\max\{d_o\tau_o-\widetilde{m},a_o\})^+ +\mathbb{I}_{d_o\tau_o\geq m}0, & 1<r_i\leq\lambda_o;\\ (x_o^i-a_o)^+, & r_i=\lambda_o+1,\end{cases} \tag{3}$$

where $\mathbb{I}_M$ denotes the indicator of mathematical statement M, which equals 1 if M is true, and 0 otherwise, $(\cdot)^+$ is the nonnegative value, d_o is the number of products required for order o, a_o is the time interval between the arrival times of orders o and $o+1$, $\widetilde{m}$ is the total remaining capacity of *serus* with ranks higher than *seru* i ($\widetilde{m}=x_o^{[1]}+x_o^{[2]}+\cdots+x_o^{[r_i-1]}$), and m is the total remaining

capacity of *seru i* and the *serus* whose ranks are higher than *seru i* ($m = x_o^{[1]} + x_o^{[2]} + \cdots + x_o^{[r_i]} = \widetilde{m} + x_o^{[r_i]}$).

By the definition of $q_o^i(x,\omega)$ given in Section 3.1, $q_o^i(x,\omega)$ is calculated as follows.

$$q_o^i(x,\omega) = \begin{cases} \mathbb{I}_{d_o\tau_o < x_o^i} d_o\tau_o + \mathbb{I}_{d_o\tau_o \geq x_o^i} x_o^i, & r_i = 1; \\ \mathbb{I}_{d_o\tau_o < \widetilde{m}} 0 + \mathbb{I}_{\widetilde{m} \leq d_o\tau_o < m} (d_o\tau_o - \widetilde{m}) + \mathbb{I}_{d_o\tau_o \geq m} x_o^i, & 1 < r_i \leq \lambda_o; \\ 0, & r_i = \lambda_o + 1. \end{cases} \tag{4}$$

4. Flexibility and Efficiency of a *Seru* System

Diniminishing returns and substitution effect are key properties of submodular functions (Milgrom and Strulovici 2009). We elucidate the relationships between efficiency and diminishing returns, and flexibility and substitution in submodular functions using a simple example.

- **Efficiency and Diminishing Returns in Submodular Functions**

Adding a display to a PC enhances our computing experience. The cost of this display is low compared to the substantial benefit it adds, resulting in high *efficiency* and a strong benefit margin. However, if we add more displays—second, third, fourth, and so on—*efficiency* declines. Each additional display still offers benefits, such as better multitasking capabilities, but the incremental benefit becomes smaller over time. This *diminishing returns* means that each new display contributes less to the overall benefit than the previous display. The marginal benefit of adding another display may become smaller than its cost, leading to a situation where adding more displays results in negative profit. At this point, costs outweigh benefits, illustrating now *diminishing returns* in submodular functions directly impacts efficiency.

- **Flexibility Through Substitution in Submodular Functions**

Adding multiple displays to our PC increases the system's *flexibility*, since it can swiftly adapt to unexpected changes or failures, maintaining functionality and performance under uncertainty. With multiple displays, if one display malfunctions or becomes unavailable, others can seamlessly *substitute* for it, ensuring that the PC system can continue to operate without interruption. This ability to *substitute* one component for another without significantly degrading performance is a key benefit of *substitution* in submodular functions. Such flexibility is valuable in dynamic environments where reliability and adaptability are important.

- **Trade-off between Efficiency and Flexibility in Submodular Functions**

This example of adding multiple displays to a PC highlights a trade-off between flexibility and efficiency inherent in submodular functions. On one hand, increasing the number of displays enhances the system's flexibility to maintain functionality in uncertain or changing environments. On the other hand, as more displays are added, efficiency declines because of the diminishing returns, eventually leading to negative profits when costs surpass benefits. Decision-makers should balance efficiency and flexibility to ensure that systems remain both economically viable and capable of adapting to changing environments.

The flexibility and efficiency trade-off in a dynamic JIT-OS *seru* system is investigated using submodular theory. Flexibility refers to the ability to quickly respond to uncertain demands. Efficiency refers to meeting demands with minimal costs. This trade-off implies that a highly efficient *seru* system may lack the ability to respond to changing conditions. Or a highly flexible *seru* system may compromise efficiency to retain its adaptability. Hence, it is useful to find a balance between efficiency and flexibility that aligns with the goals of a *seru* system.

This trade-off, resulting from variability, can't be eliminated but can be managed through strategic decisions such as *seru* capacity and order assignment, as explored in this paper. We demonstrate this trade-off with two straightforward examples. In terms of capacity (comparable to inventory management where too much inventory may lead to excess stock and too little may result in lost sales), having too much capacity can lead to high costs from underutilization, while having too little can reduce flexibility.

- **Example 1: Highly Flexible versus Highly Efficient *Seru* Systems**

Consider a *seru* system consisting of two *serus*: *seru* 1 specializes in skill 1, and *seru* 2 in skills 1 and 2. The cost for and revenue from each skill and component are \$100 and \$120, respectively. Capacities of 1 and 19 are allocated to *serus* 1 and 2, respectively, for a highly flexible system, and 14 and 6 for a highly efficient system, maintaining total capacity at 20 for both systems. The flexible system allocates greater capacity to *seru* 2, which handles multiple skills, enhancing its adaptability. The efficient system prioritizes lower cost, with a total cost of \$2,600 compared to \$3,900 for the flexible system. During the period, two orders arrive: the first order needs component 1 with a volume of 10. The second order requires components 1 and 2, also with a volume of 10, making the total demand 20. The highly flexible system can fulfill all 20 units of demand at a loss of $(10 \times 120 + 10 \times 240) - (1 \times 100 + 19 \times 200) = -\$300$. The highly efficient system can only meet 16 units of demand, resulting in a profit of $(10 \times 120 + 6 \times 240) - (14 \times 100 + 6 \times 200) = \$40$. Thus, the highly flexible system achieves a 100% service level (highest responsiveness) but incurs a financial deficit (low efficiency). The highly efficient system sacrifices flexibility and potential sales, achieving a $16/20 = 80\%$ service level. Another metric, the competitive ratio, is now defined as the actual profit relative to the maximum possible profit obtainable under the static JIT-OS mode. Specifically, this static JIT-OS mode utilizes the capacity of *serus* 1 and 2 at 10 units each to fulfill orders 1 and 2, respectively, achieving the highest profit of $(10 \times 120 + 10 \times 240) - (10 \times 100 + 10 \times 200) = \$600$. Even though the highly efficient system is financially viable, its realized profit competitive ratio is only $40/600 = 6.7\%$, indicating poor performance. Therefore, managing the capacity trade-off (similar to applying the newsvendor model to inventory issues) is crucial to improve the competitive ratio.

Order assignment can also significantly influence the performance of a dynamic JIT-OS.

- **Example 2: Order Assignment in a *Seru* System**

In continuation of Example 1, suppose that supply-demand capacities are perfectly matched within the *seru* system, and both *serus* 1 and 2 are allocated a capacity of 10 each. Assigning orders 1

and 2 to *serus* 1 and 2, respectively, allows the maximum profit of \$600, achieving a 100% service level—a perfect balance of efficiency and flexibility. Conversely, if order 1 is assigned to the highly flexible *seru* 2, the resultant profit and service level plummet to $1200 - 3000 = -\$1,800$ and 50%, respectively, illustrating the poorest balance. (Note that *seru* 1 lacks the skills required to assemble order 2.)

Examples 1 and 2 highlight that although having multiple parallel *serus* increases the system's flexibility to manage a variety of customer orders, suboptimal managerial decisions can undermine this benefit, leading to adverse outcomes. This section delves into the structural dynamics of this trade-off within a general dynamic JIT-OS framework. Effective decision-making strategies are presented in Sections 5, 6 and 7.

The substitution effect of submodularity enables the flexibility to allocate *serus* to respond to uncertain demands. The diminishing returns of submodulairty provides direction for efficiency improvement in *seru* utilization. The substitution effect means that managers can change the allocation of *serus* to meet demand. The diminishing returns demonstrates the phenomenon of the decline of marginal *seru* system value when adding more capacity to a specific *seru*. For a comprehensive explanation of the substitution effect and the diminishing returns, see Milgrom and Strulovici (2009) and Kapralov et al. (2013). A review of submodular optimization definitions is in Section EC.1 of the online Appendix.

4.1. Flexibility of a *Seru* System is Automatic

Flexibility of a *seru* system is investigated using Lemma 1 and Theorem 2. All proofs for Lemmas are in Section EC.4 of the online Appendix.

LEMMA 1. *For order o assembled on an arbitrary sample path ω, capacity consumption $q_o^i(x,\omega)$ of seru i is submodular and exhibits diminishing returns with respect to the initial capacity of another seru j $(i \neq j)$.*

Capacity consumption of *seru* i, $q_o^i(x,\omega)$, is examined using Lemma 1. All subsequent results of this section are based on and extensions of Lemma 1, which says that $q_o^i(x,\omega)$ is a submodular function. Thus $q_o^i(x,\omega)$ possesses the substitution effect (Milgrom and Strulovici 2009), which means that one input can be substituted by another input. For a *seru* system, this means that a portion of the initial capacity of *seru* i (x^i) can be replaced by some of the initial capacity of another *seru* j (x^j) with the effect of using the capacity to meet customer orders. Specific conditions under which capacity of one *seru* can be substituted by capacity of another *seru* are in Lemma EC.8.

The most significant managerial insight from Lemma 1 is its highlight on the ***flexibility*** of *seru* systems. As discussed in Section 1, *seru* systems exhibit greater flexibility and adaptability in comparison to TPSs, especially in volatile markets characterized by frequently changing product models, fluctuating volumes, and short product life cycles. TPSs typically use a long assembly line that can be disrupted because of frequent changes, unlike *seru* systems. Lemma 1 suggests that if *seru* i is unable to accommodate changes in product models or volumes, another *seru* j can

substitute i to adapt to these changes. The ability to use different *serus* to accommodate changing demands is a key aspect of JIT-OS.

Insights derived from Lemma 1 are further expanded in Theorem 2.

THEOREM 2. *For an arbitrary sample path ω, total capacity consumption $\sum_{o\in\mathcal{O}} q_o^i(x,\omega)$ of seru i is submodular and exhibits diminishing returns with respect to the initial capacity of an arbitrary seru j $(i \neq j)$.*

During a production period, orders arrive one by one. Lemma 1 focuses on analyzing a single order o that arrives at a specific point in time. Theorem 2 says that submodularity and diminishing returns apply to all orders along an arbitrary sample path throughout the production period.

The significance of Theorem 2 lies in highlighting dynamic flexibility during the production period. This means that managers can utilize alternative *serus* dynamically to accommodate demand changes during a production period. The ability to quickly respond to changing demand is a crucial advantage of *seru* systems. Theorem 2 provides a theoretical foundation to explain this dynamic flexibility.

4.2. Efficiency of a *Seru* System is Not Automatic

Efficiency in a *seru* system requires deliberate design and strategic management. We now explore how initial capacities of *serus* influence system profit and demonstrate that efficiency emerges only from a careful allocation of resources. We begin by examining the inherent diminishing returns in a *seru* system because of the arbitrary nature of *serus* in Theorems 3.

THEOREM 3. *For an arbitrary sample path ω, profit $R(x^i,\omega)$ from seru i is submodular and exhibits diminishing returns with respect to the initial capacity of another seru j $(i \neq j)$.*

Since both *serus* i and j are arbitrary, diminishing returns exists in the *seru* system as noted in Theorem 3. Because the sum of submodular functions is also submodular (see Lemma EC.1 in the online Appendix), the profit of *seru* system $R(x,\omega) = \sum_{i\in\mathcal{I}} R(x^i,\omega)$ is submodular. This implies that increasing the capacity of a *seru* can reduce the system's marginal profit, because Theorem 3 shows that profit of a *seru* system is submodular and exhibits diminishing returns, and because submodularity literature such as Milgrom and Strulovici (2009) has demonstrated that diminishing returns implies the reduction of marginal benefit from an additional input. Theorem 3 indicates a decline in *efficiency*, similar to the example of adding multiple displays to a PC.

The key insight from Theorem 3 is that efficiency in a *seru* system is not automatic. As we have shown in the example of adding multiple displays to a PC and Section 4.1, automatic would imply that adding capacity to a *seru system* can improve performance (flexibility in Section 4.1) of a *seru* system. However, Theorem 3 shows that adding capacity to a *seru* system can reduce efficiency, because profit margin may become negative, like the example of adding multiple displays to a PC. If a manager adds more displays to a PC, the benefit from an additional display can become smaller than the cost of the display, resulting in negative profit margin. *Efficiency* is defined in Section 2.2 as the gap between output (benefit in the PC example) and input (cost in the PC

example). Negative profit margin means that efficiency is low. So, efficiency in a *seru* system is not automatic just by adding more capacity. Managers should carefully determine the initial capacity of each *seru* to maximize profit. This is illustrated in Example 1, where both highly flexible and highly efficient systems are suboptimal. Instead, a balanced system (with capacities of 10 for both *serus* 1 and 2) yields the highest profit.

Next, we explore how to improve efficiency by leveraging the substitution effect property of submodular functions in Theorem 4.

THEOREM 4. *For an arbitrary sample path ω, profit $R(x^i,\omega)$ from seru i is submodular and exhibits strictly diminishing returns with respect to the initial capacity of another seru j $(i \neq j)$, provided that for an order o on ω, $r_j < r_i \leq \lambda_o$ and $m \leq d_o\tau_o \leq m + \beta_j$. This indicates that seru j can efficiently substitute for seru i in fulfilling order o, resulting in increased efficiency in the system.*

Here, β_j represents additional amount of capacity added to *seru* j. The distinction between diminishing returns and strictly diminishing returns lies in the rate at which marginal profit decreases. In the case of diminishing returns, the additional amount of capacity results in a *smaller or equal* decrease in marginal profit. For strictly diminishing returns, the additional amount of capacity leads to a *strictly smaller* decrease in marginal profit.

Theorem 4 highlights that profit from *seru* i exhibits strictly diminishing returns with respect to the initial capacity of *seru* j. This means two things. First, both *serus* i and j are mutually substitutable in fulfilling order o, because $r_j < r_i \leq \lambda_o$ indicates that $r_j \leq \lambda_o$ and $r_i \leq \lambda_o$ are both true. Second, using *seru* j to replace *seru* i to assemble order o improves efficiency, because $r_j < r_i$ says that *seru* j is cheaper than *seru* i.

Comparison of Theorems 3 and 4 highlights the fact that efficiency is not a given and automatic outcome. To achieve efficiency, the JIT-OS should be designed carefully. Theorem 3 implies that *seru* j can replace *seru* i, regardless of whether this *seru* j has a higher or lower labor cost than *seru* i. On the other hand, Theorem 4 shows that efficiency can only be achieved by using a lower cost *seru* to replace a higher cost *seru*. Efficiency can be achieved by using our designed allocation rules ($r_j < r_i$). Hence, the problem of "how" to design a *seru* system to achieve high efficiency is crucial, which is explored next.

The findings from Sections 4.1 and 4.2 are summarized in Managerial Insight 2.

Managerial Insight 2: *In dynamic uncertain environments, optimizing the efficiency of a seru system is significantly more challenging than in static stable environments. The responsiveness of a seru system remains inherently automatic because of its flexible structure, where multiple serus (e.g., i, j, and others) can substitute for one another to handle unpredictable orders.*

4.3. Further Analysis on Randomness and Comparisons of *Seru* Systems and TPSs

The robustness of the theories established in Sections 4.1 and 4.2 are now addressed under conditions of uncertainty. Comparison between a *seru* system and a TPS line is provided.

REMARK 1. *The expectation of capacity consumed by a seru i ($\mathbb{E}_\omega[q_0^i(x,\omega)]$) is submodular.*

Remark 1 follows directly from Lemma 1 and Milgrom and Roberts (1990), who established the principle of "The expectation of a submodular function is submodular". They demonstrated that analyzing an instance of a submodular function is equivalent to analyzing the function under uncertainty. This conclusion has been used in operations management literature (Simchi-Levi and Wei 2012, Zacharias and Pinedo 2017, Hu and Zhou 2022), where submodular and supermodular functions are used to analyze problems under uncertainty. Our analysis in this section follows this method stream.

Lemma 1 says that the capacity consumption function of a *seru* $q_o^i(x,\omega)$ is submodular and exhibits diminishing returns for a given sample path ω. However, Lemma 1 does not account for the level of uncertainty or demand volatility. $\mathbb{E}_\omega\left[q_0^i(x,\omega)\right]$ is the average capacity consumed by *seru* i across all possible sample paths, providing insights into the system's adaptability under varying levels of demand uncertainty, regardless of specific demand realizations. Such system adaptability is a strength of a *seru* system, enabling it to efficiently respond to a wide range of demand scenarios.

Next, we compare the *seru* system with a TPS assembly line.

Remark 2. *A TPS assembly line lacks submodularity.*

This observation is derived from Lemma 1 and the definition of submodularity, which imply that a variable i (*seru* or line in this paper) can be substituted by another varaible j. For a TPS assembly line, there is only one variable, the TPS assembly line. A TPS assembly line cannot substitute with itself. So, a TPS assembly line does not possess the submodular properties of substitution and diminishing returns.

To compare the performance of a *seru* system with a TPS assembly line under highly uncertain environments, consider the context of *maximum randomness* as defined in **Factory Physics** (Hopp and Spearman 2011). In this scenario, every possible demand situation is equally likely to occur, making it difficult to predict actual demand. For details, see Chapter 7 of Hopp and Spearman (2011). Both a TPS assembly line and a 2-*seru* system have identical operational conditions under *maximum randomness*. Comparative results are summarized in Remark 3.

Remark 3. *Under maximum randomness, consider a customer order for w products, each requiring N operations. Both a TPS assembly line and a 2-seru system are balanced, with each comprising N workers who have identical processing speeds of $1/t$ per operation (= t time to complete each operation). In the TPS line, each worker is responsible for a single operation. In the 2-seru system, each worker manages two operations within a seru. Under these identical conditions, the 2-seru system outperforms the TPS line by achieving a cycle time that is shorter by t time.*

Hopp and Spearman (2011) (page 230) defines cycle time as the average time required to assemble a product. Cycle times are $(N+w-2)t$ and $(N+w-1)t$ (time per product) for a 2-*seru* system and TPS line, respectively. Details on deriving these cycle times are in Appendix EC.5. Notice that the maximum performance improvement ratio is 50% when $N=2$ and $w=1$ (where w is a batch, with each *seru* assembling half of the batch).

In summary, flexibility refers to the capability of a system to adapt its operations to quickly respond to uncertain demands. Remark 1 shows that the substitution effect of a *seru* system is robust under uncertain conditions. Remark 2 demonstrates that a TPS line lacks such substitution effect. Finally, Remark 3 illustrates that a 2-*seru* system outperforms a TPS line under high uncertainty, achieving up to a 50% reduction in cycle time.

Next, to compare TPS lines with *seru* systems under the dynamic JIT-OS mode, the conditions of the Penny Fab One example presented in Section 2.2 are relaxed. Static JIT-OS mode is transitioned to dynamic JIT-OS mode.

- **Dynamic Version of Penny Fab One**

The original scenario of one customer ordering 12 products in Section 2.2 is changed to 12 independent customers, each ordering one product. In this scenario, customer arrival time is *maximum random*, meaning that the arrival time of each customer is equally likely to occur at any time from 0 to ∞. Two service metrics used to evaluate the performance are lead time and inventory availability (Cachon and Terwiesch 2017). For a dynamic JIT-OS production system, cycle time and response rate are the two service metrics.

Response rate is the probability or frequency of serving a customer with an assembled product when they arrive. The cycle time is approximated with a normal distribution. The cycle time of the TPS line is $\mu_{line} = \frac{(8+10+\cdots+30)}{12} = 19$ hours per product with a standard deviation of $\sigma_{line} = 6.9$ hours per product. The cycle time of the 4-*seru* system is $\mu_{seru} = \frac{(8+16+24)\times 4}{12} = 16$ hours per product with a standard deviation of $\sigma_{seru} = 6.53$ hours per product. The 4-*seru* system achieves a 15.79% shorter cycle time than the TPS line.

For the response rate, suppose that a customer arrives at time $t \geq 0$. The standard normal distribution value of t is $z = \frac{t-\mu}{\sigma}$. Response rate is given by $\Phi(z)$. The 4-*seru* system outperforms the TPS line in terms of response rate, since $z_{seru} = \frac{t-\mu_{seru}}{\sigma_{seru}} > z_{line} = \frac{t-\mu_{line}}{\sigma_{line}}$. For example, if $t = 19$ hours, the response rate of the TPS line is $\Phi(z_{line} = 0) = .5$, and the response rate of the 4-*seru* system is $\Phi\left(z_{seru} = \frac{3}{6.53}\right) = .68$. The 4-*seru* system achieves a 36% higher response rate than the TPS line. The *seru* system's superior response rate results from its shorter cycle time, μ_{seru}, and lower standard deviation, σ_{seru}.

The findings from Section 4.3 are summarized in Managerial Insight 3.

Managerial Insight 3: *In highly uncertain environments, a seru system's responsiveness remains robust because of its inherent substitution effect. This enables seru systems to outperform TPS lines by achieving shorter cycle times and higher response rates.*

Finally, to illustrate the efficiency and responsiveness of *seru* systems under *maximum randomness*, the performance of various *seru* systems is plotted in Figure 1.

- **Efficiency versus Responsiveness of *Seru* Systems under Maximum Randomness**

The horizontal axis is the number of *serus* in the system. The left and right vertical axes give the response rate and the efficiency of the *serus,* respectively. As discussed in Section 4.1, response rate of a *seru* system increases with the number of *serus*. In Figure 1, one line and four *serus*

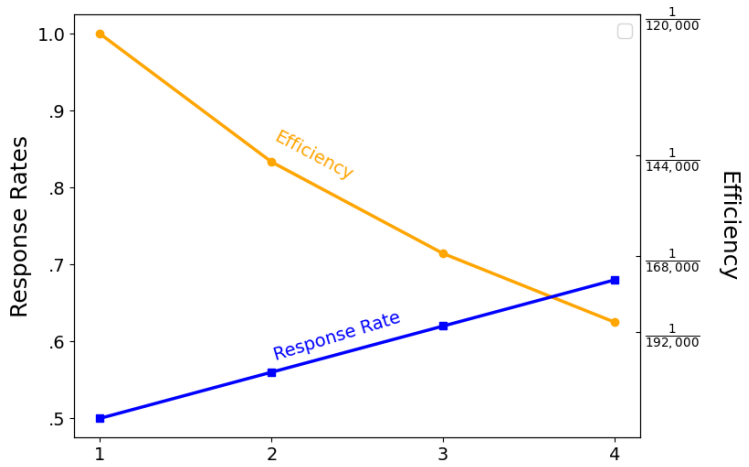

Figure 1 **Trade-off between efficiency and response rate when a customer arrives at 19 hours.**

represent the TPS line and 4-*seru* system in the Penny Fab One example. 3 *serus* means that there are 2 *yatais* with 1 worker each and the 3rd *seru* has 2 workers. All systems are balanced, with the same number of four workers.

Workers are trained in advance to perform all tasks required at their workstations. Each *seru* system uses four identical workers. These training costs are included in the *seru* formation costs. Efficiency is calculated based on the construction cost of each *seru* system. Total wages of each system are fixed at 100,000 JPY per day (approximately $667 per day, based on an exchange rate of $1 = 150 JPY). The first *seru* needs to set up four workstations and provide all necessary tools at each. This initial cost is assumed to be 20,000 JPY. Therefore, the total cost to build one *seru* is 120,000 JPY. Efficiency can be represented as the reciprocal of cost (Cachon and Terwiesch 2017). The efficiency of 1-*seru* system is $\frac{1}{120{,}000}$.

When these four workers are assigned to two *serus*, each *seru* contains two workers. Each worker has mastered the skills for both workstations. Creating two *serus* increases the need for more tools, resulting in extra costs. Assuming these additional tooling costs are 24,000 JPY, the total cost is 120,000 + 24,000 = 144,000 JPY. The efficiency of a 2-*seru* system is $\frac{1}{144{,}000}$.

To construct three *serus* with the same four workers, one *seru* has two workers, and the other two *yatais* each has one worker. In this case, with additional tooling costs of 24,000 JPY, the total cost is 144,000 + 24,000 = 168,000 JPY. The efficiency of the 3-*seru* system is $\frac{1}{168{,}000}$. When four *yatais* are constructed, with additional tooling costs of 24,000 JPY, the total cost is 192,000 JPY. The efficiency is $\frac{1}{192{,}000}$.

Figure 1 shows that response rate increases with the number of *serus*, while efficiency decreases. This result is summarized in Managerial Insight 4.

Managerial Insight 4: *There is a trade-off between efficiency and responsiveness in seru systems.*

5. Decisions on the Initial Capacity Level and Order Assignment

This paper investigates two questions: the "why" and "how" of *seru* systems. The "why" question is the efficiency-flexibility trade-off discussion in Section 2.2 for static JIT-OS and in Section 4 for dynamic JIT-OS. The "how" question focuses on methods to enhance efficiency in a *seru*

system, as discussed in Sections 5, 6, 7 and the online Appendix. The first "how" problem that managers face when creating a cost-efficient *seru* system is determining the initial capacity levels, $x = (x^1, x^2, ..., x^I)$, to maximize profit. To determine the best x, concavity of profit function $R(x, \omega)$ is analyzed. However, Theorem 5 states that the profit function is not quasiconcave.

THEOREM 5. *There exists a sample path ω where $R(x, \omega)$ is not quasiconcave.*

Finding the global optimum of non-quasiconcave functions efficiently remains an open question in optimization. Non-quasiconcave functions are difficult to optimize because they have multiple local optima. To address this issue, a Stochastic Gradient Algorithm (SGA) is developed that seeks stationary points of the expected profit function. Details are in Section EC.2.1 of the online Appendix.

The second "how" problem, creating a cost-efficient dynamic *seru* system, is to determine how to allocate one-by-one arriving orders among one or more *serus*. First-come-first-served (FCFS) is usually used to schedule in this dynamic system. FCFS is reasonable in practice because a primary objective of dynamic JIT-OS is quick response. FCFS is simple and straightforward and can be implemented quickly. In a period where all orders are urgent, quick response times are important. FCFS allows the *seru* system to usually start working on orders as they arrive, without waiting for scheduling decisions or optimizations. Unfortunately, how to optimally allocate one-by-one arriving orders among one or more *serus* is computationally difficult to solve unless NP=P. Order assignment for the dynamic JIT-OS is proved to be NP-hard in Theorem 6.

THEOREM 6. *Order assignment for the dynamic JIT-OS Problem is NP-hard.*

Theorem 6 is extended to Theorem 7 to show the hardness of creating an approximation approach.

THEOREM 7. *It is NP-hard to approximate optimal order assignment for the dynamic JIT-OS Problem within a factor better than $1 - 1/e$.*

To better allocate orders, a Labor Cost Minimization (LCM) Algorithm is proposed. Since profitability is inversely related to costs for any order, minimizing costs maximizes profits. Denote U_o as any remaining production requirements of order o that have not yet been satisfied, W_i as the remaining capacity of *seru* i, $P(o, x)$ as the profit obtained by assembling order o under initial capacity x, and $A(o, x)$ as the set of *serus* with which order o is assembled.

Labor Cost Minimization Algorithm

Step 1. For new order o, form set $\mathcal{G}_o$ whose elements are *serus* that can assemble order o.

Step 2. From $\mathcal{G}_o$, among all *serus* that have positive capacity, select *seru* i that has the lowest labor cost for this order. Assemble order o on *seru* i as much as possible.

Step 3. Update the remaining production requirements that have not yet been satisfied for order o (U_o), the available capacity of *seru* i (W_i), profit $P(o, x)$, and current solution set $A(o, x)$.

Step 4. If the production period ends or there is no more *seru* capacity, STOP. Otherwise, if the unsatisfied requirements of order o are 0, go to Step 1. Otherwise, go to Step 2.

The LCM Algorithm is an online algorithm that dynamically handles tasks. To evaluate its worst-case performance, its competitive ratio, CR is calculated. CR is defined as an online algorithm's worst result against its optimal counterpart, a hypothetical algorithm with full foresight of all future conditions. See Cormen et al. (2022) for details. In this paper, CR is the ratio of the possible minimum profit to the possible maximum profit. This comparison, focusing on the most extreme scenarios, is useful to understand the robustness of the LCM Algorithm to meet unforeseen future orders.

Define $b_{\max}$ as the highest revenue obtained from any order in set $\mathcal{O}$, and $b_{\min}$ as the lowest, within a sample path ω. $m_{\max}$ is the maximum profit margin achievable by any *seru* i, calculated as $(b_{\max} - e_i)/b_{\max}$. Similarly, $m_{\min}$ is the minimum profit margin, determined by $(b_{\min} - e_i)/b_{\min}$. The worst-case analysis is detailed in Theorem 8.

THEOREM 8. *For any seru, the LCM Algorithm has a competitive ratio of*

$$CR = \frac{m_{\min}}{m_{\max}} \times \frac{1 - m_{\max}}{1 - m_{\min}}. \tag{5}$$

This CR falls below 1. A higher value indicates superior online algorithm performance. The CR in this paper is margin-dependent. A constant CR is often preferred because it offers a uniform performance guarantee, simplifying the evaluation and comparison of different algorithms. This consistency makes it easier to benchmark and understand an algorithm's efficacy across diverse scenarios. However, certain online problems, particularly those centered around revenue optimization, present challenges in establishing a constant CR. This difficulty arises because the objective functions in these problems are intricately tied to specific input parameters, such as the varying profit margins, which influence optimization decisions, making the algorithm's performance highly input-sensitive. Consequently, input-dependent CRs become essential in these contexts, which can provide a more accurate and realistic performance measure tailored to the specific operational environment and leverage the detailed structure of the input to optimize revenue more effectively (A detailed discussion is in the the Appendix, following the proof of Theorem 8).

Typical *seru* systems cater to innovative products with profit margins ranging from .2 to .6 (Fisher 1997). The theoretical worst-case competitive ratio for the LCM Algorithm could be as low as $.2/.6 \times .4/.8 = 1/6$.

LEMMA 2. *Assume that for all orders, the highest and lowest profit margins are* $m_{\max} = .6$ *and* $m_{\min} = .2$. *Then the worst-case competitive ratio for the LCM Algorithm is 1/6. Also, there exists an instance of the online order-assignment problem for which the LCM Algorithm achieves exactly this 1/6 ratio relative to an optimal offline solution. Hence, the worst-case competitive ratio of 1/6 is tight.*

Practical operational strategies can significantly improve this ratio to approach nearly 1. Specifics of these strategies and their impact on competitive ratios are elaborated in Section 7.

6. Comparisons and Near-Optimal Performance of SGA

The Stochastic Gradient Algorithm is now validated by comparing its effectiveness with that of the newsvendor algorithm. SGA offers greater flexibility than a newsvendor algorithm. SGA can address issues beyond the newsvendor model's scope, such as when $a_o > 0$, as in Equations 3 and 4. Therefore, in Section 6.1, our performance comparison focuses on problems that are solvable by both methodologies.

In Section 6.2, we compare the performance of SGA against the global optimum obtained via offline optimization. SGA is tested with various initial capacity settings to assess its robustness and optimality gap.

6.1. Comparative Results

The number of orders is modeled as a Poisson distribution with mean λ. The probability of an order using *seru* i is $\mathbb{P}(i)$, resulting that the number of orders using *seru* i follows a thinned Poisson distribution with mean $\lambda\mathbb{P}(i)$. For each order o, demand is normally distributed, with a common mean μ and standard deviation σ. Then the mean total demand for *seru* i is $\lambda\mathbb{P}(i)\mu$. Total demand's standard deviation for *seru* i is $\sqrt{\lambda\mathbb{P}(i)}\sigma$. An order's demand coefficient of variation is $cv_o = \sigma/\mu$. The total demand's coefficient of variation for *seru* i is $cv_i = \sigma/\sqrt{\lambda\mathbb{P}(i)}\mu$.

Parameters for experiments are as follows. $\lambda = 8$ and $\mu = 5$. Probability $\mathbb{P}(i)$ is uniformly set at $1/I$, where $I = 5$ is the total number of *serus*. The *seru* system encompasses a set of five distinct skills. The skill sets from *seru* one to five are $\{1,2\},\{2,3\},\{3,4\},\{4,5\},\{5,1\}$. c_j $(j = 1, ..., J)$ is the labor cost of assembly skill j, which is \$100 for all skills in our experiment. The skill required for order o is determined by a discrete uniform distribution ranging from one to five. The processing time for a single product of an order is uniformly set to one. Market volatility is simulated via cv_o. Higher cv_o values indicate increased volatility. Values of cv_o of .2, .4, .6, .8, and 1 are investigated.

In the newsvendor model, $F(x^i) = c_u/(c_u + c_o)$, where $F(x^i)$ is the probability that demand for *seru* i is less than or equal to its initial capacity x^i. c_o is the overage cost that equals cost minus salvage price. In this paper, the overage cost is labor cost e_i. Salvage price is zero, so $c_o = e_i$. $c_o = \$200$, are labor costs of the two skills in each *seru*. $c_u = b_o - e_i$ is the underage cost, the lost opportunity cost when demand exceeds prepared capacity. Capacity x^i is $x^i = \lambda\mathbb{P}(i)\mu + Z^i\sqrt{\lambda\mathbb{P}(i)}\sigma$, where $Z^i = \Phi^{-1}(c_u/(c_u + c_o))$ is the standard normal distribution's z statistic.

Each experimental scenario is simulated by generating sample paths using three random variables: number of orders ($\lambda = 8$), product type (uniformly chosen from one to five), and each order's demand ($\mu = 5$). To capture effects of varying market volatility, ten experiments for each cv_o level (.2, .4, .6, .8, and 1) are conducted. Both SGA and newsvendor use the same LCM Algorithm to assign orders to *serus*.

Fisher (1997) indicates that profit margins for innovative products vary between .2 and .6. Our study examines problems with profit margins set at .33 and .6 to align with these observations.

The experimental results when profit margin is .33 are in Figure 2. Revenue (v_j) for using a skill $(j = 1, ..., J)$ is \$300. Average profit of SGA exceeds that of the newsvendor method by 28.7%,

highlighting substantial improvement. The data indicates that SGA consistently outperforms the newsvendor method across all tested levels of market volatility.

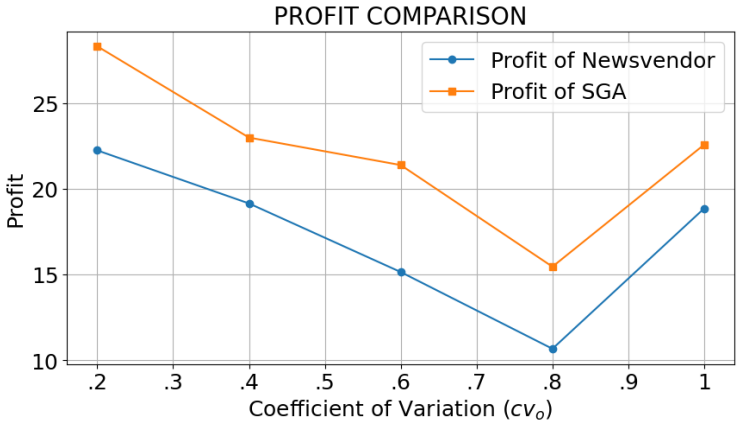

Figure 2 **Profit comparison of Newsvendor and SGA when profit margin is .33.**

A paired-observation t-test was conducted to determine if there is a significant difference in profits generated by the two methods. The null hypothesis for the test is stated as $\mu_{SGA} - \mu_{Newsvendor} \leq 0$, which says that SGA does not yield a higher profit than the newsvendor method. The alternative hypothesis is that SGA yields a higher profit. The test returned a t-value of 3.96, which is significantly higher than the critical value of 2.4 at the .01 α-level. The corresponding p-value is .0001, indicating strong evidence against the null hypothesis. With 49 degrees of freedom, these results strongly suggest that the average profit from SGA is significantly higher than profit from the newsvendor method.

The experimental results when profit margin is .6 are shown in Figure 3. In these experiments, revenue (v_j) for requiring a skill $(j = 1, ..., J)$ is \$500. Average profit of SGA exceeds that of the newsvendor method by 13.6%. The t-test returned a t-value of 3.404. The corresponding p-value is .0007. Again, the results strongly suggest that the average profit from SGA is significantly higher than profit from the newsvendor method.

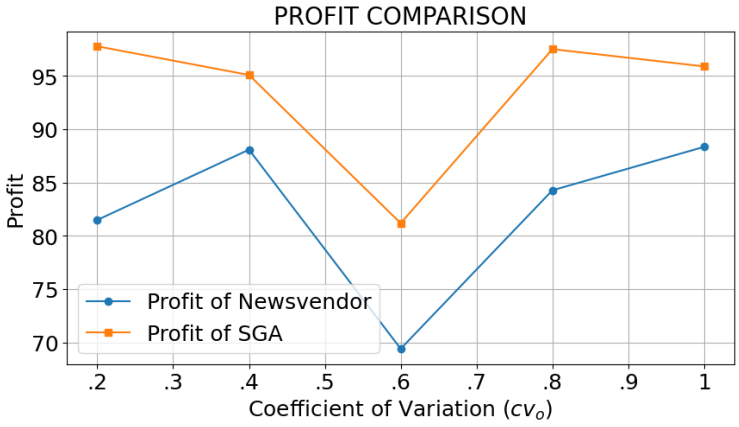

Figure 3 **Profit comparison of Newsvendor and SGA when profit margin is .6.**

The findings show that SGA surpasses the newsvendor method in efficiency by generating higher profits and demonstrates greater flexibility, accommodating scenarios that the newsvendor method

cannot. On the other hand, the newsvendor method is useful for its simple structure. A key finding in determining capacity levels is that a *seru* i's critical ratio $F(x^i) = c_u/(c_u + c_o)$ equals the *seru*'s profit margin $(b_o - e_i)/b_o$. Idle times lack salvage values, so $c_o = e_i$. The equality between $F(x^i)$ and profit margin is consistent with practitioners' intuitive understanding, and provides practical benefits. Most companies maintain precise profit margin records, making the newsvendor method implementable for capacity calculations. This finding is particularly advantageous for small factories who do not have complex capacity management systems. Newsvendor enables manual and effective capacity planning using profit margins. Insights from the above analysis lead to Managerial Insight 5.

Managerial Insight 5: *SGA surpasses the newsvendor method in performance and flexibility. SGA is an approach for companies to realize higher profits and accommodate diverse scenarios. In contrast, the newsvendor method's simplicity and its intuitive link of service level to profit margin make it a user-friendly option for capacity planning, especially for small enterprises.*

6.2. Near-Optimal Performance of SGA

We now compare the performance of SGA against the global optimum obtained via offline optimization. The purpose of this experiment is described as followed based on the common knowledge of using SGA to solve non-quasiconcave problems. Using SGA to solve non-quasiconcave problems, starting from different initial solutions (i.e., initial capacities of *serus* in this paper) can result in various outcomes because the optimization landscape can contain multiple local optima, saddle points, and flat regions. Starting from different initial solutions typically generates a diverse range of solutions, highlighting the complexity of solving non-quasiconcave problems.

To effectively test the performance of an SGA, it is important to evaluate its performance across multiple initial solutions. When solutions are constrained to be nonnegative with a mean μ based on their distribution, a practical and efficient way to test involves using a set of representative initial solutions: 0 (minimal value), μ (the statistical mean value from the distribution), a large value (such as twice μ, representing an upper bound), and random values uniformly distributed between 0 and the large value. These initializations provide a comprehensive sample of the solution space, enabling the evaluation of SGA's ability to converge to high-quality solutions consistently.

Thirty experimental sample paths were generated as done in Section 6.1. Interested readers can contact the authors for our Python code. Parameters that are different from Section 6.1 are λ and μ. In this section, $\lambda = 10$ and $\mu = 1.05$. cv_o and profit margin are fixed to 1 and 1/3, respectively. Other parameters such as the number of *serus* I are the same as those in Section 6.1. Experimental results are in Table 3.

Table 3 Profit Summary Table Across Multiple Initial *Seru* Capacities

30-Sample	**Optimum**	**0-Initial**	**μ-Initial**	**Random-Initial**	**2μ-Initial**
Average Profit	\$10.4	\$10.3	\$10.1667	\$10.2667	\$10.1333
Approximation Ratio	100%	99.04%	97.76%	98.72%	97.44%
Success Samples	30	28	26	27	25
Success Rate	100%	93.33%	86.67%	90%	83.33%

Table 3 is a summary table across multiple initial *seru* capacities The Optimum Column of offline optimization is the benchmark to serve as the best solution and to allow other solutions (the next 4 columns) to compare with this benchmark to find other solutions' gaps to this benchmark solution. The Optimum Column provides the overall global optimum profit for 30 problems. The 4 columns from 0-Initial to 2μ-Initial give SGA's performance under different initial capacity settings. 0-Initial: *serus* start with 0 initial capacities. μ-Initial: *serus* start with initial capacities of the statistical mean μ of the distribution, which equals $\lambda\mu/I = 10 * 1.05/5 = 2.1$. Random-Initial: *serus* start with random initial capacities ranging between 0 and 4.2. 2μ-Initial: *serus* start with twice the average mean $= 2 \times \mu$-Initial, which equals 4.2. The metrics evaluated to assess the performance of each initialization are as follows. Average Profit: Average profit obtained across 30 sample paths. Approximation Ratio: Ratio of the average profit of each initialization to the global optimum, for example $\$10.3/\$10.4 = 99.04\%$ for 0-Initial SGA. Success Samples: Number of problems where an initialization achieved the global optimum profit. Success Rate: Percentage of problems where an initialization achieved the global optimum, for example $28/30 = 93.33\%$ for 0-Initial SGA.

Average Profit is important as it evaluates SGA's ability to consistently generate high-value solutions across diverse experimental scenarios. To assess whether there were statistically significant differences between the performance of the different initializations, an ANOVA test was implemented on Average Profit. The p-value from the one-way ANOVA was .99, which is far above the typical significance level of .05. This means that we fail to reject the null hypothesis, indicating no significant difference between the performance of SGA initializations and the offline optimization. This result is promising as it statistically confirms that SGA, regardless of the initial condition, can achieve performance on par with global optimization.

The lowest Approximation Ratio is 97.44% under the 2μ-Initial condition, showing its ability to closely approach the global optimum. Approximation Ratios for the other SGA variants—99.04% (0-Initial), 97.76% (μ-Initial), and 98.72% (Random-Initial)—highlight the algorithm's robust capability to maintain near-optimal performance consistently.

For 0-Initial, there are two instances where the global optimum was not achieved (28 optima out of 30 problems). However, gaps between these two instances and the optimal are quite small. In one instance, the difference from the optimal value is 1, and in the other, it is 2. Thus, the average distance to optimal is calculated as $(1+2)/30 = .1$. This gap is also reflected in the Average Profit difference, where $\$10.4 - \$10.3 = \$.1$ again, highlighting the near-optimal performance of SGA.

7. Discussion of Competitive Ratios

The LCM Algorithm is evaluated by its competitive ratio $CR = \frac{m_{\min}}{m_{\max}} \times \frac{1-m_{\max}}{1-m_{\min}}$. CR assesses the algorithm's performance in a worst-case circumstance compared to the optimal solution. Figure 4 shows that CR varies in response to the range between the minimum and maximum profit margins. For example, consider the contour level $CR = .4$. This occurs at point A when the maximum profit margin $m_{\max}$ is .385 and the minimum profit margin $m_{\min}$ is .2, when $m_{\max} = .6$ and $m_{\min} = .375$

at point B, and when $m_{\max} = .493$ and $m_{\min} = .28$ at point C. $CR = .4$ means that the LCM Algorithm's worst-case performance is 60% less than the optimal solution.

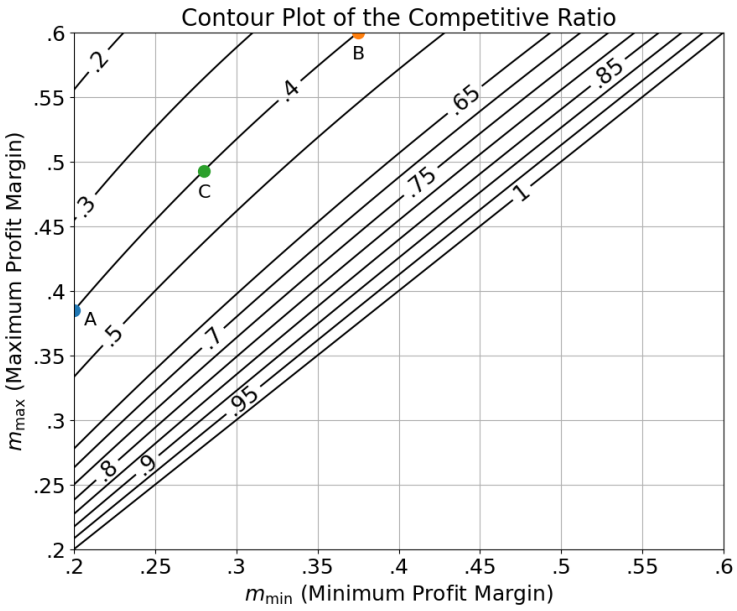

Figure 4 **Competitive ratio CR with respect to minimum and maximum profit margins.**

The CR formula can be used as a strategic tool for managers to preemptively manage and mitigate risk in worst-case scenarios. Either CR, $m_{\min}$, or $m_{\max}$ can be calculated, when the other two are given. For instance, suppose that the highest profit margin within a current product mix is $m_{\max} = .46$, and a manager is aiming for a worst-case CR of .9. The CR formula can be used to compute $m_{\min} = \frac{CR \times m_{\max}}{1-(1-CR)m_{\max}} = .434$. This means that the manager should aim for the minimum profit margin to be at least .434. This enables precise control over product selection and system performance.

The most unfavorable competitive ratio is 1/6=.17 (upper-left corner in Figure 4), occurring in a scenario where all capacity is allocated to orders with a profit margin of .2 (because $CR = 1/6$ is the result of $m_{\min} = .2$ and $m_{\max} = .6$. $CR = \frac{m_{\min}}{m_{\max}} \times \frac{1-m_{\max}}{1-m_{\min}} = \frac{.2}{.6} \times \frac{1-.6}{1-.2} = 1/6$), leaving orders with higher profit margins unfulfilled. Such an extreme case can arise under three possible specific conditions. (1) The *seru* system is dealing with a wide range of orders, encompassing both the lowest and highest profit margins, .2 and .6. (2) Orders with the lowest profit margin are processed before those with the highest. (3) *Seru* capacity is entirely used by the orders with the lowest profit margins, leaving no room for more profitable orders.

This worst scenario is highly unlikely as it requires the simultaneous occurrence of all three stringent conditions, which is rare in practice. Strategic approaches using the CR formula suggest that there are straightforward ways to increase CR towards an ideal value of nearly 1. These strategies are designed to balance capacity and order fulfillment, aiming for more effective capacity utilization by orders with different profit margins.

Figure 4 shows that $CR = 1$ when $m_{\min} = m_{\max}$ on the diagonal. Typically, a *seru* system produces a product category with relatively similar profit margins, such as premium cameras. In

situations where a *seru* system handles a product mix with significant variations in profit margins, one pragmatic approach to increase CR is to segment the single *seru* system into multiple systems. Each system would produce products with similar profit margins, ensuring higher CR within each system. For instance, a consulted electronics manufacturer employs two production systems to separately produce high and low profit margin products. (This electronics manufacturer's products are an indispensable peripheral for computers. The manufacturer is the largest in the world with about one-third market share. The first author of this paper consulted with this company on April 22, 2024, at the company's headquarters in southern China, visiting several of its main factories, to conduct a production management evaluation for the company.)

Managerial Insight 6: *To maintain high efficiency and control risk in worst-case scenarios, managers should ensure that within a seru system, product types exhibit similar profit margins. If significant margin disparities exist, subdividing the system into multiple seru systems can increase CR. This strategic segmentation aligns operational focus with profit optimization.*

Managerial Insight 6 contrasts with the traditional Toyota mixed-product assembly line. TPS permits a diverse range of profit margins but requires similar processing tasks for each product. In contrast, a *seru* system flips this approach, requiring products within the system to have similar profit margins while accommodating a variety of processing tasks.

Subdividing a *seru* system into multiple systems tailored to specific profit margins may sometimes be challenging. An alternative strategy could be to manage the sequence of customer order acceptance based on their profit margins, from highest to lowest. For example, a company could use the first day of the order cycle exclusively for orders with the highest profit margins, followed by middle-margin orders on the second day, and so forth. Since the LCM Algorithm uses FCFS, this sequencing ensures that higher-margin orders are prioritized and capacity is optimally allocated without the need for system subdivision.

Consider a manufacturer of electronic devices who prepares a production period of 4 days for China's Double 11 with a make-to-order production mode. During these four days, the company accepts randomly arriving orders from customers. The company's products are categorized as high, medium, and low profit margins. On the first two days, the company accepts orders only for the high profit margin products. On the third day, it accepts orders only for high and medium profit margin products. On the fourth day, it accepts the remaining orders for all products. The company can use the LCM Algorithm to assign orders to *serus* with FCFS. This tiered approach to order acceptance allows the company to take advantage of the higher profitability of premier products while still fulfilling a broad product demand. For example, OpenOrder is a software application published by FasterCapital, a consulting firm. OpenOrder can be used to manage customer orders. OpenOrder has many functions. One function prioritizes customer orders based on customer importance. This can improve profitability. This function has the same mechanism as our strategy of accepting orders based on profit margins.

Managerial Insight 7: *When subdividing a seru system is impractical, strategically scheduling order submissions based on profit margins can be an effective alternative. By aligning order intake with product profitability, high-margin orders first, followed by those with lower margins, companies can achieve efficient capacity utilization and profit improvement.*

For industry leaders like Nintendo and Apple, with their extensive base of loyal customers and retailers competing fiercely for their products, the strategy of Managerial Insight 7 is particularly practical. Given their substantial market influence and negotiation power, these companies can effectively implement a strategic order acceptance based on profit margins. This allows them to prioritize high-profit orders, optimizing operational efficiency and maximizing revenue. Essentially, Nintendo and Apple can manage their supply to meet the most profitable demands first, leveraging their strong brand positions to enhance both supply chain efficiency and profit margins.

For companies like Fujitsu, NEC, Dell, and HP, which are prominent competitors in the PC industry, employing Managerial Insight 7 strategy involves more sophisticated tactics. These firms typically feature online platforms where customers can configure products by selecting from a variety of available components such as CPUs and GPUs. To implement Managerial Insight 7 strategy effectively, these companies could strategically release component lists in a chronological order based on profit margins. By doing so, they can prioritize the visibility of high-margin parts, offering these first and subsequently releasing low-margin components. This approach not only streamlines product customization but also maximizes profitability by aligning component visibility with potential earnings from each part, improving the probability that the most lucrative options are sold first. This method enhances the company's ability to manage supply effectively while catering to customer preferences in a competitive market.

For small businesses with little competitive strength, securing orders often takes precedence over strategic order acceptance based on profit margins. These enterprises typically offer a narrow range of products, with relatively small differences in profit margins between them. Consequently, such companies can naturally achieve a high CR without the need for complex strategies. For those small businesses that do experience significant differences in profit margins across their products, employing the strategy outlined in Managerial Insight 6 could be more advantageous.

8. Conclusion

Seru production systems demonstrate significant adaptability and efficacy to address challenges posed by market volatility that sometimes disrupts production systems like TPS. Originally adopted by electronics such as Canon and Sony to manage rapid changes in demand and product models, *seru* has expanded its influence to sectors ranging from food and automotive parts manufacturing to healthcare and logistics. This versatility is highlighted in its ability to maintain efficiency and responsiveness through its operational modes: regular, for consistent daily production; seasonal, to adapt to peak demand periods; and emergent, to respond to crises like pandemics. Such adaptability shows that *seru* systems have capabilities to manage production dynamics, achieve profitability and responsiveness across diverse market conditions.

A *seru* system is managed by JIT-OS, which incorporates decisions on setting each *seru*'s capacity and assigning customer orders. JIT-OS operates in two modes: static and dynamic. The static mode predetermines *seru* capacities and order assignments, and is optimally solvable in polynomial time, effectively achieving both responsiveness and efficiency.

In contrast, the dynamic mode deals with real-time incoming orders, adapting flexibly to changing demands. This mode naturally offers flexibility because of the independence of separate *serus*. If one *seru* cannot satisfy a particular demand, other *serus* can be quickly utilized to meet this need, a process explained through the substitution property of submodular functions. However, efficiency is not automatically assured in dynamic JIT-OS. Selecting a high labor cost *seru* to fulfill demand can reduce efficiency, as described by the diminishing returns property of submodular functions. This leads to a significant "how" question: how can a *seru* system be designed to enhance efficiency? This design challenge is complex and categorized as NP-hard. To address non-quasiconcave problems within this context, a convergent stochastic gradient algorithm is developed to find stable solutions, and for NP-hard issues, an online polynomial-time algorithm is implemented. In our experimental comparisons, the stochastic gradient algorithm outperforms the newsvendor method and can generate near-optimal solutions. For the polynomial-time algorithm, we introduce two strategies that enhance competitive ratios, nearly reaching one in practical applications.

We have provided insights into why and how *seru* systems are responsive and efficient in volatile markets, leading to several interesting future research directions. First, The methodologies formulated in this paper can serve as benchmarks for future research. For the submodular profit function, optimization techniques related to submodularity could enhance algorithm efficiency. Additionally, the SGA explored here could be compared with similar methods such as Newton's method, Quasi-Newton methods, and Conjugate Gradient method. Regarding the greedy order assignment algorithm, which exhibits a competitive ratio of 1/6, further development of online algorithms could aim to achieve competitive ratios greater than 1/6, pushing the boundaries of current algorithmic efficiency. Second, one particularly important area is the impact of *seru* systems on supply chain resilience. The recent emphasis on resilience because of COVID-19 and frequent natural disasters, like earthquakes in Japan, has highlighted the potential of reconfigurable production systems. As noted by Cohen et al. (2022) and Ivanov (2023), such systems, especially *seru* systems, are viewed as exemplary models for enhancing supply chain resilience. Investigating whether *seru* systems can indeed improve resilience presents a compelling research opportunity. Third, another promising direction is the cross-industry application of *seru* systems. Although lean principles are traditionally associated with manufacturing, they have seen successful applications in other sectors. As introduced in Section 2.1, *seru* principles have been applied successfully beyond the manufacturing domain. Exploring these applications across different industries offers an attractive avenue for research.

Acknowledgments

We are grateful to Department Editor Panos Kouvelis, the Senior Editor, and the three anonymous

reviewers for their thoughtful comments and constructive suggestions. Their feedback has substantially improved the clarity, rigor, and overall quality of this paper. This paper was supported by KAKEN 25K08175, and the Natural Science Foundation of China under Projects 72471169 and 72231005.

References

Baldwin CY, Clark KB (2000) *Design Rules: The Power of Modularity*, Volume 1 (Cambridge, MA: MIT Press).

Bendoly E, Bachrach DG, Esper TL, Blanco C, Iversen J, Yin Y (2021) Operations in the upper echelons: leading sustainability through stewardship. *International Journal of Operations & Production Management* 41(11):1737–1760.

Boysen N, Fliedner M, Scholl A (2009) Assembly line balancing: Joint precedence graphs under high product variety. *IIE Transactions* 41(3):183–193.

Boysen N, Schulze P, Scholl A (2022) Assembly line balancing: What happened in the last fifteen years? *European Journal of Operational Research* 301(3):797–814.

Cachon G, Terwiesch C (2017) *Operations Management* (New York: McGraw-Hill Education).

Cohen M, Cui S, Doetsch S, Ernst R, Huchzermeier A, Kouvelis P, Lee H, Matsuo H, Tsay A (2022) Bespoke supply-chain resilience: the gap between theory and practice. *Journal of Operations Management* 68(5):515–531.

Cormen TH, Leiserson CE, Rivest RL, Stein C (2022) *Introduction to Algorithms* (Cambridge, MA: MIT Press).

de Treville S, Ketokivi M, Singhal V (2017) Competitive manufacturing in a high-cost environment. *Journal of Operations Management* 49:1–5.

Editor (2011) *Seru* production changes the frontlines. *Food Plant Manager* 175:17–33.

Fisher M (1997) What is the right supply chain for your product? *Harvard Business Review* 75:105–117.

Gai Y, Yin Y, Li D, Zhang Y, Tang J (2023) Maximizing the throughput of a rotating *seru* with nonpreemptive discrete stations. *Naval Research Logistics* 70(8):910–928.

Hopp W, Spearman M (2011) *Factory Physics: Foundations of Manufacturing Management, 3rd Ed* (Boston: Irwin/McGraw-Hill).

Hopp WJ, Spearman MS (2021) The lenses of lean: Visioning the science and practice of efficiency. *Journal of Operations Management* 67(5):610–626.

Hu M, Zhou Y (2022) Dynamic type matching. *Manufacturing & Service Operations Management* 24(1):125–142.

Ivanov D (2023) Two views of supply chain resilience. *International Journal of Production Research* Latest Articles:https://doi.org/10.1080/00207543.2023.2253328, URL `http://dx.doi.org/10.1080/00207543.2023.2253328`, published online: 04 Sep 2023.

Jiang K, Lee H, Seifert R (2006) Satisfying customer preferences via mass customization and mass production. *IIE Transactions* 38(1):25–38.

Kapralov M, Post I, Vondrák J (2013) Online submodular welfare maximization: Greedy is optimal. *Proceedings of the Twenty-fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, 1216–1225 (Society for Industrial and Applied Mathematics).

Li D, Jiang Y, Zhang J, Cui Z, Yin Y (2023) An on-line *seru* scheduling algorithm with proactive waiting considering resource conflicts. *European Journal of Operational Research* 309(2):506–515.

Mendelson H, Parlaktürk A (2008) Product-line competition: Customization vs. proliferation. *Management Science* 54(12):2039–2053.

Milgrom P, Roberts J (1990) The economics of modern manufacturing: Technology, strategy, and organization. *The American Economic Review* 80(3):511–528.

Milgrom P, Strulovici B (2009) Substitute goods, auctions, and equilibrium. *Journal of Economic Theory* 144(1):212–247.

Mitamura F (2012) *Iris Ohyama: Transparent Management Technique* (Tokyo: Toyokeizai Publishing Co.).

Mitra D, Mitrani I (1990) Analysis of a kanban discipline for cell coordination in production lines. I. *Management Science* 36(12):1548–1566.

Mitra D, Mitrani I (1991) Analysis of a kanban discipline for cell coordination in production lines, II: Stochastic demands. *Operations Research* 39(5):807–823.

Roth A, Singhal J, Singhal K, Tang C (2016) Knowledge creation and dissemination in operations and supply chain management. *Production and Operations Management* 25(9):1473–1488.

Simchi-Levi D, Wei Y (2012) Understanding the performance of the long chain and sparse designs in process flexibility. *Operations Research* 60(5):1125–1141.

Stecke K, Yin Y, Kaku I, Murase Y (2012) *Seru*: The organizational extension of JIT for a super-talent factory. *International Journal of Strategic Decision Sciences* 3(1):105–118.

Sudo K (2019) What is the *seru* nursing provision method that allows for optimal performance beside the patient? *New Medical World Weekly* 3339:1–2.

Swaminathan J, Tayur S (1998) Managing broader product lines through delayed differentiation using vanilla boxes. *Management Science* 44(12):S161–S172.

Takano A (2005) Only original factories survive: Part 2—canon's strategy seen in the acquisition of NEC machinery. *Nikkei Monozukuri* 1–2.

Tayur SR (1993) Structural properties and a heuristic for kanban-controlled serial lines. *Management Science* 39(11):1347–1368.

Wimo (2020) Grand reveal of the manufacturing process: Delivering high-quality coozy. `https://www.wimo.co.jp/blog/manufacturing`.

Wu Z, Tong T, Lan Y, Yin Y (2024) The effects of learning and fatigue on pick efficiency. *Working Paper* .

Yamaha (2024) The scene of *seru* production by skilled craftsmen. `https://global.yamaha-motor.com/jp/design_technology/craftsmanship/mc/mc2.html`.

Yin Y, Chung SH, Ma HL, Li D, Zhang Z, Liu C, Yu Y, Gai Y, Tang J, Kaku I (2025) *Seru* production systems. *Encyclopedia in Operations Management (Elsevier)* Forthcoming chapter.

Yin Y, Kaku I, Stecke K (2008) The evolution of *seru* production systems throughout Canon. *Operations Management Education Review* 2:27–40.

Yin Y, Stecke K, Swink M, Kaku I (2017) Lessons from *seru* production on manufacturing competitively in a high cost environment. *Journal of Operations Management* 49:67–76.

Zacharias C, Pinedo M (2017) Managing customer arrivals in service systems with multiple identical servers. *Manufacturing & Service Operations Management* 19(4):639–656.

Zhang Z, Song X, Gong X, Yin Y, Lev B, Zhou X (2022) An exact quadratic programming approach based on convex reformulation for *seru* scheduling problems. *Naval Research Logistics* 69(8):1096–1107.

Appendix

We give some notation that is required. The profit obtained from assembling one product for order o by *seru* i is defined as $\theta_{io} = b_o - e_i$, if $\alpha_{io} = 1$, and $-\infty$, otherwise.

EC.1. Definitions

Now some definitions are given.

DEFINITION EC.1. A set $X \subset \Re^n$ is a lattice if for any $x, y \in X$ there is $x \wedge y$ and $x \vee y$ in X. The vector $x \wedge y$, called the *meet*, is a vector whose ith component is $\min\{x^i, y^i\}$. The vector $x \vee y$, called the *join*, is a vector whose ith component is $\max\{x^i, y^i\}$.

We extend the definitions of submodular function and diminishing returns in Kapralov et al. (2013) for a *seru* capacity system in the following Definition EC.2.

DEFINITION EC.2. Let $X \subset \Re^n$ be a lattice and $f : X \to \Re$. A vector $z \in X$ can be represented as (z_{-ij}, z^i, z^j), where z^i and z^j are respective components i and j of z. Let $(z_{-ij}, \widehat{z}^i, \widehat{z}^j)$ be the vector obtained from z by replacing components i and j with respective $\widehat{z}^i$ and $\widehat{z}^j$, where $\widehat{z}^i \geq z^i$, $\widehat{z}^j \geq z^j$, and $(z_{-ij}, \widehat{z}^i, \widehat{z}^j) \in X$, $(z_{-ij}, \widehat{z}^i, z^j) \in X$, $(z_{-ij}, z^i, \widehat{z}^j) \in X$. Function f is submodular and satisfies *diminishing returns* on X if and only if

$$f(z_{-ij}, \widehat{z}^i, \widehat{z}^j) - f(z_{-ij}, \widehat{z}^i, z^j) \leq f(z_{-ij}, z^i, \widehat{z}^j) - f(z_{-ij}, z^i, z^j). \tag{EC.1}$$

It is easy to see that our Definition EC.2 aligns with the concept of submodular functions by expressing formula (EC.1) as $f(z_{-ij}, \widehat{z}^i, \widehat{z}^j) + f(z_{-ij}, z^i, z^j) \leq f(z_{-ij}, z^i, \widehat{z}^j) + f(z_{-ij}, \widehat{z}^i, z^j)$, which is the definition of submodular functions. Thus f is submodular and satisfies the diminishing returns principle (Kapralov et al. 2013). We next extend Definition EC.2 to multiple functions in Lemma EC.1.

LEMMA EC.1. *If both functions f and g satisfy diminishing returns on X, then function $f + g$ satisfies diminishing returns on X.*

In the following analysis, Lemma EC.1 and the definitions are used to examine the flexibility and efficiency of *seru* systems and explain why they can achieve both in volatile markets. Define $X \subset \Re^I_+$ as the set of initial capacity levels for a *seru* system. Lemma EC.2 proves that any initial capacity level, x, of a *seru* system is an instance of Definition EC.1.

LEMMA EC.2. *Let $X \subset \Re^I_+$ be the set of initial capacity levels of a seru system. Then X is a lattice.*

Effects of inputs on outputs in a *seru* system are now analyzed using the concept of submodular function in Definition EC.2. Inputs are the initial capacities of two heterogeneous *serus*, x^i and x^j, where $i \neq j$. Using formula (EC.1), two outputs, capacity consumption $q^i_o(x, \omega)$ of *seru* i and profit $R(x^i, \omega)$ of the *seru* system from *seru* i, are analyzed, to investigate flexibility and efficiency.

EC.2. Algorithms

EC.2.1. Stochastic Gradient Algorithm (SGA)

Many algorithms for unconstrained optimization rely on gradient methods that are based on approximating a function with a low-degree partial derivative (usually of degree one or two), using Taylor's expansion. Gradient algorithms have been used to solve operations management problems (Mahajan and Van Ryzin 2001, Newton et al. 2018). Similar to them, a one-degree partial derivative algorithm is used in this paper.

In the following SGA, t is the iteration-index inside the loop, z_t is the initial capacity of the *seru* system in the tth iteration of the algorithm, ω_t is the sample path in the tth iteration, and γ_t is the step size in the tth iteration.

Stochastic Gradient Algorithm

Step 1. Choose the initial capacity $z_0 \in \Re^I$. Let $t=0$.

Step 2. Given a sample path ω_t, generate a gradient $\nabla\mathbb{E}[R(z_t,\omega_t)]$.

Step 3. Compute $z_{t+1} = z_t + \gamma_t \nabla\mathbb{E}[R(z_t,\omega_t)]$.

Step 4. Let $t=t+1$ and go to **Step 2**.

There are three questions related to the stochastic gradient algorithm. The first is how to choose the initial capacity z_0 and the step size γ_t. z_0 is a vector that can be simply set to zeros. γ_t is usually defined as $\gamma_t > 0$, $\sum_{t=0}^{\infty}\gamma_t = \infty$, and $\sum_{t=0}^{\infty}\gamma_t^2 < \infty$. Our algorithm follows this. The second is how to calculate gradient $\nabla\mathbb{E}[R(z_t,\omega_t)]$. Third, SG algorithm does not have a termination criterion. What is the convergence of this algorithm? The latter two questions on gradients and convergence are addressed next.

EC.2.1.1. Calculation of Gradients

Defining the gradient (or subgradient at nonsmooth points) is direct if $\nabla\mathbb{E}[R(x,\omega)] = \mathbb{E}[\nabla R(x,\omega)]$. However, Olofsson and Andersson (2012) illustrated that if g is a function, most often $g[\mathbb{E}(X)] \neq \mathbb{E}[g(X)]$. Fortunately in Lemma EC.3, our expectation and derivative can be interchanged under specific conditions.

LEMMA EC.3. *Let X be a random function. $X(\theta)$ is its value at θ, $\theta \in [a,b]$. In interval $[a,b]$, if X is differentiable, and X is integrable, and X satisfies a Lipschitz condition with modulus K_X, $\mathbb{E}[K_X] < \infty$, then the derivatives $\{\mathbb{E}[X(\theta)]', \theta \in [a,b]\}$ exist and $\mathbb{E}[X'(\theta)] = \mathbb{E}[X(\theta)]'$ for all $\theta \in [a,b]$.*

Lemmas EC.4, EC.5, and EC.6 show that $R(x,\omega)$ satisfies the three conditions in Lemma EC.3.

LEMMA EC.4. *For any order $o \in \mathcal{O}$, x_o is a Lipschitz function.*

LEMMA EC.5. *For any order $o \in \mathcal{O}$ and any seru $i \in \mathcal{I}$, then*

$$\|q_o^i(x,\omega) - q_o^i(y,\omega)\| \leq \|(x_o^i - x_{o+1}^i) - (y_o^i - y_{o+1}^i)\|.$$

LEMMA EC.6. *For order $o \in \mathcal{O}$, if its demand d_o has an absolutely continuous cumulative distribution function, and d_o and O are bounded, then $R(x,\omega)$ satisfies the three conditions in Lemma EC.3.*

To calculate $\nabla R(x,\omega)$, several gradients are introduced as follows. From Equation (4), the gradient of the used capacity of *seru* i is obtained from order o. $\nabla q_o^i(x,\omega)$, $1 \le i \le I$ and $1 \le o \le O$. $\nabla q_o^i(x,\omega)$ is a column vector with I components.

$$\nabla q_o^i(x,\omega) = \begin{pmatrix} \frac{\partial}{\partial x_o^1} q_o^i(x,\omega) \\ \vdots \\ \frac{\partial}{\partial x_o^I} q_o^i(x,\omega) \end{pmatrix},$$

where a component $\frac{\partial}{\partial x_o^l} q_o^i(x,\omega)$, $1 \le l \le I$, is defined as follows.
If $r_i < \lambda_o + 1$, then

$$\frac{\partial}{\partial x_o^\ell} q_o^i(x,\omega) = \begin{cases} -\mathbb{I}_{\widetilde{m} < d_o \tau_o \le m}, & \text{if } r_i > r_\ell; \\ \mathbb{I}_{d_o \tau_o > m}, & \text{if } r_i = r_\ell; \\ 0, & \text{if } r_i < r_\ell. \end{cases} \tag{EC.2}$$

If $r_i = \lambda_o + 1$, then $\frac{\partial}{\partial x_o^\ell} q_o^i(x,\omega) = 0$. Therefore, the gradient of capacity used for a *seru* system by order o, $\nabla q_o(x,\omega)$, is an $I \times I$ matrix given as $\nabla q_o(x,\omega) = [\nabla q_o^1(x,\omega), \ldots, \nabla q_o^I(x,\omega)]$.

The SGA finds x that meets the equation $\mathbb{E}[\nabla R(x,\omega)] = 0$. $\nabla R(x,\omega)$ is a vector with I components. $\nabla R(x,\omega) = [\nabla R(x^1,\omega), \ldots, \nabla R(x^I,\omega)] = \boldsymbol{h} \nabla q(x,\omega) - \boldsymbol{e}$, where $\nabla q(x,\omega)$ is the capacity gradient of the *seru* system on a sample path ω, $\boldsymbol{h}$ and $\boldsymbol{e}$ is the revenue and cost of using the *seru* system. $\nabla q(x,\omega)$ is calculated using a backward approach, which starts from the capacity gradient of the final order O, $\nabla q_O(x,\omega)$, to the capacity gradient of order 1, $\nabla q_1(x,\omega)$. The backward procedure is as follows. $\mathbf{I}$ is the $I \times I$ identity matrix.

Backward Procedure to Calculate the Capacity Gradient

Step 1. Calculate the capacity gradient of the final order O using $\nabla q_O(x,\omega)$.

Step 2. $o = O - 1$.

Step 3. Calculate the capacity gradient of order o using $\nabla q_o(x,\omega) + [\mathbf{I} - \nabla q_o(x,\omega)] \times \nabla q_{o+1}(x,\omega)$, $o = O-1, ..., 1$.

Step 4. $o = o - 1$. If $o = 1$, stop. Otherwise, go to Step 3.

Step 5. Calculate the capacity gradient on sample path ω using $\nabla q(x,\omega) = \nabla q_1(x,\omega)$.

EC.2.1.2. Convergence of the Stochastic Gradient Algorithm

Convergence of the stochastic gradient algorithm is now proved. The following convergence Theorem EC.1 is used in Theorem EC.2 to prove that the SG Algorithm converges.

THEOREM EC.1. *Consider a step of an algorithm $r_{t+1} = r_t + \gamma_t s_t$, where step sizes γ_t are nonnegative and satisfy $\sum_{t=0}^{\infty} \gamma_t = \infty$ and $\sum_{t=0}^{\infty} \gamma_t^2 < \infty$. Under the following Assumption 1, the following three statements hold with probability 1.*

(a) The sequence $f(r_t)$ converges.

(b) $\lim_{t\to\infty} \nabla f(r_t) = 0$.

(c) Every limit point of r_t is a stationary point of f.

Assumption 1. Function $f:\Re^n \to \Re$ has the following four properties.
(a) $f(r) \geq 0$ for all $r \in \Re^n$.
(b) (Lipschitz continuity of ∇f) Function f is continuously differentiable. There exists some constant L such that

$$||\nabla f(r) - \nabla f(\bar{r})|| \leq L||r - \bar{r}||, \forall r, \bar{r} \in \Re^n.$$

(c) (Pseudogradient property) There exists a positive constant c such that

$$c||\nabla f(r_t)||^2 \leq -\nabla f(r_t)' E[s_t|\mathcal{F}_t], \forall t,$$

where $\mathcal{F}_t = \{r_0, \ldots, r_t, s_0, \ldots, s_{t-1}, \gamma_0, \ldots, \gamma_t\}$. $\mathcal{F}_t$ contains the history of the algorithm until time t.
(d) There exist positive constants K_1 and K_2 such that

$$E[||s_t||^2|\mathcal{F}_t] \leq K_1 + K_2||\nabla f(r_t)||^2, \forall t.$$

Lemma EC.7 and Theorem EC.2 prove that the SG algorithm converges.

Lemma EC.7. *If f and g are Lipschitz with modulus F and G, and a and b are scalars, then $af + bg$ is Lipschitz with modulus $aF + bG$.*

Theorem EC.2. *The **Stochastic Gradient Algorithm** converges to a stationary point.*

Since the profit function is not quasiconcave, the SG Algorithm converges to a stationary point, which may be the global optimum, maybe a suboptimal point, or maybe a saddle point.

EC.2.1.3. SGA, Non-Convex Optimaztion, and the Dynamic JIT-OS Problem

Non-convex optimization is a key topic in artificial intelligence research. Many machine learning models, especially deep neural networks, rely on non-convex loss functions with multiple local minima and saddle points. This complexity requires specialized optimization techniques. A recent review (Fotopoulos et al. 2024) highlights the importance of non-convex optimization in modern machine learning, as it helps reduce computational costs while improving model performance.

SGAs are widely used to optimize non-convex problems. Despite the challenges of non-convex optimization, SGAs have shown strong empirical performance. A design for SGA is to escape saddle points and converge to a local optimum. However, there is no guarantee that an SGA will always reach a local optimum. Recent research (Fang et al. 2019, Katende and Kasumba 2024) explores improving SGAs by using second-order methods, such as the Hessian matrix, to help escape saddle points in smooth curve functions.

Next, we discuss the structure of our objective function $R(x, \omega)$, which is a piecewise-linear, high-dimensional, non-quasiconcave function. Within each region of the decision space, the objective function is purely linear and thus has a zero Hessian. Consequently, any Hessian-based method for escaping saddle points mentioned above (Katende and Kasumba 2024) cannot be applied, because those methods rely on nonzero second derivatives for identifying saddle neighborhoods. In short, each "piece" of the objective function $R(x, \omega)$ is given by linear expressions of *seru*

capacities—once *seru* ranks and capacity allocation rules are fixed for an order, the associated profit contribution is a linear function of the decision variables. Summing over orders and *serus* preserves this piecewise-linear structure. However, $R(x,\omega)$ is not globally linear, because the rank and capacity usage indicators produce kinks at region boundaries. A "kink" is a boundary where one linear piece meets another, causing a jump or change in the slope of the function. These kink points are nondifferentiable, meaning that standard gradients do not exist there—one should instead use subgradients or other nonsmooth approaches. Also, having many linear pieces (one per combination of *seru* ranks and capacity-threshold conditions) makes $R(x,\omega)$ behave like a collection of multiple linear functions stitched together. This complicates the optimization, as an algorithm can encounter many local maxima or plateau regions. Conventional smooth or Hessian-based methods fail at the kinks, and subgradient or specialized nonsmooth algorithms are needed to navigate the piecewise-linear $R(x,\omega)$.

We define the subgradients in Section EC.2.1.1 specifically to exploit the rank-based, piecewise-linear structure of the JIT-OS problem. In each region (fixed *seru* ranks and capacity-usage inequalities), the usage q_o^i is a linear function of the decision variables. Subgradients are computed by taking partial derivatives of these linear expressions with respect to the relevant capacities, carefully incorporating indicator-based conditions (such as $\widetilde{m} < d_o\tau_o \leq m$). This piecewise approach ensures that our SGA can move in a valid direction even at "kink" boundaries—where classical gradients do not exist—and so find candidate solutions. However, this subgradient construction is tailored to the JIT-OS environment: it hinges on the rank-priority and capacity-spillover logic unique to *seru* assignments, so it may not extend to other piecewise-linear or nonsmooth problems. Indeed, further research is needed to analyze theoretical convergence for this high-dimensional, non-concave family of piecewise-linear programs, and to develop more general methods for navigating large numbers of kinks and local optima.

While Theorem EC.2 establishes convergence of the SGA to a stationary point, we acknowledge that such convergence—under mild conditions—is a well-established result in the non-convex optimization literature. See Bottou et al. (2018) and Li and Lin (2023). Our objective here is not to claim novelty in the convergence guarantee per se, but rather to demonstrate how standard gradient-based methods can be effectively applied to the dynamic, high-dimensional, non-quasiconcave structure of the JIT-OS problem. In this sense, the SGA serves as a representative example of a broader class of gradient-based algorithms that are viable under our problem setting. Moreover, due to the piecewise-linear and nonsmooth nature of our objective function $R(x,\omega)$, we develop a tailored subgradient construction that respects the combinatorial structure of *seru* assignments, ensuring that directional updates remain valid even at nondifferentiable kink points. While our implementation of SGA is specific to the JIT-OS model, we see value in future work comparing its empirical performance to alternative nonsmooth optimization methods—particularly relative to the Newsvendor benchmark—under varying capacity scenarios and structural constraints. (We thank a reviewer for raising these points.)

EC.2.2. Assignment Algorithms

Three greedy algorithms, Profit Fulfillment Maximization Algorithm (**PFM Algorithm**), Skill Waste Minimization Algorithm (**SWM Algorithm**), and Labor Cost Minimization Algorithm (**LCM Algorithm**) are proposed to find and choose different optima. For example, if all skill levels are covered or are enough, profit could have more importance. Being at capacity, skill level may be the best way to improve profit. A high labor cost may have a big impact on profit. **PFM Algorithm** maximizes the profit fulfillment ratio. **SWM Algorithm** minimizes skill waste. **LCM Algorithm** minimizes labor cost. Managers can choose which algorithm to use in different scenarios. The **LCM Algorithm** has been given in Section 5, we give the other two algorithms and Pseudocodes as follows.

In the **PFM Algorithm**, $h_i = \sum_{j=1}^{J} p_j s_i^j$, which is the maximum revenue that a *seru* can provide by assembling a product that exactly matches its skill vector. This is the maximum revenue that a *seru* can provide from assembling one product, when making full use of its skills. Define $\xi_o^i = \big(b_o - e_i\big)/\big(h_i - e_i\big)$ as the profit fulfillment ratio if order o is assigned to *seru* i, where $h_i - e_i$ is the maximum profit from *seru* i assembling one product and $b_o - e_i$ is the profit that *seru* i provides by assembling one product of order o.

Now these three greedy algorithms are given. Denote U_o as the remaining production requirements of order o that have not yet been satisfied, W_i as the remaining capacity of *seru* i, $P(o,x)$ as the profit obtained by assembling order o under initial capacity x, $A(o,x)$ as the set of *serus* with which order o is assembled, and ρ_o^i as the skill waste incurred by using *seru* i to assemble order o, $\rho_o^i = \sum_{j=1}^{J}(s_i^j - v_o^j)$.

Profit Fulfillment Maximization Algorithm

Step 1. For new order o, form set $\mathcal{G}_o$ whose elements are *serus* that can assemble order o.

Step 2. From $\mathcal{G}_o$, among all *serus* that have positive capacity, *seru* i has the highest profit fulfillment ratio ξ_o^i. Assemble order o on *seru* i as much as possible. (If two or more *serus* have the same profit fulfillment value, the *seru* with lowest labor cost assembles order o.)

Step 3. Update the remaining production requirements that have not yet been satisfied for order o (U_o), the available capacity of *seru* i (W_i), the profit $P(o,x)$, and the current solution set $A(o,x)$.

Step 4. If the production period ends or there is no more *seru* capacity, STOP. Otherwise, if the unsatisfied requirements of order o are 0, go to **Step 1**. Otherwise, go to **Step 2**.

Skill Waste Minimization Algorithm

Step 1. For new order o, form set $\mathcal{G}_o$ whose elements are *serus* that can assemble order o.

Step 2. From $\mathcal{G}_o$, among all *serus* that have positive capacity, *seru* i has the lowest skill waste ρ_o^i. Assemble order o on *seru* i as much as possible. (If two or more *serus* have the same skill waste value, the *seru* with lowest labor cost assembles order o.)

Step 3. Update the remaining production requirements that have not yet been satisfied for order o (U_o), the available capacity of *seru* i (W_i), the profit $P(o,x)$, and the current solution set $A(o,x)$.

Step 4. If the production period ends or there is no more *seru* capacity, STOP. Otherwise, if the unsatisfied requirements of order o are 0, go to **Step 1**. Otherwise, go to **Step 2**.

Algorithm 1 Profit Fulfillment Maximization (PFM) Algorithm

Sort the *serus* in nondecreasing order of labor cost:
  Find a permutation π such that $e_{\pi(1)} \le e_{\pi(2)} \le \cdots \le e_{\pi(I)}$.
while a new order o arrives with demand d_o **do**
  Initialize $U_o \leftarrow d_o$ and $W_i \leftarrow x_o^i$.
  Determine the set $\mathcal{G}_o$, denote $\lambda_o = |\mathcal{G}_o|$.
  for $i \in \mathcal{G}_o$ **do**
    Calculate the value of profit fulfillment $\xi_o^i = (b_o - e_i)/(h_i - e_i)$.
  end for
  Sort the *serus* in $\mathcal{G}_o$ in nonincreasing order of ξ_o^i:
    Find a permutation ϕ in $\mathcal{G}_o$ so that $\xi_o^{\phi(1)} \ge \xi_o^{\phi(2)} \ge \cdots \ge \xi_o^{\phi(\lambda_o)}$. When $\xi_o^i = \xi_o^k$, and $e_i \le e_k$, let $\phi(i) \le \phi(k)$.
  for $i = 1$ to λ_o **do**
    if $W_{\phi(i)} > 0$ and $U_o > 0$ **then**
      $t \leftarrow \min\{U_o, W_{\phi(i)}\}$.
      Assemble t products of order o in *seru* $\phi(i)$. Update $A(o,x)$.
      Update: $U_o \leftarrow U_o - t$, $W_{\phi(i)} \leftarrow W_{\phi(i)} - t$, $P(o,x) = P(o,x) + t\theta_{\phi(i),o}$.
    end if
  end for
end while
Output: $P(o,x)$ and the associated solution $A(o,x)$.

Algorithm 2 Skill Waste Minimization (SWM) Algorithm

Sort the *serus* in nondecreasing order of labor cost:
  Find a permutation π such that $e_{\pi(1)} \le e_{\pi(2)} \le \cdots \le e_{\pi(I)}$.
while a new order o arrives with demand d_o **do**
  Initialize $U_o \leftarrow d_o$ and $W_i \leftarrow x_o^i$.
  Determine the set $\mathcal{G}_o$, denote $\lambda_o = |\mathcal{G}_o|$.
  for $i \in \mathcal{G}_o$ **do**
    Calculate the value of skill waste $\rho_o^i = \sum_{j=1}^{J}(s_i^j - v_o^j)$.
  end for
  Sort the *serus* in $\mathcal{G}_o$ in nondecreasing order of ρ_o^i:
    Find a permutation ψ in $\mathcal{G}_o$ so that $\rho_o^{\psi(1)} \le \rho_o^{\psi(2)} \le \cdots \le \rho_o^{\psi(\lambda_o)}$. When $\rho_o^i = \rho_o^k$, and $e_i \le e_k$, let $\psi(i) \le \psi(k)$.
  for $i = 1$ to λ_o **do**
    if $W_{\psi(i)} > 0$ and $U_o > 0$ **then**
      $t \leftarrow \min\{U_o, W_{\psi(i)}\}$.
      Assemble t products of order o in *seru* $\psi(i)$. Update $A(o,x)$.
      Update: $U_o \leftarrow U_o - t$, $W_{\psi(i)} \leftarrow W_{\psi(i)} - t$, $P(o,x) = P(o,x) + t\theta_{\psi(i),o}$.
    end if
  end for
end while
Output: $P(o,x)$ and the associated solution $A(o,x)$.

EC.3. Proofs of Theorems

THEOREM 1. *A general static JIT-OS problem can be optimized in polynomial-time.*

Proof of Theorem 1. If the customer orders are known at the beginning of the production period, it is easy to see that in order to assemble all the customer orders and obtain a maximum profit, the total capacity of the *serus* must be exactly equal to the total demand of the customer orders. Therefore, we can transform the general static JIT-OS problem into the following minimum cost flow problem (see Figure EC.1) where the customer orders are the supply nodes, the given

Algorithm 3 Labor Cost Minimization (LCM) Algorithm

while a new order o arrives with demand d_o **do**
  Initialize $U_o \leftarrow d_o$ and $W_i \leftarrow x_o^i$.
  Determine the set $\mathcal{G}_o$, denote $\lambda_o = |\mathcal{G}_o|$.
  for $i \in \mathcal{G}_o$ **do**
    Sort the *serus* in $\mathcal{G}_o$ in nondecreasing order of labor cost:
      Find a sequence so that $e_{[1]} \le e_{[2]} \le \cdots \le e_{[\lambda_o]}$.
  end for
  for $i = 1$ to λ_o **do**
    if $W_{[i]} > 0$ and $U_o > 0$ **then**
      $t \leftarrow \min\{U_o, W_{[i]}\}$.
      Assemble t products of order o in *seru* $[i]$. Update $A(o,x)$.
      Update: $U_o \leftarrow U_o - t$, $W_{[i]} \leftarrow W_{[i]} - t$, $P(o,x) = P(o,x) + t\theta_{[i],o}$.
    end if
  end for
end while
Output: $P(o,x)$ and the associated solution $A(o,x)$.

seru types are the transhipment nodes, and a dummy node t is introduced as the demand node. An arc between customer order o and *seru* type i is created if $\alpha_{io} = 1$.

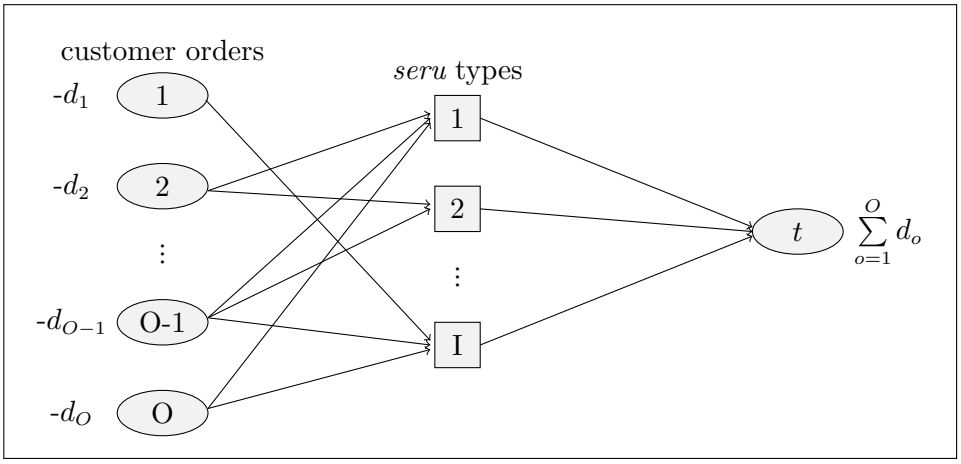

Figure EC.1 The minimum cost flow used in the proof of Theorem 1.

We set the cost per-unit-flow between customer order o and *seru* type i to $-\theta_{io}$, set the cost per-unit-flow between *seru* types and the dummy sink t to 0, and set the capacity of each arc in the graph to $+\infty$. Then the optimal flow from customer orders to *seru* types in Figure EC.1 determines the assignment of customer orders, and the optimal flow from *seru* types to sink t determine the optimal capacity of *serus*. Note that the above minimum cost flow problem can be solved by any linear programming algorithm with an integral optimal solution, so the claim follows. □

THEOREM 2. *For an arbitrary sample path ω, total capacity consumption $\sum_{o\in\mathcal{O}} q_o^i(x,\omega)$ of seru i is submodular and satisfies diminishing returns with respect to the initial capacity of an arbitrary seru j $(i \neq j)$, x^j on X.*

Proof of Theorem 2. Using Lemmas EC.1, EC.2, and 1, we can get the result. □

THEOREM 3. *For an arbitrary sample path ω, profit $R(x^i,\omega)$ from seru i satisfies diminishing returns with respect to the initial capacity of another seru j $(i \neq j)$, x^j on X.*

Proof of Theorem 3. Using Theorem 2, we can get the result. □

THEOREM 4. *For an arbitrary sample path ω, profit of seru system $R(x^i,\omega)$ from seru i, satisfies strictly diminishing returns with respect to the initial capacity of an arbitrary seru j $(i \neq j)$, x^j on X, if for an arbitrary order o on ω, $r_j < r_i \leq \lambda_o$ and $m \leq d_o\tau_o \leq m+\beta_j$. That is, $R(x_{-ij},\widehat{x}^i,\widehat{x}^j,\omega) - R(x_{-ij},x^i,\widehat{x}^j,\omega) < R(x_{-ij},\widehat{x}^i,x^j,\omega) - R(x_{-ij},x^i,x^j,\omega)$.*

Proof of Theorem 4. Using Lemma EC.8, we can get the result. □

THEOREM 5 *There exists a sample path ω on which the profit function $R(x,\omega)$ is not quasiconcave.*

Proof of Theorem 5. To show that $R(x,\omega)$ is not quasiconcave, it suffices to show that there is a sample path ω that makes the convex combination $R(\alpha x+\beta y,\omega)$, with $\alpha,\beta \geq 0$ and $\alpha+\beta=1$, of two initial *seru* capacities x, y to be smaller than both $R(x,\omega)$ and $R(y,\omega)$. That is $R(\alpha x+\beta y,\omega) < \min\{R(x,\omega),R(y,\omega)\}$ (Lay 1982).

Consider an SPS with 4 *serus*. Let $x=(8,2,0,1)$ and $y=(0,8,2,1)$ be two different initial *seru* capacities. Suppose that $\alpha=0.9$ and $\beta=0.1$. Then $\alpha x+\beta y=(7.2,2.6,0.2,1)$. Assume that the cost to use *seru* i, $e_i=\$100$, $i=1,2,3$, and 4, and the profit of order o, $b_o=\$200$, for $o\in\mathcal{O}$.

Consider a sample path ω, with $O=3$ orders as follows. Assume that order 1 can only be produced by *seru* 4, order 2 by *serus* 1 and 3 in decreasing rank, and order 3 by *serus* 1 and 2 in decreasing rank. Suppose that demands are $d_1=1$, $d_2=2$, and $d_3=8$. Then $R(x,\omega)=R(y,\omega)=200\times 11-100\times 11=\$1100$ and $R(\alpha x+\beta y,\omega)=200\times 10.8-100\times 11=\$1060$. □

THEOREM 6 *Order assignment for the dynamic JIT-OS Problem is NP-hard.*

Proof of Theorem 6. We show that the order assignment for the dynamic JIT-OS Problem is NP-hard even when there is only one order with only one product of demand and only two *serus* can assemble this order. The problem of finding a minimum vertex cover in an undirected graph $G=(V,E)$ is NP-hard (Garey and Johnson 1979). Given an integer parameter k, the vertex cover problem is to find whether there exists a subset of vertices $V'\subset V$ of cardinality at most k, such that each edge $e\in E$ is incident to some vertex in V'. Given any instance $\mathcal{A}$ of the minimum vertex cover problem, $V=\{v_1,\ldots,v_I\}$. An instance $\mathcal{A}'$ of the dynamic JIT-OS Problem is constructed as follows. Let the number of *serus* be $I=|V|$, one *seru* corresponding to each vertex. Let w be the order with only one unit of demand that requires one unit of capacity. The two *serus* that can assemble this product are i and j, $\{(i,j):(v_i,v_j)\in E\}$, which occurs with probability $1/|E|$. The available capacity of the total number of *serus* is $\sum_{i=1}^{I} x_1^i = k$ units. This order can be assembled if and only if there is a vertex cover with size k at most in instance $\mathcal{A}$.

Suppose that there is a vertex cover $V'\subset V$ such that $|V'|\leq k$. Consider a capacity $x_1=(x_1^1,\ldots,x_1^i,\ldots,x_1^I)$, where $x_1^i=1$ unit if $v_i\in V'$ and $x_1^i=0$ otherwise. In other words, a single capacity unit of *seru* i is stocked if and only if v_i is a vertex of the vertex cover V'. Clearly the total number of capacity units stocked is at most $\sum_{i=1}^{I} x_1^i = k$. Let

$$\mathbb{E}[R(x_1,w)] = \frac{1}{|E|}\sum_{(v_i,v_j)\in E}\max\{x_1^i,x_1^j\}.$$

Since V' is a vertex cover, for any edge $(v_i, v_j) \in E$, at least one of v_i or v_j is in V'. Therefore, $\max\{x_1^i, x_1^j\} = 1$ for all $(v_i, v_j) \in E$ and $\mathbb{E}[R(x_1, w)] = 1$, which means that an arbitrary order w can be assembled.

Conversely, consider a capacity $x_1 = (x_1^1, \ldots, x_1^i, \ldots, x_1^I)$ that contains at most k units and can assemble order w. The following shows that $V' = \{v_i : x_1^i \geq 1\}$ is a vertex cover. This set consists of at most k vertices. Because the order can be assembled,

$$\mathbb{E}[R(x_1, w)] = \frac{1}{|E|} \sum_{(v_i, v_j) \in E} \min\{\max\{x_1^i, x_1^j\}, 1\} = 1.$$

Thus, it follows that $\max\{x_1^i, x_1^j\} \geq 1$ for any edge $(v_i, v_j) \in E$, implying that at least one of the vertices v_i and v_j belongs to V'. In other words, the set V' is a vertex cover. □

THEOREM 7 *It is NP-hard to approximate order assignment for the dynamic JIT-OS Problem within a factor better than* $1 - 1/e$.

Proof of Theorem 7. We show that it is NP-hard to approximate order assignment for the dynamic JIT-OS Problem within a factor better than $1 - 1/e$, even when there is only one order with only one product of demand. We reduce any instance of the maximum coverage problem to the dynamic JIT-OS Problem. The maximum coverage problem is described as follows: Given a ground set $G = \{e_1, e_2, \ldots, e_I\}$ and its power set $\mathcal{P}(G)$, a family set $\mathcal{F} \subseteq \mathcal{P}(G)$, and an integer k. The maximum coverage problem is to find a subset $g \subseteq G$ with k elements such that the cover $\widehat{g} = |\{f \in \mathcal{F} : f \cap g \neq \emptyset\}|$ is maximized.

Consider a maximum coverage instance $\mathcal{A}$. An instance $\mathcal{A}'$ of the dynamic JIT-OS Problem is as follows. Let the number of *serus* be $I = |G|$, one *seru* corresponding to each element in G. Let w be the order with only one unit of demand that requires one unit of capacity. The *serus* that can assemble this product construct a set f, which is an element of $\mathcal{F}$ with occurence probability $1/|\mathcal{F}|$. The available capacity of the total number of *serus* is $\sum_{i=1}^{I} x_1^i = k$ units.

Let g be an arbitrary subset of G, $g \subseteq G$ and $|g| = k$, generating the maximum coverage instance $\mathcal{A}'$. Consider a capacity $x_1 = (x_1^1, \ldots, x_1^i, \ldots, x_1^I)$, where $x_1^i = 1$ unit if $e_i \in g$ and $x_1^i = 0$ otherwise. In other words, a single capacity unit of *seru* i is stocked if and only if e_i is an element of g. So the total number of capacity units stocked is at most $\sum_{i=1}^{I} x_1^i \leq |g| = k$.

To finish the proof, it is sufficient to show that x_1 generates an expected revenue of $\widehat{g}/|\mathcal{F}|$. For an arbitrary subset $f \subset G$, let $\mathbb{I}_{(f,g)}$=1 if $f \cap g \neq \emptyset$ and $\mathbb{I}_{(f,g)} = 0$ otherwise. In other words, $\mathbb{I}_{(f,g)}$ is an indicator to judge whether f and g have a non-empty intersection.

$$\mathbb{E}[R(x_1, w)] = \sum_{f \in \mathcal{F}} \left[\frac{1}{|\mathcal{F}|} \times \mathbb{I}_{(f,g)} \right] = \frac{\widehat{g}}{|\mathcal{F}|}.$$

Given a capacity $x_1 = (x_1^1, \ldots, x_1^I)$ that has at most k non-zero units, construct a set $g \subseteq G$, $|g| \leq k$, such that $\widehat{g} = |\mathcal{F}| \times \mathbb{E}[R(x_1, w)]$. To this end, let $g = \{e_i : x_1^i = 1\}$. Then, $|g| \leq k$ and

$$\widehat{g} = \sum_{f \in \mathcal{F}} \mathbb{I}_{(f,g)} = |\mathcal{F}| \times \mathbb{E}[R(x_1, w)].$$

Finally, Feige (1998) has shown that it is NP-hard to approximate within a factor better than $1-1/e$ for a maximum coverage problem, so our claim follows. □

THEOREM 8 *For any seru, the* **LCM** *algorithm has a competitive ratio of*

$$\frac{m_{\min}}{m_{\max}} \times \frac{1-m_{\max}}{1-m_{\min}}. \tag{EC.3}$$

Proof of Theorem 8. Consider e_i as the labor cost associated with an arbitrary *seru*. In the context of worst-case scenarios, the competitive ratio manifests as $(b_{\min}-e_i)/(b_{\max}-e_i)$. Given that $m_{\max}$ is defined as $(b_{\max}-e_i)/b_{\max}$, it follows that $b_{\max}$ can be expressed as $e_i/(1-m_{\max})$. In a parallel manner, $b_{\min}$ equates to $e_i/(1-m_{\min})$. By substituting these expressions into the formula for the competitive ratio, we arrive at the desired conclusion. □

- Further Discussion on the Input-Dependent Competitive Ratio

The competitive ratio (CR) measures how well an online algorithm performs compared to an ideal offline algorithm that knows all future information. For the single-price assortment revenue management problem, Golrezaei et al. (2014) developed an online algorithm that achieves a CR approaching $1-1/e$ in the asymptotic regime, assuming large inventories. Their algorithm carefully manages inventory, penalizing items with lower remaining stock, to maximize expected revenue even without knowledge of future customer arrivals.

However, when each item has multiple possible prices (i.e., multiple fare classes), Ma and Simchi-Levi (2020) showed that it becomes impossible to achieve a universally fixed, nonzero CR for all input parameters. This impossibility arises because without carefully chosen "booking limits" (i.e., rejecting certain customers), an online algorithm might sell too much inventory at low prices, missing later higher-price opportunities. In the worst-case case, Ma and Simchi-Levi (2020) illustrated that the CR of Golrezaei et al. (2014) changes from input-independent $1-1/e$ to input-dependent ratio $\frac{r^{(L)}}{r^{(H)}}$, where $r^{(L)}$ and $r^{(H)}$ are the lowest and highest prices of an item.

We show that our CR $(\frac{m_{\min}}{m_{\max}} \times \frac{1-m_{\max}}{1-m_{\min}})$ is consistent with Ma and Simchi-Levi's CR $(\frac{r^{(L)}}{r^{(H)}})$.

In assortment revenue management, an item is similar to a *seru* in a *seru* production system. Like Golrezaei's method, our **LCM** algorithm follows a First-Come-First-Serve policy without booking limits. For any *seru* i, its lowest and highest revenues are $b_{\min}$ and $b_{\max}$, respectively. Since our objective function is profit, Ma and Simchi-Levi's CR $(\frac{r^{(L)}}{r^{(H)}})$, should correspond to the ratio of the lowest profit to the highest profit. This relationship has been proven in Theorem 8, leading to the result $CR = \frac{m_{\min}}{m_{\max}} \times \frac{1-m_{\max}}{1-m_{\min}}$.

For more theoretical analysis on input-dependent CRs, see Mehta (2013).

(We thank a reviewer for raising these points.)

THEOREM EC.2 *The* ***Stochastic Gradient Algorithm*** *converges to a stationary point.*

Proof of Theorem EC.2. $f(x) = \mathbb{E}[R(x,\omega)]$. For condition (a), $f(x) = \mathbb{E}[R(x,\omega)] \geq 0$, which is obviously true. For condition (b), from Lemma EC.6, it is sufficient to show that there exists a constant L such that

$$\mathbb{E}\left[||\nabla R(x^i,\omega) - \nabla R(y^i,\omega)||\right] \leq L||x-y||.$$

Note that $R(x^i,\omega)=\sum_{o\in\mathcal{O}} b_o[q_o^i(x,\omega)/\tau_o]-e_i x^i$. From Lemma EC.7, it is sufficient to show that there exists a constant L' such that

$$\mathbb{E}\left[||\nabla q_o^i(x,\omega)-\nabla q_o^i(y,\omega)||\right]\le L'||x-y||.$$

From Equation (EC.2), the partial derivatives of the capacity used function satisfy

$$\frac{\partial}{\partial x_o^\ell}q_o^i(x,\omega)\in\begin{cases}\{0,1\}, & \text{if } i=\ell;\\ \{-1,0\}, & \text{if } i\neq\ell.\end{cases}$$

Because

$$P(\nabla q_o^i(x,\omega)\neq\nabla q_o^i(y,\omega))\le\sum_{\ell=1}^{I}P(\frac{\partial}{\partial x_o^\ell}q_o^i(x,\omega)\neq\frac{\partial}{\partial x_o^\ell}q_o^i(y,\omega)),\tag{EC.4}$$

we have

$$\mathbb{E}\left[||\nabla q_o^i(x,\omega)-\nabla q_o^i(y,\omega)||\right]\le\sqrt{I}\times P(\nabla q_o^i(x,\omega)\neq\nabla q_o^i(y,\omega)).\tag{EC.5}$$

The right hand side of Equation (EC.4) is analyzed as the following three cases.

Case 1. $\underline{r_i>r_\ell}$.

$$\begin{aligned}P(\frac{\partial}{\partial x_o^\ell}q_o^i(x,\omega)\neq\frac{\partial}{\partial x_o^\ell}q_o^i(y,\omega))\le&P(x_o^{[r_i-1]}+\cdots+x_o^{[1]}<d_o\tau_o)\times P(y_o^{[r_i-1]}+\cdots+y_o^{[1]}>d_o\tau_o)\\&+P(x_o^{[r_i]}+\cdots+x_o^{[1]}>d_o\tau_o)\times P(y_o^{[r_i]}+\cdots+y_o^{[1]}<d_o\tau_o).\end{aligned}$$

From Lemma EC.4, $\left|\left|x_o^\ell-y_o^\ell\right|\right|\le||x_o-y_o||\le M||x-y||$. For all $\ell=1,\ldots,I$,

$$\left|\left|\sum_{\ell=1}^{r_i}(x_o^{[\ell]}-y_o^{[\ell]})\right|\right|\le\sum_{\ell=1}^{r_i}\left|\left|x_o^{[\ell]}-y_o^{[\ell]}\right|\right|\le Mr_i||x-y||.$$

Let $F_o(\cdot)$ denote the marginal distribution of demand d_o, $o=1,\ldots,O$. It is clear that $F_o(\cdot)$ is a Lipschitz function. That is, $||F_o(x)-F_o(y)||\le K||x-y||$. Therefore,

$$\begin{aligned}P(\frac{\partial}{\partial x_o^\ell}q_o^i(x,\omega)\neq\frac{\partial}{\partial x_o^\ell}q_o^i(y,\omega))\le&||F_o(x)-F_o(y)||\left(\left|\left|\sum_{\ell=1}^{r_i}(x_o^{[\ell]}-y_o^{[\ell]})\right|\right|+\left|\left|\sum_{\ell=1}^{r_i-1}(x_o^{[\ell]}-y_o^{[\ell]})\right|\right|\right)\\\le&2KMr_i||x-y||\le 2KMI||x-y||.\end{aligned}$$

Case 2. $\underline{r_i=r_\ell}$.

$$P(\frac{\partial}{\partial x_o^\ell}q_o^i(x,\omega)\neq\frac{\partial}{\partial x_o^\ell}q_o^i(y,\omega))\le P(x_o^{[r_i]}+\cdots+x_o^{[1]}>d_o\tau_o)\times P(y_o^{[r_i]}+\cdots+y_o^{[1]}<d_o\tau_o)\le KMI||x-y||.$$

Case 3. $\underline{r_i<r_\ell}$.

$$P(\frac{\partial}{\partial x_o^\ell}q_o^i(x,\omega)\neq\frac{\partial}{\partial x_o^\ell}q_o^i(y,\omega))\le||x-y||.$$

From Equation (EC.5) and Cases 1, 2, and 3, let $L'=\sqrt{I^3}(3KMI+1)$. This proves the Lipschitz condition of $\nabla\mathbb{E}[R(x,\omega)]$.

For condition (c), fix $c=1$. Note that $\mathbb{E}[\nabla R(r_t,\omega_t)|\mathcal{F}_t]=\mathbb{E}[\nabla R(r_t,\omega_t)]$. The result is obtained from Lemma EC.6.

For condition (d), it is sufficient to show that there exist two positive constants K_1 and K_2 such that

$$||\nabla R(r_t,\omega_t)||^2=\left|\left|\sum_{i\in\mathcal{I}}\sum_{o\in\mathcal{O}}[b_o/\tau_o]\nabla q_o^i(r_t,\omega_t)-\sum_{i\in\mathcal{I}}e_i\nabla x^i\right|\right|^2\le K_1+K_2||\nabla R(r_t,\omega_t)||^2.$$

For all $t=0,1,2,\cdots$, since $\min\{\frac{\partial}{\partial x_o^i}q_o^i(r_t,\omega_t)\}=0$ and $\min\{\frac{\partial}{\partial x_o^\ell}q_o^i(r_t,\omega_t)\}=-1$, for $i\neq\ell$, let

$$K_1=\left(\left|\left|-\left(\sum_{o\in\mathcal{O}}b_o/\tau_o\right)\mathbf{1}-\sum_{i\in\mathcal{I}}e_i\right|\right|\right)^2$$

and K_2 be an arbitrarily small and positive number. Then condition (d) holds. □

The proof of Theorem EC.1 can be found in Bertsekas and Tsistiklis (1996).

EC.4. Proofs of Lemmas

LEMMA 1 *For order o assembled on an arbitrary sample path ω, capacity consumption $q_o^i(x,\omega)$ of seru i is submodular and satisfies diminishing returns with respect to the initial capacity of another seru j $(i\neq j)$, x^j on X.*

Proof of Lemma 1. Let ω be an arbitrary sample path. Let $(x_{-ij},\widehat{x}^i,\widehat{x}^j)$ be the vector obtained from vector x by replacing components i and j with $\widehat{x}^i$ and $\widehat{x}^j$, respectively. Suppose that $\widehat{x}^i\ge x^i$ and $\widehat{x}^j\ge x^j$, $i\neq j$. Denote $\widehat{q}_o^i(\gamma,\delta)=q_o^i(x_{-ij},x^i+\gamma,x^j+\delta,\omega)$, where $\gamma,\delta\ge 0$. Define $\beta_i=\widehat{x}^i-x^i$ and $\beta_j=\widehat{x}^j-x^j$.

The problem defined by $\widehat{q}_o^i(0,\delta)$ is examined. A parametric analysis on δ is performed to find the relationship between $\widehat{q}_o^i(0,0)$ and $\widehat{q}_o^i(0,\beta_j)$. From Equation (4), $\widehat{q}_o^i(0,0)=q_o^i(x,\omega)$ and $\widehat{q}_o^i(0,\beta_j)$ are defined as follows.

$$\widehat{q}_o^i(0,\beta_j)=\begin{cases}\mathbb{I}_{\widetilde{m}+\beta_j\le d_o\tau_o<m+\beta_j}(d_o\tau_o-\widetilde{m}-\beta_j)+\mathbb{I}_{m+\beta_j\le d_o\tau_o}x_o^i, & r_j<r_i\le\lambda_o;\\ \widehat{q}_o^i(0,0), & \text{otherwise.}\end{cases}\tag{EC.6}$$

From Equation (EC.6), $\widehat{q}_o^i(0,0)$ and $\widehat{q}_o^i(0,\beta_j)$ are different only when $r_j<r_i\le\lambda_o$. This difference, defined as $\Delta\widehat{q}_o^i(0,\Delta_\delta)=\widehat{q}_o^i(0,\beta_j)-\widehat{q}_o^i(0,0)$, is calculated by comparing β_j with x_o^i in the following three cases.

Case 1.1 $\beta_j<x_o^i$.

$$\begin{aligned}\Delta\widehat{q}_o^i(0,\Delta_\delta)&=[\mathbb{I}_{\widetilde{m}+\beta_j\le d_o\tau_o<m+\beta_j}(d_o\tau_o-\widetilde{m}-\beta_j)+\mathbb{I}_{m+\beta_j\le d_o\tau_o}x_o^i]-[\mathbb{I}_{\widetilde{m}\le d_o\tau_o<m}(d_o\tau_o-\widetilde{m})+\mathbb{I}_{m\le d_o\tau_o}x_o^i]\\&=\mathbb{I}_{\widetilde{m}\le d_o\tau_o<\widetilde{m}+\beta_j}(\widetilde{m}-d_o\tau_o)+\mathbb{I}_{\widetilde{m}+\beta_j\le d_o\tau_o<m}(-\beta_j)+\mathbb{I}_{m\le d_o\tau_o<m+\beta_j}(d_o\tau_o-\widetilde{m}-\beta_j-x_o^i).\end{aligned}\tag{EC.7}$$

Case 1.2 $\beta_j=x_o^i$.

$$\Delta\widehat{q}_o^i(0,\Delta_\delta)=\mathbb{I}_{\widetilde{m}\le d_o\tau_o<m}(\widetilde{m}-d_o\tau_o)+\mathbb{I}_{m\le d_o\tau_o<m+\beta_j}(d_o\tau_o-\widetilde{m}-\beta_j-x_o^i).\tag{EC.8}$$

Case 1.3 $\underline{\beta_j > x_o^i}$.

$$\begin{aligned}\Delta\widehat{q}_o^i(0,\Delta_\delta) = {} & \mathbb{I}_{\widetilde{m}\le d_o\tau_o<m}(\widetilde{m}-d_o\tau_o)+\mathbb{I}_{m\le d_o\tau_o<\widetilde{m}+\beta_j}(-x_o^i)\\ & +\mathbb{I}_{\widetilde{m}+\beta_j\le d_o\tau_o<m+\beta_j}(d_o\tau_o-\widetilde{m}-\beta_j-x_o^i).\end{aligned} \tag{EC.9}$$

Next the problem defined by $\Delta\widehat{q}_o^i(\gamma,\Delta_\delta)=\widehat{q}_o^i(\gamma,\beta_j)-\widehat{q}_o^i(\gamma,0)$ is studied. A parametric analysis on γ is performed to investigate the monotonicity of $\Delta\widehat{q}_o^i(\gamma,\Delta_\delta)$. There are three cases.

Case 2.1 $\underline{r_i=\lambda_o+1}$.

For all γ and δ, $\widehat{q}_o^i(\gamma,\delta)=0$. Therefore, $\Delta\widehat{q}_o^i(\gamma,\Delta_\delta)=0$, for all γ. In other words, $\Delta\widehat{q}_o^i(\gamma,\Delta_\delta)$ is monotonically nonincreasing in γ.

Case 2.2 $\underline{r_i<r_j}$.

According to Equation (2), $r_i<r_j$ implies that $1\le r_i\le\lambda_o$. By Lemma EC.9, increasing γ is equal to increasing x_o^i. From Equation (4),

$$\widehat{q}_o^i(\gamma,\cdot)=\begin{cases}\mathbb{I}_{d_o\tau_o<x_o^i+\gamma}d_o\tau_o+\mathbb{I}_{d_o\tau_o\ge x_o^i+\gamma}(x_o^i+\gamma), & r_i=1;\\ \mathbb{I}_{\widetilde{m}\le d_o\tau_o<m+\gamma}(d_o\tau_o-\widetilde{m})+\mathbb{I}_{d_o\tau_o\ge m+\gamma}(x_o^i+\gamma), & 1<r_i\le\lambda_o.\end{cases} \tag{EC.10}$$

From Equation (EC.10), δ has no effect on $\Delta\widehat{q}_o^i(\gamma,\Delta_\delta)$, which means that for each γ, $\widehat{q}_o^i(\gamma,\cdot)=\widehat{q}_o^i(\gamma,\delta_1)=\widehat{q}_o^i(\gamma,\delta_2)$, for any arbitrary δ_1,δ_2. So $\Delta\widehat{q}_o^i(\gamma,\cdot)=\Delta\widehat{q}_o^i(\gamma,\Delta_\delta)=0$.

By Equation (EC.6), $\widehat{q}_o^i(0,\cdot)=\widehat{q}_o^i(0,0)=q_o^i(x,\omega)$. So $\Delta\widehat{q}_o^i(0,\cdot)=\Delta\widehat{q}_o^i(0,\Delta_\delta)=0$. Therefore, $\Delta\widehat{q}_o^i(\gamma,\Delta_\delta)$ is monotonically nonincreasing in γ.

Case 2.3 $\underline{r_j<r_i<\lambda_o+1}$.

Since x_o^i can be expressed as a monotonically increasing function of γ, Equations (EC.7), (EC.8), and (EC.9) are clearly monotonically decreasing in γ.

From Cases 2.1, 2.2, and 2.3, $\Delta\widehat{q}_o^i(\gamma,\Delta_\delta)$ is a monotonically nonincreasing function of γ. This means that $\Delta\widehat{q}_o^i(\beta_i,\Delta_\delta)\le\Delta\widehat{q}_o^i(0,\Delta_\delta)$ or $\widehat{q}_o^i(\beta_i,\beta_j)-\widehat{q}_o^i(\beta_i,0)\le\widehat{q}_o^i(0,\beta_j)-\widehat{q}_o^i(0,0)$ or

$$q_o^i(x_{-ij},\widehat{x}^i,\widehat{x}^j,\omega)-q_o^i(x_{-ij},\widehat{x}^i,x^j,\omega)\le q_o^i(x_{-ij},x^i,\widehat{x}^j,\omega)-q_o^i(x_{-ij},x^i,x^j,\omega). \tag{EC.11}$$

Equation (EC.11) can also be written as follows.

$$q_o^i(x_{-ij},\widehat{x}^i,\widehat{x}^j,\omega)-q_o^i(x_{-ij},x^i,\widehat{x}^j,\omega)\le q_o^i(x_{-ij},\widehat{x}^i,x^j,\omega)-q_o^i(x_{-ij},x^i,x^j,\omega). \tag{EC.12}$$

Because ω is an arbitrary sample path, i and j are arbitrary elements, o is an arbitrary order, Equations (EC.11) and (EC.12) show that $q_o^i(x,\omega)$ is submodular and satisfies diminishing returns with respect to $x^1,\ldots,x^I$ on X. □

LEMMA 2 *Assume that for all orders, the highest and lowest profit margins are* $m_{\max}=.6$ *and* $m_{\min}=.2$. *Then the worst-case competitive ratio for the LCM Algorithm is 1/6. Also, there exists an instance of the online order-assignment problem for which the LCM Algorithm achieves exactly this 1/6 ratio relative to an optimal offline solution. Hence, the worst-case competitive ratio of 1/6 is tight.*

Proof of Lemma 2. To prove that a competitive ratio (CR) is tight, we need do two things:

(1). Upper Bound (Universal Guarantee): Show that for *every* valid input scenario σ, the LCM Algorithm's profit is at least CR times the offline optimal profit:

$$\text{Profit}_{\text{LCM}}(\sigma) \geq CR \times \text{Profit}_{\text{OPT}}(\sigma),$$

where $\text{Profit}_{\text{LCM}}(\sigma)$ and $\text{Profit}_{\text{OPT}}(\sigma)$ are profits generated by the algorithms of LCM and offline, for instance σ, respectively.

(2). Lower Bound (Existence of a Worst-Case Instance): Construct (or identify) a specific input ω such that LCM's profit on ω is exactly CR times (or arbitrarily close to CR times) the profit of an offline optimal solution:

$$\text{Profit}_{\text{LCM}}(\omega) = CR \times \text{Profit}_{\text{OPT}}(\omega).$$

When both (1) and (2) hold, we conclude that the worst-case ratio is at least CR (because we have an example that hits CR) and at most CR (because no instance can do worse than CR). Hence the ratio is exactly CR, so CR is tight.

We now apply this structure to prove Lemma 2, which states that under $m_{\max} = .6$ and $m_{\min} = .2$, the competitive ratio is 1/6 and there is an input instance that attains 1/6.

We break the proof into two steps. (1). Upper Bound: Show that under these margins (and corresponding costs/revenues), no input can produce a ratio below 1/6. (2). Lower Bound (Example): Construct one input scenario on which LCM's ratio is exactly 1/6.

Part A: The Universal Upper-Bound Argument

A detailed proof of the general formula $CR = \dfrac{m_{\min}}{m_{\max}} \times \dfrac{1 - m_{\max}}{1 - m_{\min}}$ is given by Theorem 8.

1. LCM always chooses the lowest-cost *seru* for the next production.
2. Margins m_o (for an order o) are defined by $(b_o - e_i)/b_o$, where b_o is the order's revenue, and e_i is the production cost using *seru* i.
3. By analyzing how LCM allocates orders vs. how an optimal offline algorithm can reorder or reallocate (knowing future orders in advance), one can show the following.

$$\text{Profit}_{LCM}(\sigma) \;\geq\; CR \times \text{Profit}_{OPT}(\sigma) \quad \text{for all input sequences } \sigma,$$

where $CR = \dfrac{m_{\min}}{m_{\max}} \times \dfrac{1 - m_{\max}}{1 - m_{\min}}$. By substituting $m_{\max} = .6$ and $m_{\min} = .2$ into the CR formula, we get that

$$CR = \frac{.2}{.6} \times \frac{1 - .6}{1 - .2} = \frac{1}{3} \times \frac{.4}{.8} = \frac{1}{3} \times \frac{1}{2} = \frac{1}{6}.$$

Thus, under the given margins, LCM guarantees that

$$\text{Profit}_{LCM}(\sigma) \;\geq\; \frac{1}{6} \times \text{Profit}_{OPT}(\sigma) \quad \text{for any valid instance } \sigma.$$

This completes the universal upper-bound proof: no scenario can make LCM fall below the ratio of 1/6.

Part B: Constructing an "Adversarial Instance" (Lower Bound)

Consider the following instance ω.

- *Seru:* There is one *seru* i with two production skills. The cost of each skill is \$40, so the total production cost for any order assigned to the *seru* is \$80 (since the production cost applies to all skills regardless of usage). The *seru* is designed to assemble exactly one order, meaning its capacity is sufficient to fully satisfy one order. Each order requires the *seru*'s entire capacity to be completely assembled.
- Order 1: Requires only the first skill, with order price (revenue) = \$100, production cost = \$80, profit = revenue - cost = $100 - 80 = \$20$, and profit margin = $\frac{20}{100} = .2$, which matches $m_{\min} = .2$.
- Order 2: Requires both skills, with order price (revenue) = \$200, production cost = \$80, profit = revenue - cost = $200 - 80 = \$120$, and profit margin = $\frac{120}{200} = .6$, which matches $m_{\max} = .6$.
- Order Arrival Sequence: Order 1 arrives first, followed by Order 2.

The LCM Algorithm processes orders online as follows. When order 1 arrives, the set $\mathcal{G}_1$ is formed, the *serus* that can assemble order 1. Since the *seru* has both skills and order 1 requires only the first skill, $\mathcal{G}_1 = \{i\}$ (the single *seru*). Among *serus* in $\mathcal{G}_1$ with positive capacity, select the *seru* with the lowest labor cost. The only *seru* has $W_i = 1 > 0$, and its labor cost is \$80. Assign order 1 to *seru* i as much as possible. Suppose order 1 needs 1 *seru*, and $W_i = 1$ (i.e., *seru* i can assemble only this order, using all of its capacity). Order 1 is fully assigned to *seru* i. Update:

- $U_1 = 0$ (order 1 is fully satisfied).
- $W_i = 1 - 1 = 0$ (capacity is fully used).
- $P(1, x) = \$20$ (profit from order 1).
- $A(1, x) = \{i\}$ (order 1 is assigned to *seru* i).

Unsatisfied requirements of order 1 are 0, so proceed to the next order. When order 2 arrives, $\mathcal{G}_2$ is formed. *Seru* i has both skills, and order 2 requires both, so $\mathcal{G}_2 = \{i\}$. We check *serus* with positive capacity. The *seru*'s capacity is $W_i = 0$, so no *serus* have positive capacity. Thus, order 2 cannot be assigned. No updates occur since no assignment is made. No capacity remains, so the algorithm stops.

Total Profit=Profit$_{LCM}(\omega) = \$20$**.**

Since the LCM Algorithm is online, it assigns order 1 without knowledge of order 2, exhausting the *seru*'s capacity and leaving no room for the more profitable order 2.

The optimal offline algorithm knows both orders in advance and can choose the best assignment given the capacity constraint:

- Assign order 2: Profit = \$120.
- Order 1 is not assigned.

Since the *seru* can process only one order, the optimal choice is order 2, yielding $\text{Profit}_{OPT}(\omega) =$ \$120.

Therefore, for this instance ω, $\text{Profit}_{LCM}(\omega) = \$20$, and $\text{Profit}_{OPT}(\omega) = \$120$.

$$\frac{\text{Profit}_{LCM}(\omega)}{\text{Profit}_{OPT}(\omega)} = \frac{20}{120} = \frac{1}{6}.$$

This exactly matches the ratio claimed by the upper bound. So in this worst-case instance, LCM's performance is precisely 1/6 times that of OPT. This completes the proof of Lemma 2. □

LEMMA EC.1 *If both functions f and g satisfy diminishing returns on X, then function $f+g$ satisfies diminishing returns on X.*

Proof of Lemma EC.1. Function f and g satisfy *diminishing returns* on X, we have

$$f(z_{-ij}, \widehat{z}^i, \widehat{z}^j) - f(z_{-ij}, \widehat{z}^i, z^j) \leq f(z_{-ij}, z^i, \widehat{z}^j) - f(z_{-ij}, z^i, z^j). \tag{EC.13}$$

$$g(z_{-ij}, \widehat{z}^i, \widehat{z}^j) - g(z_{-ij}, \widehat{z}^i, z^j) \leq g(z_{-ij}, z^i, \widehat{z}^j) - g(z_{-ij}, z^i, z^j). \tag{EC.14}$$

The result of above (EC.13) + (EC.14) is as follows.

$$\begin{aligned}&[f(z_{-ij}, \widehat{z}^i, \widehat{z}^j) + g(z_{-ij}, \widehat{z}^i, \widehat{z}^j)] - [f(z_{-ij}, \widehat{z}^i, z^j) + g(z_{-ij}, \widehat{z}^i, z^j)] \\ \leq\ &[f(z_{-ij}, z^i, \widehat{z}^j) + g(z_{-ij}, z^i, \widehat{z}^j)] - [f(z_{-ij}, z^i, z^j) + g(z_{-ij}, z^i, z^j)],\end{aligned}$$

which is the result wanted. □

LEMMA EC.2 *Let $X \subset \Re_+^I$ be the set of initial capacity levels of a seru system. Then X is a lattice.*

Proof of Lemma EC.2. Suppose that x and y are two arbitrary elements of X, i.e., $x \in X$ and $y \in X$, and $x \neq y$. Obviously, $x \wedge y$ and $x \vee y$ in X. □

LEMMA EC.4 *For any order $o \in \mathcal{O}$, x_o is a Lipschitz function.*

Proof of Lemma EC.4. From Equation (3), it is sufficient to show that $\|x_{o+1} - y_{o+1}\| \leq M\|x_o - y_o\|$, where M is a positive constant number. Because the square root is a subadditive function, $\|x_o - y_o\| \leq \sum_{i \in \mathcal{I}} \|x_o^i - y_o^i\|$, it is sufficient to show that $\|x_{o+1}^i - y_{o+1}^i\| \leq M\|x_o - y_o\|$, or more precisely, $\|x_{o+1}^i - y_{o+1}^i\| \leq M\|x_o^i - y_o^i\|$, for $i \in \mathcal{I}$. Show that $\|x_{o+1}^i - y_{o+1}^i\| \leq M\|x_o^i - y_o^i\|$ is true for $r_i = 1, 2, \ldots, \lambda_o, \lambda_o + 1$. There are three cases.

Case 1. $r_i = 1$.

$$\|x_{o+1}^i - y_{o+1}^i\| = \|(x_o^i - \max\{a_o, d_o\tau_o\})^+ - (y_o^i - \max\{a_o, d_o\tau_o\})^+\|.$$

Case 1.1 $x_o^i \geq \max\{a_o, d_o\tau_o\}$ and $y_o^i \geq \max\{a_o, d_o\tau_o\}$.

$$\|x_{o+1}^i - y_{o+1}^i\| = \|x_o^i - \max\{a_o, d_o\tau_o\} - y_o^i + \max\{a_o, d_o\tau_o\}\| = \|x_o^i - y_o^i\|.$$

Case 1.2 $x_o^i < \max\{a_o, d_o\tau_o\}$ and $y_o^i \geq \max\{a_o, d_o\tau_o\}$.

$$\|x_{o+1}^i - y_{o+1}^i\| = \|0 - y_o^i + \max\{a_o, d_o\tau_o\}\| < \|x_o^i - y_o^i\|.$$

Case 1.3 $x_o^i \geq \max\{a_o, d_o\tau_o\}$ and $y_o^i < \max\{a_o, d_o\tau_o\}$.

$$\|x_{o+1}^i - y_{o+1}^i\| = \|x_o^i - \max\{a_o, d_o\tau_o\} - 0\| < \|x_o^i - y_o^i\|.$$

Case 1.4 $x_o^i < \max\{a_o, d_o\tau_o\}$ and $y_o^i < \max\{a_o, d_o\tau_o\}$.

$$\|x_{o+1}^i - y_{o+1}^i\| = \|0 - 0\| \leq \|x_o^i - y_o^i\|.$$

Case 2. $\underline{1 < r_i \leq \lambda_o}$.

$$\begin{aligned}\|x_{o+1}^i - y_{o+1}^i\| = \|&\mathbb{I}_{d_o\tau_o<\widetilde{m}}(x_o^i - a_o)^+ + \mathbb{I}_{\widetilde{m}\leq d_o\tau_o<m}\left(x_o^i - \max\{d_o\tau_o - \widetilde{m}, a_o\}\right)^+ \\ &- \mathbb{I}_{d_o\tau_o<\widetilde{m}'}(y_o^i - a_o)^+ - \mathbb{I}_{\widetilde{m}'\leq d_o\tau_o<m'}\left(y_o^i - \max\{d_o\tau_o - \widetilde{m}', a_o\}\right)^+\|,\end{aligned}$$

where $y_o^{[1]} + y_o^{[2]} + \cdots + y_o^{[r_i-1]} = \widetilde{m}'$, and $y_o^{[1]} + y_o^{[2]} + \cdots + y_o^{[r_i]} = m'$. Because $\widetilde{m} \geq \widetilde{m}'$ and $m \geq m'$, $\widetilde{m}$ needs to be compared with m'.

(1) If $m' \leq \widetilde{m}$, then

$$\begin{aligned}\|x_{o+1}^i - y_{o+1}^i\| = \|&\mathbb{I}_{d_o\tau_o<\widetilde{m}'}\left[(x_o^i - a_o)^+ - (y_o^i - a_o)^+\right] \\ &+ \mathbb{I}_{\widetilde{m}'\leq d_o\tau_o<m'}\left[(x_o^i - a_o)^+ - (y_o^i - \max\{d_o\tau_o - \widetilde{m}', a_o\})^+\right] \\ &+ \mathbb{I}_{m'\leq d_o\tau_o<\widetilde{m}}\left(x_o^i - a_o\right)^+ + \mathbb{I}_{\widetilde{m}\leq d_o\tau_o<m}(x_o^i - \max\{d_o\tau_o - \widetilde{m}, a_o\})^+\|.\end{aligned}$$

For intervals $d_o\tau_o \leq \widetilde{m}'$, $\widetilde{m}' < d_o\tau_o \leq m'$, $m' < d_o\tau_o \leq \widetilde{m}$, and $\widetilde{m} < d_o\tau_o \leq m$, positive constants C_1, C_2, C_3, and C_4 can be found for each interval, respectively, to make $\|x_{o+1}^i - y_{o+1}^i\| \leq C_k\|x_o^i - y_o^i\|$, $k = 1, ..., 4$.

(2) If $m' > \widetilde{m}$, then

$$\begin{aligned}\|x_{o+1}^i - y_{o+1}^i\| = \|&\mathbb{I}_{d_o\tau_o<\widetilde{m}'}\left[(x_o^i - a_o)^+ - (y_o^i - a_o)^+\right] \\ &+ \mathbb{I}_{\widetilde{m}'\leq d_o\tau_o<\widetilde{m}}\left[(x_o^i - a_o)^+ - (y_o^i - \max\{d_o\tau_o - \widetilde{m}', a_o\})^+\right] \\ &+ \mathbb{I}_{\widetilde{m}\leq d_o\tau_o<m'}[(x_o^i - \max\{d_o\tau_o - \widetilde{m}, a_o\})^+ - (y_o^i - \max\{d_o\tau_o - \widetilde{m}', a_o\})^+] \\ &+ \mathbb{I}_{m'\leq d_o\tau_o<m}(x_o^i - \max\{d_o\tau_o - \widetilde{m}, a_o\})^+\|.\end{aligned}$$

For intervals $d_o\tau_o \leq \widetilde{m}'$, $\widetilde{m}' < d_o\tau_o \leq \widetilde{m}$, $\widetilde{m} < d_o\tau_o \leq m'$, and $m' < d_o\tau_o \leq m$, positive constants C_5, C_6, C_7, and C_8 can be found for each interval, respectively, to make $\|x_{o+1}^i - y_{o+1}^i\| \leq C_k\|x_o^i - y_o^i\|$, $k = 5, ..., 8$.

Case 3. $\underline{r_i = \lambda_o + 1}$.

$$\|x_{o+1}^i - y_{o+1}^i\| = \|(x_o^i - a_o)^+ - (y_o^i - a_o)^+\| \leq \|x_o^i - y_o^i\|.$$

Therefore, always let $M = max\{C_k, 1\}$, so that $\|x_{o+1} - y_{o+1}\| \le M\|x_o - y_o\|$. $\square$

LEMMA EC.5 *For any order $o \in \mathcal{O}$ and any seru $i \in \mathcal{I}$, then*

$$\|q_o^i(x,\omega) - q_o^i(y,\omega)\| \le \|(x_o^i - x_{o+1}^i) - (y_o^i - y_{o+1}^i)\|.$$

Proof of Lemma EC.5. To clarify, Equation (3) is rewritten as follows.

$$x_{o+1}^i = \begin{cases} \mathbb{I}_{a_o \le d_o\tau_o \le x_o^i}(x_o^i - d_o\tau_o) + \mathbb{I}_{d_o\tau_o \le a_o \le x_o^i}(x_o^i - a_o) + \mathbb{I}_{\text{otherwise}}0, & r_i = 1; \\ \mathbb{I}_{(d_o\tau_o < \widetilde{m})\&(a_o \le x_o^i)}(x_o^i - a_o) + \mathbb{I}_{(\widetilde{m} \le d_o\tau_o < m)\&(a_o < d_o\tau_o - \widetilde{m})}(x_o^i - (d_o\tau_o - \widetilde{m})) \\ \quad + \mathbb{I}_{(\widetilde{m} \le d_o\tau_o < m)\&(d_o\tau_o - \widetilde{m} \le a_o \le x_o^i)}(x_o^i - a_o) + \mathbb{I}_{\text{otherwise}}0, & 1 < r_i \le \lambda_o; \\ \mathbb{I}_{a_o \le x_o^i}(x_o^i - a_o) + \mathbb{I}_{a_o > x_o^i}0, & r_i = \lambda_o + 1. \end{cases}$$

Then $x_o^i - x_{o+1}^i$ is as follows.

$$x_o^i - x_{o+1}^i = \begin{cases} \mathbb{I}_{a_o \le d_o\tau_o \le x_o^i} d_o\tau_o + \mathbb{I}_{d_o\tau_o \le a_o \le x_o^i} a_o + \mathbb{I}_{\text{otherwise}} x_o^i, & r_i = 1; \\ \mathbb{I}_{(d_o\tau_o < \widetilde{m})\&(a_o \le x_o^i)} a_o + \mathbb{I}_{(\widetilde{m} \le d_o\tau_o < m)\&(a_o < d_o\tau_o - \widetilde{m})}(d_o\tau_o - \widetilde{m}) \\ \quad + \mathbb{I}_{(\widetilde{m} \le d_o\tau_o < m)\&(d_o\tau_o - \widetilde{m} \le a_o \le x_o^i)} a_o + \mathbb{I}_{\text{otherwise}} x_o^i, & 1 < r_i \le \lambda_o; \\ \mathbb{I}_{a_o \le x_o^i} a_o + \mathbb{I}_{a_o > x_o^i} x_o^i, & r_i = \lambda_o + 1. \end{cases}$$

From Equation (4), $q_o^i(x,\omega) - q_o^i(y,\omega)$ is as follows.

$$q_o^i(x,\omega) - q_o^i(y,\omega) = \begin{cases} \mathbb{I}_{d_o\tau_o < x_o^i} d_o\tau_o + \mathbb{I}_{d_o\tau_o \ge x_o^i} x_o^i - \mathbb{I}_{d_o\tau_o < y_o^i} d_o\tau_o - \mathbb{I}_{d_o\tau_o \ge y_o^i} y_o^i, & r_i = 1; \\ \mathbb{I}_{d_o\tau_o < \widetilde{m}} 0 + \mathbb{I}_{\widetilde{m} \le d_o\tau_o < m}(d_o\tau_o - \widetilde{m}) + \mathbb{I}_{d_o\tau_o \ge m} x_o^i \\ \quad - \mathbb{I}_{d_o\tau_o < \widetilde{m}'} 0 - \mathbb{I}_{\widetilde{m}' \le d_o\tau_o < m'}(d_o\tau_o - \widetilde{m}') - \mathbb{I}_{d_o\tau_o \ge m'} y_o^i, & 1 < r_i \le \lambda_o; \\ 0, & r_i = \lambda_o + 1. \end{cases}$$

where $y_o^{[1]} + y_o^{[2]} + \cdots + y_o^{[r_i - 1]} = \widetilde{m}'$, and $y_o^{[1]} + y_o^{[2]} + \cdots + y_o^{[r_i]} = m'$. Without loss of generality, assume that $x_o^i \ge y_o^i$. From Lemma EC.9, $x_{o+1}^i \ge y_{o+1}^i$. There are three cases.

Case 1: $r_i = 1$.

$$\begin{aligned} \|q_o^i(x,\omega) - q_o^i(y,\omega)\| &= \|\mathbb{I}_{d_o\tau_o < x_o^i} d_o\tau_o + \mathbb{I}_{d_o\tau_o \ge x_o^i} x_o^i - \mathbb{I}_{d_o\tau_o < y_o^i} d_o\tau_o - \mathbb{I}_{d_o\tau_o \ge y_o^i} y_o^i\| \\ &= \|\mathbb{I}_{d_o\tau_o < y_o^i}(d_o\tau_o - d_o\tau_o) + \mathbb{I}_{y_o^i \le d_o\tau_o < x_o^i}(d_o\tau_o - y_o^i) + \mathbb{I}_{d_o\tau_o \ge x_o^i}(x_o^i - y_o^i)\|. \end{aligned}$$

$$\begin{aligned} \|(x_o^i - x_{o+1}^i) - (y_o^i - y_{o+1}^i)\| &= \|\mathbb{I}_{a_o \le d_o\tau_o \le x_o^i} d_o\tau_o + \mathbb{I}_{d_o\tau_o \le a_o \le x_o^i} a_o + \mathbb{I}_{\text{otherwise}} x_o^i \\ &\quad - \mathbb{I}_{a_o \le d_o\tau_o \le y_o^i} d_o\tau_o - \mathbb{I}_{d_o\tau_o \le a_o \le y_o^i} a_o - \mathbb{I}_{\text{otherwise}} y_o^i\| \\ &= \|\mathbb{I}_{a_o \le d_o\tau_o \le y_o^i}(d_o\tau_o - d_o\tau_o) + \mathbb{I}_{d_o\tau_o \le a_o \le y_o^i}(a_o - a_o) + \mathbb{I}_{y_o^i \le d_o\tau_o \le x_o^i}(d_o\tau_o - y_o^i) \\ &\quad + \mathbb{I}_{y_o^i \le a_o \le x_o^i}(a_o - y_o^i) + \mathbb{I}_{\text{otherwise}}(x_o^i - y_o^i)\|. \end{aligned}$$

For each interval, $\|q_o^i(x,\omega)-q_o^i(y,\omega)\| \le \|(x_o^i-x_{o+1}^i)-(y_o^i-y_{o+1}^i)\|$.

Case 2: $\underline{1<r_i\le\lambda_o}$.

Because $\widetilde{m}\ge\widetilde{m}'$ and $m\ge m'$, $\widetilde{m}$ needs to be compared with m'.

(1) If $m'\le\widetilde{m}$, then

$$\begin{aligned}\|q_o^i(x,\omega)-q_o^i(y,\omega)\| = \|&\mathbb{I}_{d_o\tau_o<\widetilde{m}'}0+\mathbb{I}_{\widetilde{m}'\le d_o\tau_o<m'}\left(\widetilde{m}'-d_o\tau_o\right)+\mathbb{I}_{m'\le d_o\tau_o<\widetilde{m}}(-y_o^i)\\&+\mathbb{I}_{\widetilde{m}\le d_o\tau_o<m}(d_o\tau_o-\widetilde{m}-y_o^i)+\mathbb{I}_{d_o\tau_o\ge m}\left(x_o^i-y_o^i\right)\|.\end{aligned}$$

$$\begin{aligned}\|(x_o^i-x_{o+1}^i)-(y_o^i-y_{o+1}^i)\| = \|&\mathbb{I}_{(d_o\tau_o<\widetilde{m})\&(a_o\le x_o^i)}a_o+\mathbb{I}_{(\widetilde{m}\le d_o\tau_o<m)\&(a_o<d_o\tau_o-\widetilde{m})}(d_o\tau_o-\widetilde{m})\\&+\mathbb{I}_{(\widetilde{m}\le d_o\tau_o<m)\&(d_o\tau_o-\widetilde{m}\le a_o\le x_o^i)}a_o+\mathbb{I}_{\text{otherwise}}x_o^i\\&-\mathbb{I}_{(d_o\tau_o<\widetilde{m}')\&(a_o\le y_o^i)}a_o-\mathbb{I}_{(\widetilde{m}'\le d_o\tau_o<m')\&(a_o<d_o\tau_o-\widetilde{m}')}(d_o\tau_o-\widetilde{m}')\\&-\mathbb{I}_{(\widetilde{m}'\le d_o\tau_o<m')\&(d_o\tau_o-\widetilde{m}'\le a_o\le y_o^i)}a_o-\mathbb{I}_{\text{otherwise}}y_o^i\|\\=\|&\mathbb{I}_{(d_o\tau_o<\widetilde{m}')\&(a_o\le y_o^i)}(a_o-a_o)+\mathbb{I}_{(d_o\tau_o<\widetilde{m}')\&(y_o^i<a_o\le x_o^i)}(a_o-y_o^i)\\&+\mathbb{I}_{(\widetilde{m}'\le d_o\tau_o<m')\&(a_o<d_o\tau_o-\widetilde{m}')}[a_o-(d_o\tau_o-\widetilde{m}')]\\&+\mathbb{I}_{(\widetilde{m}'\le d_o\tau_o<m')\&(d_o\tau_o-\widetilde{m}'<a_o<y_o^i)}(a_o-a_o)\\&+\mathbb{I}_{(m'<d_o\tau_o<\widetilde{m})\&(a_o\le x_o^i)}(a_o-y_o^i)+\mathbb{I}_{(\widetilde{m}\le d_o\tau_o<m)\&(a_o\le d_o\tau_o-\widetilde{m})}(d_o\tau_o-\widetilde{m}-y_o^i)\\&+\mathbb{I}_{(\widetilde{m}\le d_o\tau_o<m)\&(d_o\tau_o-\widetilde{m}\le a_o\le x_o^i)}(a_o-y_o^i)+\mathbb{I}_{\text{otherwise}}(x_o^i-y_o^i)\|.\end{aligned}$$

For each interval of $d_o\tau_o\le\widetilde{m}'$, $\widetilde{m}'<d_o\tau_o\le m'$, $m'<d_o\tau_o\le\widetilde{m}$, $\widetilde{m}<d_o\tau_o\le m$, and $m<d_o\tau_o$, $\|q_o^i(x,\omega)-q_o^i(y,\omega)\| \le \|(x_o^i-x_{o+1}^i)-(y_o^i-y_{o+1}^i)\|$.

(2) If $m'>\widetilde{m}$, then

$$\begin{aligned}\|q_o^i(x,\omega)-q_o^i(y,\omega)\| = \|&\mathbb{I}_{d_o\tau_o<\widetilde{m}'}0+\mathbb{I}_{\widetilde{m}'\le d_o\tau_o<\widetilde{m}}\left(\widetilde{m}'-d_o\tau_o\right)+\mathbb{I}_{\widetilde{m}\le d_o\tau_o<m'}(d_o\tau_o-\widetilde{m}-d_o\tau_o+\widetilde{m}')\\&+\mathbb{I}_{m'\le d_o\tau_o<m}(d_o\tau_o-\widetilde{m}-y_o^i)+\mathbb{I}_{d_o\tau_o\ge m}\left(x_o^i-y_o^i\right)\|.\end{aligned}$$

$$\begin{aligned}\|(x_o^i-x_{o+1}^i)-(y_o^i-y_{o+1}^i)\| = \|&\mathbb{I}_{(d_o\tau_o<\widetilde{m})\&(a_o\le x_o^i)}a_o+\mathbb{I}_{(\widetilde{m}\le d_o\tau_o<m)\&(a_o<d_o\tau_o-\widetilde{m})}(d_o\tau_o-\widetilde{m})\\&+\mathbb{I}_{(\widetilde{m}\le d_o\tau_o<m)\&(d_o\tau_o-\widetilde{m}\le a_o\le x_o^i)}a_o+\mathbb{I}_{\text{otherwise}}x_o^i\\&-\mathbb{I}_{(d_o\tau_o<\widetilde{m}')\&(a_o\le y_o^i)}a_o-\mathbb{I}_{(\widetilde{m}'\le d_o\tau_o<m')\&(a_o<d_o\tau_o-\widetilde{m}')}(d_o\tau_o-\widetilde{m}')\\&-\mathbb{I}_{(\widetilde{m}'\le d_o\tau_o<m')\&(d_o\tau_o-\widetilde{m}'\le a_o\le y_o^i)}a_o-\mathbb{I}_{\text{otherwise}}y_o^i\|\\=\|&\mathbb{I}_{(d_o\tau_o<\widetilde{m}')\&(a_o\le y_o^i)}(a_o-a_o)+\mathbb{I}_{(d_o\tau_o<\widetilde{m}')\&(y_o^i<a_o\le x_o^i)}(a_o-y_o^i)\\&+\mathbb{I}_{(\widetilde{m}'\le d_o\tau_o<\widetilde{m})\&(a_o<d-\widetilde{m}')}[a_o-(d_o\tau_o-\widetilde{m}')]\\&+\mathbb{I}_{(\widetilde{m}'\le d_o\tau_o<\widetilde{m})\&(d_o\tau_o-\widetilde{m}'<a_o<y_o^i)}(a_o-a_o)\\&+\mathbb{I}_{(\widetilde{m}\le d_o\tau_o<m')\&(a_o<d_o\tau_o-\widetilde{m})}(d_o\tau_o-\widetilde{m}-d_o\tau_o+\widetilde{m}')\\&+\mathbb{I}_{(\widetilde{m}\le d_o\tau_o<m')\&(d_o\tau_o-\widetilde{m}'\le a_o<y_o^i)}(a_o-a_o)\\&+\mathbb{I}_{(\widetilde{m}\le d_o\tau_o<m')\&(d_o\tau_o-\widetilde{m}\le y_o^i\le a_o<x_o^i)}(a_o-y_o^i)+\mathbb{I}_{\text{otherwise}}(x_o^i-y_o^i)\|.\end{aligned}$$

For each interval of $d_o\tau_o \le \widetilde{m}'$, $\widetilde{m}' < d_o\tau_o \le \widetilde{m}$, $\widetilde{m} < d_o\tau_o \le m'$, $m' < d_o\tau_o \le m$, and $m < d_o\tau_o$, $\|q_o^i(x,\omega) - q_o^i(y,\omega)\| \le \|(x_o^i - x_{o+1}^i) - (y_o^i - y_{o+1}^i)\|$.

Case 3: $\underline{r_i = \lambda_o + 1}$.

$$\|q_o^i(x,\omega) - q_o^i(y,\omega)\| = 0.$$

$$\begin{aligned}\|(x_o^i - x_{o+1}^i) - (y_o^i - y_{o+1}^i)\| &= \|\mathbb{I}_{a_o \le x_o^i} a_o + \mathbb{I}_{a_o > x_o^i} x_o^i - \mathbb{I}_{a_o \le y_o^i} a_o - \mathbb{I}_{a_o > y_o^i} y_o^i\| \\ &= \|\mathbb{I}_{a_o \le y_o^i}(a_o - a_o) + \mathbb{I}_{y_o^i < a_o \le x_o^i}(a_o - y_o^i) - \mathbb{I}_{a_o > x_o^i}(x_o^i - y_o^i)\| \ge 0.\end{aligned}$$

From Cases 1, 2, and 3, it is concluded that

$$\|q_o^i(x,\omega) - q_o^i(y,\omega)\| \le \|(x_o^i - x_{o+1}^i) - (y_o^i - y_{o+1}^i)\|. \quad \square$$

LEMMA EC.6 *For order $o \in \mathcal{O}$, if its demand d_o has an absolutely continuous cumulative distribution function, and d_o and O are bounded, then $R(x,\omega)$ satisfies the three conditions in Lemma EC.3.*

Proof of Lemma EC.6. 1. Differentiability. From Equation (1), it is sufficient to show that $q_o^i(x,\omega)$ is differentiable. Because d_o has an absolutely continuous cumulative distribution function, d_o is differentiable for every $o \in \mathcal{O}$. So from Equation (4), $q_o^i(x,\omega)$ is differentiable for all $o \in \mathcal{O}$ and $i \in \mathcal{I}$.

2. Integrability. It is sufficient to show that $R(x,\omega)$ is an admissible function because every admissible function is integrable (Edwards 1973). An admissible function is one that (a) is bounded; (b) has bounded support; (c) is continuous except on a negligible set. $R(x,\omega)$ satisfies these three admissible conditions as follows. (a) $R(x,\omega)$ is clearly bounded since d_o and O are bounded; (b) the bounded support of $R(x,\omega)$ is $[1,O]$; (c) since $R(x,\omega)$ is differentiable, it is continuous.

3. Lipschitz condition with modulus K_X. It is sufficient to show that there exists a constant real number K_R, such that $\| R(x,\omega) - R(y,\omega) \| \le K_R \| x - y \|$. Apply Lemma (EC.5) in the following process.

$$\begin{aligned}\|R(x,\omega) - R(y,\omega)\| &= \left\| \sum_{i\in\mathcal{I}} \sum_{o\in\mathcal{O}} b_o/\tau_o [q_o^i(x,\omega) - q_o^i(y,\omega)] - \sum_{i\in\mathcal{I}} e_i [x^i - y^i] \right\| \\ &\le \left\| \sum_{i\in\mathcal{I}} \sum_{o\in\mathcal{O}} b_o/\tau_o [(x_o^i - y_o^i) - (x_{o+1}^i - y_{o+1}^i)] - \sum_{i\in\mathcal{I}} e_i [x_1^i - y_1^i] \right\| \\ &\le \left\| \sum_{i\in\mathcal{I}} \sum_{o\in\mathcal{O}} b_o/\tau_o [x_o^i - y_o^i] \right\| + \left\| \sum_{i\in\mathcal{I}} \sum_{o\in\mathcal{O}} b_o/\tau_o [x_{o+1}^i - y_{o+1}^i] \right\| + \left\| \sum_{i\in\mathcal{I}} e_i [x_1^i - y_1^i] \right\| \\ &\le \sum_{i\in\mathcal{I}} \sum_{o\in\mathcal{O}} b_o/\tau_o (\| x_o^i - y_o^i \| + \| x_{o+1}^i - y_{o+1}^i \|) + \sum_{i\in\mathcal{I}} e_i \| x_1^i - y_1^i \| \\ &\le \sum_{o\in\mathcal{O}} b_o/\tau_o (\|x_o - y_o\| + \|x_{o+1} - y_{o+1}\|) + e_i \|x_1 - y_1\|.\end{aligned}$$

By Lemma EC.4, two positive constant numbers C_1 and C_2 can be found, such that $\|x_o - y_o\| \leq C_1\|x_1 - y_1\|$ and $\|x_{o+1} - y_{o+1}\| \leq C_2\|x_1 - y_1\|$. Therefore, $\|R(x,\omega) - R(y,\omega)\| \leq K_R\|x - y\|$, where $K_R = \sum_{o\in\mathcal{O}}[b_o/\tau_o]C_1 + \sum_{o\in\mathcal{O}}[b_o/\tau_o]C_2 + e_i$. □

LEMMA EC.7 *If f and g are Lipschitz with modulus F and G, and a and b are scalars, then $af + bg$ is Lipschitz with modulus $aF + bG$.*

Proof of Lemma EC.7.

$$\begin{aligned}||(af(x) + bg(x)) - (af(y) + bg(y))|| &\leq a\,||f(x) - f(y)|| + b\,||g(x) - g(y)|| \\ &\leq (aF + bG)\,||x - y||\,. \quad \square\end{aligned}$$

Proofs of theorems and lemmas are facilitated using Lemma EC.8 and Lemma EC.9.

Lemma EC.8, which is a refinement of Lemma 1, demonstrates the strict diminishing returns property of the capacity consumption function $q_o^i(x,\omega)$. Two feasible initial capacities, $\widehat{x}^j$ and x^j, are defined for *seru* j, where $\widehat{x}^j \geq x^j$ and $i \neq j$. The difference between the two capacities is denoted by $\beta_j = \widehat{x}^j - x^j$.

LEMMA EC.8. *For an order o assembled on an arbitrary sample path ω, capacity consumption $q_o^i(x,\omega)$ of seru i satisfies strictly diminishing returns with respect to the initial capacity of an arbitrary seru j $(i \neq j)$, x^j on X, if ranks $r_j < r_i \leq \lambda_o$ and $m \leq d_o\tau_o \leq m + \beta_j$. That is, $q_o^i(x_{-ij}, \widehat{x}^i, \widehat{x}^j, \omega) - q_o^i(x_{-ij}, x^i, \widehat{x}^j, \omega) < q_o^i(x_{-ij}, \widehat{x}^i, x^j, \omega) - q_o^i(x_{-ij}, x^i, x^j, \omega)$.*

Proof of Lemma EC.8. The notation and definitions used in this proof are the same as those in Lemma 1. The difference between $\widehat{q}_o^i(0,0)$ and $\widehat{q}_o^i(0,\beta_j)$ under conditions $r_j < r_i \leq \lambda_o$ and $m \leq d_o\tau_o \leq m + \beta_j$ is examined. This difference, defined as $\Delta\widehat{q}_o^i(0,\Delta_\delta) = \widehat{q}_o^i(0,\beta_j) - \widehat{q}_o^i(0,0)$, is calculated by comparing β_j with x_o^i in the following three cases.

Case 1: $\beta_j < x_o^i$.

$$\begin{aligned}\Delta\widehat{q}_o^i(0,\Delta_\delta) &= [\mathbb{I}_{\widetilde{m}+\beta_j \leq d_o\tau_o < m+\beta_j}\,(d_o\tau_o - \widetilde{m} - \beta_j) + \mathbb{I}_{m+\beta_j \leq d_o\tau_o} x_o^i] \\ &\quad - [\mathbb{I}_{\widetilde{m} \leq d_o\tau_o < m}\,(d_o\tau_o - \widetilde{m}) + \mathbb{I}_{m \leq d_o\tau_o} x_o^i] \\ &= d_o\tau_o - \widetilde{m} - \beta_j - x_o^i. \end{aligned} \tag{EC.15}$$

Case 2: $\beta_j = x_o^i$.

$$\Delta\widehat{q}_o^i(0,\Delta_\delta) = d_o\tau_o - \widetilde{m} - \beta_j - x_o^i. \tag{EC.16}$$

Case 3: $\beta_j > x_o^i$.

$$\Delta\widehat{q}_o^i(0,\Delta_\delta) = \mathbb{I}_{m \leq d_o\tau_o < \widetilde{m}+\beta_j}(-x_o^i) + \mathbb{I}_{\widetilde{m}+\beta_j \leq d_o\tau_o < m+\beta_j}(d_o\tau_o - \widetilde{m} - \beta_j - x_o^i). \tag{EC.17}$$

Next a parametric analysis on γ is performed to investigate the monotony of $\Delta\widehat{q}_o^i(\gamma,\Delta_\delta)$, where $\Delta\widehat{q}_o^i(\gamma,\Delta_\delta) = \widehat{q}_o^i(\gamma,\beta_j) - \widehat{q}_o^i(\gamma,0)$. Since x_o^i can be expressed as a monotonically increasing function

of γ, Equations (EC.15), (EC.16), and (EC.17) are clearly strictly monotonically decreasing on γ. This means that $\Delta\widehat{q}_o^i(\beta_i,\Delta_\delta) < \Delta\widehat{q}_o^i(0,\Delta_\delta)$ or $\widehat{q}_o^i(\beta_i,\beta_j) - \widehat{q}_o^i(\beta_i,0) < \widehat{q}_o^i(0,\beta_j) - \widehat{q}_o^i(0,0)$ or

$$q_o^i(x_{-ij},\widehat{x}^i,\widehat{x}^j,\omega) - q_o^i(x_{-ij},\widehat{x}^i,x^j,\omega) < q_o^i(x_{-ij},x^i,\widehat{x}^j,\omega) - q_o^i(x_{-ij},x^i,x^j,\omega). \tag{EC.18}$$

Equation (EC.18) can also be written as follows.

$$q_o^i(x_{-ij},\widehat{x}^i,\widehat{x}^j,\omega) - q_o^i(x_{-ij},x^i,\widehat{x}^j,\omega) < q_o^i(x_{-ij},\widehat{x}^i,x^j,\omega) - q_o^i(x_{-ij},x^i,x^j,\omega). \tag{EC.19}$$

Because ω is an arbitrary sample path, i and j are arbitrary *serus*, and $i \neq j$, Equations (EC.18) and (EC.19) show that if $r_j < r_i \leq \lambda_o$ and $m \leq d_o\tau_o \leq m + \beta_j$, then $q_o^i(x,\omega)$ satisfies strictly decreasing differences with respect to $x^1,\ldots,x^I$. $\square$

LEMMA EC.9. *Suppose that $x=(x^1,x^2,\ldots,x^I)$ and $y=(y^1,y^2,\ldots,y^I)$ are two initial capacity levels of a seru system, $x\in X$, $y\in X$. For any sample path ω and order $o\in\mathcal{O}$, if $x\geq y$, then $x_o \geq y_o$. $x\geq y$ implies $x^j \geq y^j$ for $j=1,2,\ldots,I$.*

Proof of Lemma EC.9. The proof is by induction. For a sample path ω, $x_1 \geq y_1$, by $x\geq y$. Assume that $x_o \geq y_o$, so $x_{o+1} \geq y_{o+1}$ must be proved. That is, $x_{o+1} - y_{o+1} \geq 0$. Let i be an arbitrary *seru*. Checking the relation between x_{o+1}^i and y_{o+1}^i, there are three cases.

Case 1. $r_i = 1$.

$$\begin{aligned} x_{o+1}^i - y_{o+1}^i &= \mathbb{I}_{d_o\tau_o\leq a_o}\left(x_o^i - a_o\right)^+ + \mathbb{I}_{d_o\tau_o>a_o}\left(x_o^i - d_o\tau_o\right)^+ - \mathbb{I}_{d_o\tau_o\leq a_o}\left(y_o^i - a_o\right)^+ - \mathbb{I}_{d_o\tau_o>a_o}\left(y_o^i - d_o\tau_o\right)^+ \\ &= \mathbb{I}_{d_o\tau_o\leq a_o}[\left(x_o^i - a_o\right)^+ - (y_o^i - a_o)^+] + \mathbb{I}_{d_o\tau_o>a_o}[(x_o^i - d_o\tau_o)^+ - (y_o^i - d_o\tau_o)^+]. \end{aligned}$$

By assumption, $x_o \geq y_o$, so that $x_{o+1}^i - y_{o+1}^i \geq 0$.

Case 2. $1 < r_i \leq \lambda_o$.

$$\begin{aligned} x_{o+1}^i - y_{o+1}^i = {} & \mathbb{I}_{d_o\tau_o<\widetilde{m}}(x_o^i - a_o)^+ + \mathbb{I}_{\widetilde{m}\leq d_o\tau_o<m}\left(x_o^i - \max\{d_o\tau_o - \widetilde{m}, a_o\}\right)^+ \\ & - \mathbb{I}_{d_o\tau_o<\widetilde{m}'}(y_o^i - a_o)^+ - \mathbb{I}_{\widetilde{m}'\leq d_o\tau_o<m'}\left(y_o^i - \max\{d_o\tau_o - \widetilde{m}', a_o\}\right)^+, \end{aligned}$$

where $y_o^{[1]} + y_o^{[2]} + \cdots + y_o^{[r_i-1]} = \widetilde{m}'$, and $y_o^{[1]} + y_o^{[2]} + \cdots + y_o^{[r_i]} = m'$. Because $\widetilde{m} \geq \widetilde{m}'$ and $m \geq m'$, $\widetilde{m}$ needs to be compared with m'.

(1) If $m' \leq \widetilde{m}$, then

$$\begin{aligned} x_{o+1}^i - y_{o+1}^i = {} & \mathbb{I}_{d_o\tau_o<\widetilde{m}'}\left[(x_o^i - a_o)^+ - (y_o^i - a_o)^+\right] \\ & + \mathbb{I}_{\widetilde{m}'\leq d_o\tau_o<m'}\left[(x_o^i - a_o)^+ - (y_o^i - \max\{d_o\tau_o - \widetilde{m}', a_o\})^+\right] \\ & + \mathbb{I}_{m'\leq d_o\tau_o<\widetilde{m}}\left(x_o^i - a_o\right)^+ + \mathbb{I}_{\widetilde{m}\leq d_o\tau_o<m}(x_o^i - \max\{d_o\tau_o - \widetilde{m}, a_o\})^+. \end{aligned}$$

By assumption, $x_o \geq y_o$, so that $x_{o+1}^i - y_{o+1}^i \geq 0$.

(2) If $m' > \widetilde{m}$, then

$$\begin{aligned} x^i_{o+1} - y^i_{o+1} = & \mathbb{I}_{d_o\tau_o < \widetilde{m}'} \left[(x^i_o - a_o)^+ - (y^i_o - a_o)^+\right] \\ & + \mathbb{I}_{\widetilde{m}' \le d_o\tau_o < \widetilde{m}} \left[(x^i_o - a_o)^+ - (y^i_o - \max\{d_o\tau_o - \widetilde{m}', a_o\})^+\right] \\ & + \mathbb{I}_{\widetilde{m} \le d_o\tau_o < m'} [(x^i_o - \max\{d_o\tau_o - \widetilde{m}, a_o\})^+ - (y^i_o - \max\{d_o\tau_o - \widetilde{m}', a_o\})^+] \\ & + \mathbb{I}_{m' \le d_o\tau_o < m} (x^i_o - \max\{d_o\tau_o - \widetilde{m}, a_o\})^+. \end{aligned}$$

Because $d_o\tau_o - \widetilde{m} < d_o\tau_o - \widetilde{m}'$ and $x_o \ge y_o$, then $x^i_{o+1} - y^i_{o+1} \ge 0$.

From (1) and (2), it is concluded that for $1 < r_i \le \lambda_o$,

$$x^i_{o+1} - y^i_{o+1} \ge 0.$$

Case 3. $\underline{r_i = \lambda_o + 1}$.

$$x^i_{o+1} - y^i_{o+1} = (x^i_o - a_o)^+ - (y^i_o - a_o)^+ \ge 0.$$

From Cases 1, 2, and 3, it is concluded that $x^i_{o+1} - y^i_{o+1} \ge 0$. Because i is an arbitrary *seru*, this conclusion is true for any *seru* $i \in \mathcal{I}$. So $x_{o+1} - y_{o+1} \ge 0$. □

The proof of Lemma EC.3 can be found in Glasserman (1994).

EC.5. Remarks

REMARK 3 *Consider a context of maximum randomness. A customer order needs a volume of w product, each product requires N operations. Both a TPS assembly line and a 2-seru system are balanced, consisting of N workers, each worker has the same processing speed $1/t$ for each operation. In the line, each worker is responsible for one operation, whereas in the 2-seru system, each worker manages two operations within a seru. Under these identical conditions, the 2-seru system outperforms the line by achieving a shorter cycle time by t units.*

Proof of Remark 3. In the context of maximum randomness, processing times are exponentially distributed with mean t, and all system states are equally likely. For the TPS assembly line with N workers each responsible for one operation, the expected number of other jobs ahead at each station is $\frac{w-1}{N}$. Therefore, the expected time at each station is $t + \left(\frac{w-1}{N} \times t\right)$. Multiplying by the total number of stations N, the cycle time of the line is:

$$\mathrm{CT}_{\mathrm{line}} = N\left(t + \frac{w-1}{N}t\right) = Nt + (w-1)t = t(N + w - 1).$$

For the 2-*seru* system, workers are grouped into two *serus* with each worker managing two operations, resulting in a processing time of $2t$ per station. Since jobs are evenly distributed, the expected number of other jobs ahead at each *seru* station is $\frac{(w/2)-1}{N/2} = \frac{w-2}{N}$. Thus, the expected time at each *seru* station is $2t + \left(\frac{w-2}{N} \times 2t\right)$. Multiplying by the number of stations per *seru* $N/2$, the cycle time of the 2-*seru* system is:

$$\mathrm{CT}_{seru} = \frac{N}{2}\left(2t + \frac{w-2}{N} \times 2t\right) = Nt + (w-2)t = t(N + w - 2).$$

Subtracting the two cycle times yields the gap:

$$\mathrm{CT}_{\mathrm{line}} - \mathrm{CT}_{seru} = t(N+w-1) - t(N+w-2) = t,$$

which shows that under identical conditions, the 2-*seru* system achieves a shorter cycle time by t units compared to the TPS assembly line.

References

Bertsekas D, Tsistiklis J (1996) *Neuro-Dynamic Programming* (Belmont, MA: Athena Scientific).

Bottou L, Curtis FE, Nocedal J (2018) Optimization methods for large-scale machine learning. *SIAM Review* 60(2):223–311.

Edwards C (1973) *Advanced Calculus of Several Variables* (New York: Dover).

Fang C, Lin Z, Zhang T (2019) Sharp analysis for nonconvex sgd escaping from saddle points. *Conference on Learning Theory*, 1192–1234, Proceedings of Machine Learning Research.

Feige U (1998) A threshold of $\ln n$ for approximating set cover. *Journal of the ACM* 45(4):634–652.

Fotopoulos GB, Popovich P, Papadopoulos NH (2024) Review non-convex optimization method for machine learning arXiv:2410.02017.

Garey M, Johnson D (1979) *Computers and Intractability: A Guide to the Theory of NP-Completeness* (New York: W.H. Freeman).

Glasserman P (1994) Perturbation analysis of production networks. Yao DD, ed., *Stochastic Modeling and Analysis of Manufacturing Systems*, 233–280 (New York: Springer).

Golrezaei N, Nazerzadeh H, Rusmevichientong P (2014) Real-time optimization of personalized assortments. *Management Science* 60(6):1532–1551.

Kapralov M, Post I, Vondrák J (2013) Online submodular welfare maximization: Greedy is optimal. *Proceedings of the Twenty-fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, 1216–1225 (Society for Industrial and Applied Mathematics).

Katende R, Kasumba H (2024) Efficient saddle point evasion and local minima escape in high-dimensional non-convex optimization arXiv:2409.12604.

Lay S (1982) *Convex Sets and Their Applications* (New York: John Wiley & Sons).

Li H, Lin Z (2023) Restarted nonconvex accelerated gradient descent: No more polylogarithmic factor in the $\mathcal{O}(\epsilon^{-7/4})$ complexity. *Journal of Machine Learning Research* 24(157):1–37.

Ma W, Simchi-Levi D (2020) Algorithms for online matching, assortment, and pricing with tight weight-dependent competitive ratios. *Operations Research* 68(6):1787–1803.

Mahajan S, Van Ryzin G (2001) Stocking retail assortments under dynamic consumer substitution. *Operations Research* 49(3):334–351.

Mehta A (2013) Online matching and ad allocation. *Foundations and Trends in Theoretical Computer Science* 8(4):265–368.

Newton D, Yousefian F, Pasupathy R (2018) Stochastic gradient descent: Recent trends. Gel E, Ntaimo L, eds., *Recent Advances in Optimization and Modeling of Contemporary Problems*, chapter 9, 193–220.

Olofsson P, Andersson M (2012) *Probability, Statistics, and Stochastic Processes* (New Jersey: John Wiley & Sons).