5-2021

Flexible resource allocation for service and production systems

Peng WANG
Singapore Management University

FLEXIBLE RESOURCE ALLOCATION FOR SERVICE AND PRODUCTION SYSTEMS

PENG WANG

SINGAPORE MANAGEMENT UNIVERSITY

2021

Flexible Resource Allocation for Service and Production Systems

Peng Wang

A thesis submitted to Lee Kong Chian School of Business in partial fulfillment of the requirements for the Degree of Doctor of Philosophy in Business

Singapore Management University

2021

I hereby declare that this dissertation is my original work
and it has been written by me in its entirety.
I have duly acknowledged all the sources of information
which have been used in this dissertation.

This dissertation has also not been submitted for any degree
in any university previously.

Wang Peng

Peng Wang
31 May 2021

Abstract

Resource flexibility hedges against uncertainty in service and production systems. However, flexibility also brings complexity and difficulty in allocating resources. The thesis mainly studies managing flexible resources in two scenarios. The first scenario is a type of coordination of workers in a production or assembly line – bucket brigade. Specifically, the study shows how to manage a stochastic bucket brigade with discrete work stations. The second scenario is a service system with flexible service resources. The study proposes a distributive decision rule for the allocation of the resources under both supply and demand uncertainty.

Chapter 2 studies a J-station, I-worker bucket brigade with *preemptible* work content. The time duration for each worker to serve a job at a station is exponentially distributed with a rate that depends on the station's work content and the worker's work speed. We analytically derive the throughput and the coefficient of variation (CV) of the inter-completion time. We study the system under three cases. (i) If the work speeds depend only on the workers, the throughput gap between the stochastic and the deterministic systems can be up to 47% when the number of stations is small. (ii) If the work speeds depend on the workers and the stations such that the workers may not dominate each other at every station, the asymptotic throughput can be expressed as a function of two factors. (iii) If the work speeds depend on the workers, the stations, and the jobs, there is a trade-off between the intensification of the learning experience and the diversification of the skills.

Chapter 3 further studies a J-station, I-worker bucket brigade with *non-preemptible* work content. If the work content is non-preemptible, the work on each station can not be preempted and has to be processed by the same worker. We properly denote the waiting workers, re-analyze the state of

the system, and the transition probability matrix of the reset vectors. Finally, we derive the average throughput. In the numerical experiments, we first verify the theoretical results by simulations. Then we compare the throughput difference between a non-preemptible line and a preemptible line. If the workers are sequenced slowest-to-fastest, the preemptible line dominates the non-preemptible line. However, if the workers are sequenced fastest-to-slowest, the non-preemptible line can possibly dominate the preemptible line. As such, the management needs to consider about the actual setting to enhance the performance.

Chapter 4 studies a resource allocation problem, where the planner needs to decide simultaneously on both the supply and the allocation policy to fulfill the uncertain demand over a multi-period horizon. We introduce a *distributive decision rule*, which decides on the proportion of jobs awaiting dispatch to each of the possible resource supply pools. Our model has a convex reformulation that can be solved efficiently. Through simulations, we illustrate that the optimal solution evolves with changes in service distribution, initial conditions, temporal fluctuations in demand and resource availability. At last, we benchmark our model against the static rule and a fluid model. In doing so, we justify the adaptivity of the proposed distributive decision rule and show the robustness of our model to different settings.

Contents

Acknowledgements

I would like to express my deep and sincere gratitude to my research supervisor, Professor Yun Fong Lim, for providing me invaluable guidance and support throughout the years. His dedication and keen interest to help his students had been solely and mainly responsible for completing my thesis. His timely advice, meticulous scrutiny, and scholarly advice have inspired me to a very great extent to conduct the research and accomplish this task. I especially thank his unyielding determination to push, drag, and carry me forward whenever I ran off the rails.

I own a deep sense of gratitude to Professor Gar Goei Loke for factually co-advising me on Chapter 4 of the thesis. His inspirations, suggestions, enthusiasm and dynamism have widened my research scope and enriched the content of the thesis. I would also like to thank Professor Melvyn Sim, Professor Zhichao Zheng, and Professor Yini Gao, for their valuable suggestions and questions which incentivized me to improve the thesis from various perspectives. My thanks also goes to the rest of the faculty in our department for their guidance and help on my study.

I would like to thank my peers, He Yan, Luo Qian, and Nicholas Yeo, for the fun time we shared together, and the continuing support and help we gave to each other. My Ph.D. life would not have been complete without their company.

I am extremely grateful to my parents for their love, caring and sacrifices for educating and preparing me for my future. I deeply thank them for the freedom they allowed me to pursue my own interests in my life.

"I have never looked upon ease and happiness as ends in themselves – this critical basis I call the ideal of a pigsty. The ideals that have lighted my way, and time after time have given me new courage to face life cheerfully, have been Kindness, Beauty, and Truth."

—Einstein, Albert

To my late grandma

Chapter 1

Introduction

Resource flexibility hedges against uncertainty in service and production systems. However, flexibility also brings complexity and difficulty in allocating resources. The thesis mainly studies managing flexible resources in two scenarios. The first scenario is a type of coordination of workers in a production or assembly line – bucket brigade. Specifically, the thesis shows how to manage a stochastic bucket brigade with discrete work stations. The second scenario is a service system with flexible service resources. The study proposes a distributive decision rule for the allocation of the resources under both supply and demand uncertainty.

Chapter 2 studies a J-station, I-worker bucket brigade with *preemptible* work content. The time duration for each worker to serve a job at a station is exponentially distributed with a rate that depends on the station's expected work content and the worker's work speed. Our goal is to maximize the system's productivity or to minimize its inter-completion time variability. We analytically derive the throughput and the coefficient of variation (CV) of the inter-completion time. We study the system under three cases. (i) If the work speeds depend only on the workers, the throughput gap between the stochastic and the deterministic systems can be up to 47% when the number of stations is small. Either maximizing the throughput or minimizing the CV of the inter-completion time, the slowest-to-fastest worker sequence always outperforms the reverse sequence for the stochastic bucket brigade. To maximize the throughput, more work content should be assigned to the stations near the faster workers. In contrast, to minimize the CV of the inter-completion time, more work content should be allocated to the stations near the slower workers. (ii) If the work speeds depend on the workers and the stations such that the workers may not dominate each other at every station, the asymptotic throughput can be expressed as a function of the average work speeds and the asymptotic expected blocked times of the workers, and can be interpreted as the sum of the effective production rates of all the workers. (iii) If the work speeds depend on the workers, the stations, and the jobs, there is a trade-off between the intensification of the learning experience and the diversification of the skills. We find that if the workers have similar initial work speeds, then it is more productive to intensify their learning over a narrower work zone. Otherwise, it is crucial to mitigate the negative effect of the bottleneck (slowest) worker by cross-training the workers over a wider work zone.

Chapter 3 further studies a J-station, I-worker bucket brigade with *non-preemptible* work content. If the work content is non-preemptible, the work on each station can not be preempted and has to be processed by the same worker. When one worker walks back to his predecessor, he can not take over the job immediately, but has to wait for his predecessor to finish her job at the current station, and then he continues to process the job from the next station. the status of a worker can be not only working or being blocked, but can also be waiting for his predecessor to finish his job at a station. As such, it complicates the analysis in a few perspectives. (1) The waiting workers need to be well defined. (2) The inter-completion times and the relation between the reset vectors and the states of the system have to be re-analyzed. (3) The transitions of the states have to be analyzed case by case. To address these issues, we properly denote the waiting workers, re-analyze the state of the system, and the transition probability matrix of the reset vectors. Finally, we derive the average throughput for a bucket brigade with non-preemptible work content. In the numerical experiments, we first compare the average throughput derived from the theorem to the sample paths based on simulations. We see that the throughputs converge as the number of jobs increases. Then we compare the throughput difference between a non-preemptible line and a preemptible line. If the workers are sequenced slowest-to-fastest, the preemptible line dominates the non-preemptible line. However, if the workers are sequenced fastest-to-slowest, the non-preemptible line can possibly dominate the preemptible line. As such, the management needs to consider about the actual setting to enhance the performance.

Chapter 4 proposes a distributive decision rule in resource allocation problems with the following five features: (i) supply is replenishable after some random time, (ii) supply can be partially controlled, (iii) both demand and supply is partitioned into disjoint homogeneous sources, (iv) demand can wait and need not be fully fulfilled, and (v) allocation values are known. Under this setting, the goal of the planner is to decide simultaneously on both the supply and the allocation policy to fulfill the uncertain demand over a multi-period horizon. Such settings emerge in problems such as ride-sharing, fleet re-positioning, and patient management in healthcare. We introduce a *distributive decision rule*, which decides on the proportion of jobs awaiting dispatch to each of the possible resource supply pools. Our model has a convex reformulation that can be solved efficiently. Through simulations, we illustrate that the optimal solution evolves with changes in service distribution, initial conditions, temporal fluctuations in demand and resource availability. We further provide insights on such behaviour. In addition, we also illustrate insights on network design and the impact of connectivity on the optimal policy. At last, we benchmark our model against two models: (1) The static rule; (2) The fluid model. To benchmark against the static rule, we justify the adaptivity of the proposed distributive decision rule; To benchmark against the state-of-the-art fluid model, we show the robustness of our model to different settings.

Chapter 2

Stochastic Bucket Brigades with Preemptible Work Content

2.1 Introduction

To boost an assembly line's productivity, it is crucial to coordinate workers on the line such that their production capacity is effectively used. A well-known strategy is to coordinate the workers on the line as a *bucket brigade* (Bartholdi III, Eisenstein, 1996b,a). When the workers are organized as a bucket brigade, each of them simultaneously assembles a single job (an instance of the product) along the line. Each worker carries and works on his job from work station to work station until either he hands off his job to a downstream co-worker or he completes his job at the end of the line. The worker then walks back to get another job, either from his co-worker upstream or from a buffer at the beginning of the line.

Bucket brigades are notably used in warehouses and distribution centers to coordinate workers for order-picking (Bartholdi III et al., 2001). Companies that adopt the strategy include Ford, The Gap, and Walgreen's (Bartholdi III, Eisenstein, 1996a). Bucket brigades are also applied to manufacturing environments where workers assemble jobs on discrete work stations. Examples include United Technologies Automotive (Villalobos et al., 1999a,b), Mitsubishi Consumer Electronics America, and Subway (Bartholdi III, Eisenstein, 1996a). Bucket brigades are attractive in practice because they only require the workers to follow simple rules and the work-in-process is strictly under control by the number of workers.

More importantly, bucket brigades possess a *self-balancing* dynamic behavior that is first studied by Bartholdi III, Eisenstein (1996b). In their normative model, they assume the work content is deterministic. Each worker proceeds forward with a deterministic, finite work velocity, and walks back instantaneously (with an infinite velocity). The authors proved that if the workers are sequenced from slowest to fastest in the production-flow direction based on their work velocities, then the hand-offs between any two adjacent workers will converge to a fixed location. As a result, every worker will repeatedly work on a fixed segment of the line eventually. This self-balancing behavior helps boost the productivity (in some cases, very significantly) of the implemented systems mentioned above (Bartholdi III, Eisenstein, 1996a). Furthermore, it also allows a bucket brigade to spontaneously adapt to disruptions and seasonality.

Most papers in the literature study deterministic models of bucket brigades. However, many systems in practice exhibit a certain level of randomness in the service (or processing) times at work stations (Hopp, Spearman, 2011). To the best of our understanding, only two papers have analytically studied stochastic models of bucket brigades. Bartholdi III et al. (2001) consider an assembly line with stochastic work content at work stations. The authors assume that the work content at each station is exponentially distributed with a common mean and each worker has a constant work speed. They show that when the number of stations approaches infinity, the dynamics and throughput of the stochastic system will be similar to that of a deterministic system. In practice, many manufacturing systems have only a few work stations (Hopp, Spearman, 2011) and all order-picking lines in warehouses have a finite number of rack sections (Bartholdi III, Hackman, 2019). Thus, it is worthwhile and important to study a line with a small (finite) number of discrete work stations. Bukchin et al. (2018) consider a stochastic two-worker model with continuous work content along the line. They assume that upon each hand-off, the work speed of each worker is randomly re-generated from a distribution function, and the worker maintains this speed until the next hand-off. They find that a fastest-to-slowest sequence may be optimal as long as the standard deviation of the fastest worker's speed is sufficiently large.

If the service times at work stations are stochastic, will a bucket brigade converge to some "stationary state"? Will the system "balance" itself on a line with a *finite* number of stations as observed in the deterministic model? If the work speed of each worker varies over the stations, then how should we characterize the efficiency of each worker along the line? What is the effective production rate of each worker that contributes to the throughput of the line? If the workers become faster as they learn over the jobs, should we only partially cross-train the workers to intensify their learning on a subset of stations, instead of fully cross-training them over all the stations? These issues are not well studied in the literature.

In this paper, we consider a stochastic bucket brigade line with J stations and I workers. We assume that the time duration for each worker to serve a job at a station is exponentially distributed with a rate that depends on the station's expected work content and the worker's work speed. We analytically derive the system's average throughput and the coefficient of variation (CV) of the inter-completion time. If the work speeds are independent of the jobs, we prove that the stations where hand-offs occur follow a stationary probability distribution as the number of jobs approaches infinity. Furthermore, the average throughput and the CV of the inter-completion time converge to a constant.

We make other contributions by investigating the following three cases:

i **The work speeds depend only on the workers.** We assume the workers can be ranked from slowest to fastest. Interestingly, in this situation, the stationary probability distribution of the hand-off stations is analogous to the dynamic behavior of a deterministic model.

Although the throughput difference between the stochastic and the deterministic systems gets closer as the number of stations increases, the difference can be quite significant if the number of stations is small. Either maximizing the throughput or minimizing the CV of the inter-completion time, the slowest-to-fastest sequence always outperforms the reverse sequence for the stochastic bucket brigade. Furthermore, to maximize the throughput, more work content should be assigned to the stations near the faster workers. In contrast, to minimize the CV of the inter-completion time, more work content should be assigned to the stations near the slower workers.

ii **The work speeds depend on the workers and the stations.** Given that the workers may not dominate each other along the entire line, it becomes non-trivial to characterize the efficiency of each worker. We define the average work speed of each worker as a weighted average of his work speeds at all the stations. We also derive the expected blocked time of each worker along the line. The throughput of the stochastic bucket brigade can be expressed as a function of the average work speeds and the expected blocked times of the workers. Furthermore, the throughput can be interpreted as the sum of the effective production rates of all the workers. We develop a methodology to select the best system configuration (where should the specialist specialize and how to sequence the workers), and identify the factor that causes one system configuration more productive than another.

iii **The work speeds depend on the workers, the stations, and the jobs.** We assume the workers become faster as they learn over the jobs. When the workers learn, is a bucket brigade with fully cross-trained workers more productive than a bucket brigade with each worker only partially cross-trained on a subset of stations? In the former setting, the workers are more flexible but their learning effect can be diluted because of their wider work zones. In contrast, although constrained to a subset of stations, each worker in the latter setting can intensify their learning in a narrower work zone. Thus, it is non-trivial to determine which bucket brigade is more productive.

We find that if the workers have similar initial work speeds, then it is more productive to intensify their learning over a narrower work zone. Otherwise, it is crucial to mitigate the negative effect of the bottleneck (slowest) worker by cross-training the workers over a wider work zone.

Section 2.2 discusses the relevant literature. Section 2.3 specifies the assumptions and notation of our stochastic bucket brigade model. Section 2.4 derives the average throughput and the CV of the inter-completion time, and determines the system's asymptotic behavior as the number of jobs approaches infinity. Sections 2.5, 2.6 and 2.7 study the above three cases in detail. Section 2.8 checks the robustness of the result. Section 2.9 provides some concluding remarks. All proofs can be found in Appendix A.

2.2 Literature review

This paper is related to two streams of research: (i) bucket brigade assembly lines and (ii) dynamic server assignment on stochastic systems. We discuss each stream of work as follows.

2.2.1 Bucket brigade assembly lines

Bartholdi III, Eisenstein (1996b) introduce a deterministic bucket brigade model. The authors find that an assembly line under the bucket brigade protocol can balance itself: If the workers are sequenced from slowest to fastest in the direction of the production flow, then the system always converges to a state where each worker repeatedly works on a fixed segment of the line. Furthermore, if the work content is continuous along the line, then the system's throughput reaches a maximum level. Bartholdi III et al. (1999) analyze the dynamics of a bucket brigade with two or three workers, and each worker has a constant work speed. For a two-worker line, they find that under the fastest-to-slowest worker sequence, the hand-offs between the two workers converge to a 2-cycle periodic state. Bartholdi III, Eisenstein (2005) consider a model where each worker spends a constant walk-back time and a constant hand-off time to get work from his upstream colleague. They assume the worker's constant walk-back time is independent of his upstream colleague. They find that the system will still balance itself if the workers are sequenced from slowest to fastest. Bartholdi III et al. (2009) generalize the bucket brigade protocol by allowing the workers to overtake and pass their colleagues. They show that a two-worker line may exhibit a chaotic behavior in which the hand-off locations never repeat on the line.

Most of the above results are based on a line with continuous work content. Lim, Yang (2009) study a bucket brigade on discrete work stations. For a given work-content distribution on the stations, the authors identify the best cross-training and worker-sequencing strategy to maximize the system's throughput. They find that fully cross-training the workers and sequencing them from slowest to fastest may not be optimal. Other researchers generalize the model in different ways. Armbruster, Gel (2006) consider a two-worker bucket brigade in which workers' speeds do not dominate each other along the entire line. The authors present conditions under which bucket brigades are effective. Bartholdi III et al. (2006) extend the ideas of bucket brigades to a network of sub-assembly lines so that all the sub-assembly lines are synchronized to produce at the same rate, and jobs are completed at regular, predictable time intervals. Webster et al. (2012) examine the performance of a bucket brigade order-picking system by varying the distribution of products along an aisle. Through simulations, the authors identify conditions in which the product distribution has large impact on the throughput.

Lim (2011) introduces a cellular bucket brigade, where each worker works on one side of an aisle when he proceeds in one direction and works on the other side of the aisle when he proceeds in the reverse direction. The idea is to eliminate the unproductive walk-back inherent in a traditional, serial bucket brigade. The author proposes extended rules to coordinate the workers under the new design, and identifies a sufficient condition for the cellular bucket brigade to self balance. Lim (2012) assesses the performance of cellular bucket brigades for warehouse order-picking using data from a distribution center in North America. Lim (2017) analyzes a cellular bucket brigade in which any two adjacent workers may spend different time durations in a hand-off between them. Even with significant hand-off times, the cellular bucket brigade remains substantially more productive than a traditional bucket brigade especially if the team size is small and the workers' work speeds are close to their walk speed. Lim, Wu (2014) maximize the productivity of a cellular bucket brigade on a U-shape line with discrete work stations. They find conditions for the system to self balance.

Some papers study stochastic bucket brigade models. Bartholdi III et al. (2001) assume that the time duration for a worker to serve a job at a station follows an exponential distribution with a rate that depends only on the worker. The authors prove that as the number of stations increases, the moment-to-moment behavior and the throughput of this stochastic model will increasingly resemble that of a deterministic model. Bukchin et al. (2018) consider a stochastic two-worker bucket brigade model with continuous work content. Immediately after each hand-off, they randomly re-generate each worker's work speed from a distribution function. The worker maintains this speed until the next hand-off. In contrast to these two papers, we assume that the time duration for each worker to serve a job at a station is exponentially distributed with a rate that depends on the station's expected work content and the worker's work speed. Furthermore, the work speed may depend on the worker, the station, and the job.

2.2.2 Dynamic server assignment on stochastic systems

Another stream of research studies dynamic server assignment on stochastic systems. Some papers in this stream *minimize holding costs* of systems with two stations. These include Harrison, López (1999), Williams (2000), Bell et al. (2001), Ahn et al. (2004), and Mandelbaum, Stolyar (2004), which study flexible servers in parallel queues. Other examples are Rosberg et al.

(1982), Farrar (1993), Iravani et al. (1997), Duenyas et al. (1998), Kaufman et al. (2005), and Armony et al. (2018), which study flexible servers in tandem queues. In contrast, our objective is to maximize the throughput or to minimize the CV of the inter-completion time.

Some papers find dynamic server assignment policies to *maximize throughput* of tandem lines with finite buffers. Andradóttir et al. (2001) consider a two-station, two-server Markovian system with service rates that are independent of the jobs. They identify an optimal policy that maximizes the long-run average throughput, and propose near-optimal heuristics for larger systems. Andradóttir, Ayhan (2005) study the optimal policies for two-station Markovian systems with more servers than the stations. Kırkızlar et al. (2010) show that the policies in Andradóttir et al. (2001) and Andradóttir, Ayhan (2005) can also be effective for non-Markovian systems. Andradóttir et al. (2003) and Andradóttir et al. (2007a) study general queueing networks with infinite buffers, without or with server and station failures. Kırkızlar et al. (2014) consider the trade-off between throughput and holding costs. They maximize the long-run average profit of a serial system with finite buffers by finding an optimal server assignment policy. Işık et al. (2016) study dynamic server allocation for a tandem system with non-collaborative servers. They derive an optimal policy for Markovian systems with two servers and two stations, and propose heuristics for larger systems. In contrast to the above papers, we focus on the bucket brigade policy. Furthermore, we assume the service rates may depend on the jobs.

Some researchers investigate the benefits of partial flexibility in tandem systems. For example, Andradóttir et al. (2007b) study a Markovian system with two stations, and demonstrate that making only one server flexible when the buffer is sufficiently large can attain most of the benefits of full flexibility. Kırkızlar et al. (2012) study flexible servers in understaffed lines. They prove that the best possible production rate with full server flexibility and infinite buffers can be attained with partial flexibility and zero buffers. Hopp et al. (2004) consider a system with the same number of stations and servers under a constant work-in-process policy. They show that a skill-chaining strategy with two skills per server can outperform a "cherry picking" strategy that cross-trains some servers at bottleneck stations. In contrast, we assume each worker is fully cross-trained in our stochastic bucket brigade model. For a comprehensive review on cross-trained workforce, see Hopp, Oyen (2004).

2.3 Assumptions and notation

We consider a bucket brigade assembly line with work stations sequenced as $j = 1, \dots, J$. Workers are sequenced as $i = 1, \dots, I$ in the direction of the production flow. We call workers $i-1$ and $i+1$ the *predecessor* and the *successor*, respectively, of worker i. There are K jobs to be processed by the assembly line. Let $Z_{i,j}^{(k)}$ denote the time duration for worker i to serve (or process) job k at station j. We assume $Z_{i,j}^{(k)}$ follows an exponential distribution with rate $\mu_{i,j}^{(k)}$, for $i = 1, \dots, I$, $j = 1, \dots, J$, and $k = 1, \dots, K$. We define the rate $\mu_{i,j}^{(k)} = \left(s_j / v_{i,j}^{(k)}\right)^{-1}$, where s_j represents the expected work content of station j and $v_{i,j}^{(k)}$ represents the work speed of worker i at station j on job k. We normalize the total expected work content such that $\sum_{j=1}^{J} s_j = 1$. We assume that all the workers are *fully cross-trained* such that $v_{i,j}^{(k)} > 0$, for all i, j, and k.

Defining the service (or processing) rates in this manner allows us to investigate the effects of the stations' expected work contents and the work speeds of the workers separately. Note that the expected work content s_j of station j is independent of job k. On the other hand, the work speed of worker i at station j may vary across the jobs. For example, some jobs are easier to process than other jobs because of the difference in the complexity levels or the material of the items. It is worth noting that the work speed $v_{i,j}^{(k)}$ is the inverse of the expected time to finish a *unit work content* at station j, whereas the service rate $\mu_{i,j}^{(k)}$ is the inverse of the expected time to finish a job at station j.

When worker i is working at any station, we assume the station's work content is *preemptible* such that worker $i+1$ can interrupt and take over the former's job. Worker i is *blocked* in front of a station if his job is ready to enter the station but some worker is still working at the station. We assume the workers spend negligible time when they move from one station to another and when they walk back to get more work. When the last worker I completes job k at the last station J, the system has its kth *reset*: Worker I walks back to take over the job of worker $I-1$, who in turn walks back to take over the job of worker $I-2$, and so on, until worker 1 initiates a new job at the beginning of the line. Since the workers spend negligible time to walk back, each reset is instantaneous.

Let $T^{(k)}$ denote the time point when worker I completes job k at the last station J. Let $H_i^{(k)}$ denote the station where worker i is working at immediately before $T^{(k)}$, for $i = 1, \ldots, I-1$. If worker i is blocked in front of station j immediately before $T^{(k)}$, then we set $H_i^{(k)} = j$. Thus, we have $1 \leq H_1^{(k)} \leq \ldots \leq H_{I-1}^{(k)} \leq J$. Since worker i hands off his job to worker $i+1$ at station $H_i^{(k)}$ in the kth reset, we call $H_i^{(k)}$ the kth *hand-off station* between workers i and $i+1$. Define $\boldsymbol{H}^{(k)} = \left(H_1^{(k)}, \ldots, H_{I-1}^{(k)}\right)$ as the kth *hand-off station vector*, for $k = 1, \ldots, K-1$. We set $T^{(0)} = 0$ and $\boldsymbol{H}^{(0)} = (1, \ldots, 1)$, which means that at time 0, worker I starts working on job 1 at station 1, while the first $I-1$ workers are blocked in front of station 1 (at the start of the line). Since the service times $Z_{i,j}^{(k)}$ are exponentially distributed and independent of each other, for all i, j, and k, the probability distribution of $\boldsymbol{H}^{(k)}$ can be determined by $\boldsymbol{H}^{(k-1)}$. Thus, $\left\{\boldsymbol{H}^{(k)}, k = 1, \ldots, K-1\right\}$ is a Markov process.

Define $\mathcal{H} = \{\boldsymbol{h} = (h_1, \ldots, h_{I-1}) | 1 \leq h_1 \leq \ldots \leq h_{I-1} \leq J\}$ as a set of all possible hand-off station vectors. It is straightforward to show that the cardinality of $\mathcal{H}$ is $|\mathcal{H}| = \binom{I+J-2}{I-1}$. For any $\boldsymbol{a}, \boldsymbol{b} \in \mathcal{H}$, we say $\boldsymbol{a} < \boldsymbol{b}$ if and only if there exists some m such that $a_m < b_m$ and $a_n = b_n$, for $n = m+1. \ldots, I-1$. We order the vectors in $\mathcal{H}$ such that $\boldsymbol{h}^1 < \boldsymbol{h}^2 < \ldots < \boldsymbol{h}^{|\mathcal{H}|}$, where $\boldsymbol{h}^1 = (1, \ldots, 1)$. For each $\boldsymbol{h}^n \in \mathcal{H}$, define $\pi_n^{(k)} = Pr\left\{\boldsymbol{H}^{(k)} = \boldsymbol{h}^n\right\}$ as the probability of $\boldsymbol{h}^n$ being the kth hand-off station vector. Define $\boldsymbol{\pi}^{(k)}$ as an $|\mathcal{H}|$-dimensional vector with its nth entry equals $\pi_n^{(k)}$. Note that $\boldsymbol{\pi}^{(k)}$ represents the probability distribution of the kth hand-off station vector $\boldsymbol{H}^{(k)}$. Since $\boldsymbol{H}^{(0)} = (1, \ldots, 1)$, we have $\boldsymbol{\pi}^{(0)} = (1, 0, ..., 0)$. For $\boldsymbol{h}, \boldsymbol{h}' \in \mathcal{H}$, define $p_{\boldsymbol{h},\boldsymbol{h}'}^{(k)} = Pr\left\{\boldsymbol{H}^{(k)} = \boldsymbol{h}' | \boldsymbol{H}^{(k-1)} = \boldsymbol{h}\right\}$ as the probability of $\boldsymbol{h}'$ being the kth hand-off station vector, conditioned on $\boldsymbol{h}$ being the $(k-1)$st hand-off station vector. Let $\boldsymbol{P}^{(k)} = \left(p^{(k)}_{\boldsymbol{h},\boldsymbol{h}'}\right)_{|\mathcal{H}|\times|\mathcal{H}|}$ denote the corresponding transition probability matrix.

2.4 Performance measures and asymptotic behavior

To derive the transition probability $p^{(k)}_{\boldsymbol{h},\boldsymbol{h}'}$, we need to analyze the movements of the I workers between the $(k-1)$st and the kth resets. Let $X(i)$ denote the station where worker i is located, for $i = 1, \ldots, I$. We set $X(I) = J+1$ when worker I finishes his job at station J. Define a *state* of the system as $\boldsymbol{X} = (X(1), X(2), \ldots, X(I))$. Recall that $H_i^{(k)}$ represents the kth hand-off station between workers i and $i+1$. Immediately after $T^{(k-1)}$, the state of the system is $\boldsymbol{X} = \left(1, H_1^{(k-1)}, \ldots, H_{I-1}^{(k-1)}\right)$. The workers keep working forward until $T^{(k)}$ when worker I finishes his job at station J, and the system's state becomes $\boldsymbol{X} = \left(H_1^{(k)}, \ldots, H_{I-1}^{(k)}, J+1\right)$. We call a state with $X(I) \leq J$ a *transient state*, and a state with $X(I) = J+1$ an *absorbing state*. Thus, immediately after the $(k-1)$st reset, the system progresses from one transient state to another transient state until it reaches an absorbing state when the kth reset occurs. Define $\mathcal{X} = \{\boldsymbol{x} = (x(1), \ldots, x(I)) | 1 \leq x(1) \leq x(2) \leq \ldots \leq x(I-1) \leq J,\ x(I-1) \leq x(I) \leq J+1\}$ as a set of all possible states between any two resets. Similar to $\mathcal{H}$, the cardinality of $\mathcal{X}$ is $|\mathcal{X}| = \binom{I+J-1}{I} + \binom{I+J-2}{I-1}$.

Between the $(k-1)$st and the kth resets, instead of keeping track of the system's state continuously, we only need to consider the *epochs* when a worker finishes his job at a station. Given that the service times are exponentially distributed, the probability that more than one worker finish their jobs at their respective stations at the same time is 0. For any transient state $\boldsymbol{x}$, suppose there are $N(\boldsymbol{x})$ workers that are not blocked. Let $l_1, \ldots, l_{N(\boldsymbol{x})}$ denote these $N(\boldsymbol{x})$ workers from upstream to downstream. Note that worker I is never blocked, which implies $l_{N(\boldsymbol{x})} = I$. Thus, we have

$$1 \le x(1) = \ldots = x(l_1) < x(l_1+1) = \ldots = x(l_2) < \ldots = x(l_{N(\boldsymbol{x})-1}) < x(l_{N(\boldsymbol{x})-1}+1) = \ldots = x(I) \le J.$$

Among the $N(\boldsymbol{x})$ workers that are not blocked, one of them, say worker l_n, will finish his job in the next epoch. Let $\boldsymbol{x}_n$ denote the new state immediately after worker l_n finishes his job. We have $x_n(l_n) = x(l_n) + 1$, $x_n(i) = x(i)$, for $i \ne l_n, 1 \le i \le I$. Between the $(k-1)$st and the kth resets, the system progresses from state $\boldsymbol{x}$ to another state $\boldsymbol{x'}$. Let $q^{(k)}_{\boldsymbol{x},\boldsymbol{x'}}$ denote a one-step transition probability from $\boldsymbol{x}$ to $\boldsymbol{x'}$. For any transient state $\boldsymbol{x}$, we have

$$q^{(k)}_{\boldsymbol{x},\boldsymbol{x'}} = \begin{cases} \mu^{(k+I-l_n)}_{l_n,x(l_n)} / \sum_{m=1}^{N(\boldsymbol{x})} \mu^{(k+I-l_m)}_{l_m,x(l_m)}, & if\ \boldsymbol{x'} = \boldsymbol{x}_n, n = 1, \ldots, N(\boldsymbol{x}), \\ 0, & otherwise. \end{cases}$$

For any absorbing state $\boldsymbol{x}$, we set

$$q^{(k)}_{\boldsymbol{x},\boldsymbol{x'}} = \begin{cases} 1, & if\ \boldsymbol{x'} = \boldsymbol{x}, \\ 0, & otherwise, \end{cases}$$

such that the system stays in the absorbing state $\boldsymbol{x}$. Let $\boldsymbol{Q}^{(k)} = \left(q^{(k)}_{\boldsymbol{x},\boldsymbol{x'}}\right)_{|\mathcal{X}|\times|\mathcal{X}|}$ denote the corresponding one-step transition probability matrix. Starting from any state $\boldsymbol{x}$, the system will take at most $(J-1)I+1$ epochs to reach an absorbing state. Let $\boldsymbol{R}^{(k)} = \left(\boldsymbol{Q}^{(k)}\right)^{JI-I+1}$, and let $r^{(k)}_{\boldsymbol{x},\boldsymbol{x'}}$ denote the $\boldsymbol{x}$-$\boldsymbol{x'}$ entry of $\boldsymbol{R}^{(k)}$. As the system transitions from the hand-off station vector $\boldsymbol{h}$ immediately after the $(k-1)$st reset to the hand-off station vector $\boldsymbol{h'}$ at the kth reset, the system's state progresses from $(1, \boldsymbol{h})$ to $(\boldsymbol{h'}, J+1)$. Thus, we have the following lemma.

Lemma 1. *The probability of $\boldsymbol{h'}$ being the kth hand-off station vector, conditioned on $\boldsymbol{h}$ being the $(k-1)$st hand-off station vector is*

$$p^{(k)}_{\boldsymbol{h},\boldsymbol{h'}} = r^{(k)}_{(1,\boldsymbol{h}),(\boldsymbol{h'},J+1)}.$$

Since $\boldsymbol{\pi}^{(0)}$ is given and $\boldsymbol{\pi}^{(k)} = \boldsymbol{\pi}^{(k-1)}\boldsymbol{P}^{(k)}$, we can derive the probability distribution of $\boldsymbol{H}^{(k)}$ as $\boldsymbol{\pi}^{(k)} = \boldsymbol{\pi}^{(0)}\boldsymbol{P}^{(1)} \ldots \boldsymbol{P}^{(k)}$, for $k = 1, \ldots, K-1$.

Define $E\left[T^{(K)}\right]$ as the *expected makespan* of a bucket brigade assembly line with K jobs. To derive $E\left[T^{(k)}\right]$, we define $Y(k) = T^{(k)} - T^{(k-1)}$ as the inter-completion time between job $k-1$ and job k, which equals the total service time of worker I on job k. Suppose $\boldsymbol{H}^{(k-1)} = \boldsymbol{h}$, we have $Y(k) = \sum_{j=h_{I-1}}^{J} Z_{I,j}^{(k)}$. Thus, we have

$$E\left[Y(k)|\boldsymbol{H}^{(k-1)} = \boldsymbol{h}\right] = \sum_{j=h_{I-1}}^{J} \frac{1}{\mu_{I,j}^{(k)}}, \quad \boldsymbol{h} \in \mathcal{H}. \tag{2.1}$$

Define $\boldsymbol{z}^{(k)}$ as an $|\mathcal{H}|$-dimensional column vector with its nth component equals $E\left[Y(k)|\boldsymbol{H}^{(k-1)} = \boldsymbol{h}^n\right]$, for $n = 1, \ldots, |\mathcal{H}|$. From Equation (2.1), we have

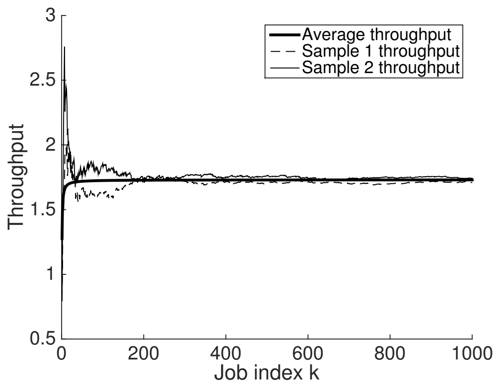
$$E\left[Y(k)\right] = \sum_{n=1}^{|\mathcal{H}|} E\left[Y(k)|\boldsymbol{H}^{(k-1)} = \boldsymbol{h}^n\right] \pi_n^{(k-1)} = \boldsymbol{\pi}^{(k-1)}\boldsymbol{z}^{(k)}. \tag{2.2}$$

Define the *average throughput* of a bucket brigade assembly line with K jobs as $\rho(K) = K/E\left[T^{(K)}\right]$. The following theorem determines the average throughput.

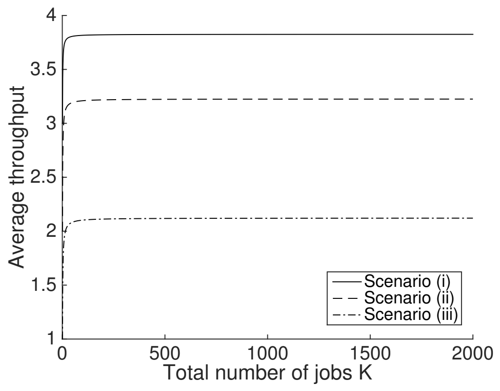
Theorem 1. *The average throughput of a bucket brigade assembly line with K jobs is*

$$\rho(K) = K \left/ E\left[T^{(K)}\right] = K \right/ \sum_{k=1}^{K} \boldsymbol{\pi}^{(0)}\boldsymbol{P}^{(1)} \cdots \boldsymbol{P}^{(k-1)}\boldsymbol{z}^{(k)}. \tag{2.3}$$

Given $\boldsymbol{\pi}^{(0)}$ and $\boldsymbol{z}^{(k)}$, we can determine the average throughput of the line using the transition probability matrices $\boldsymbol{P}^{(k)}$, for $k = 1, \ldots, K-1$. The bold solid line in Figure 2.1(a) represents the average throughput $\rho(k)$ of a bucket brigade assembly line up to job k determined by Equation (2.3). We set $I = 2, J = 3, s_1 = s_2 = s_3 = 1/3, K = 1,000$, and $v_{i,j}^{(k)} = i + j + k/K$, for $i = 1, 2$, $j = 1, 2, 3$, and $k = 1, \ldots, K$. The dashed line and the thin solid line represent two sample paths of the *actual throughput* $k/T^{(k)}$ based on simulations.



(a)



(b)

Figure 2.1: The throughput of a stochastic bucket brigade assembly line

For a special case in which the workers' service rates are independent of the jobs, we can drop the superscripts such that $\mu_{i,j}^{(k)} = \mu_{i,j}$, $\boldsymbol{P}^{(k)} = \boldsymbol{P}$, and $\boldsymbol{z}^{(k)} = \boldsymbol{z}$. Then, $\left\{\boldsymbol{H}^{(k)}, k = 1, \ldots, K-1\right\}$ becomes a homogeneous Markov process. Theorem 2 shows that, for this special case, the distribution of the kth hand-off station vector converges to a stationary distribution and the average throughput converges to a constant as the number of jobs approaches infinity. Let $\boldsymbol{e}$ denote an $|\mathcal{H}|$-dimensional unit column vector.

Theorem 2. *If the workers' service rates are independent of the jobs, then* $\lim_{k\to\infty} \boldsymbol{\pi}^{(k)} = \boldsymbol{\pi}$*, where* $\boldsymbol{\pi}$ *is a unique stationary distribution that satisfies the equations* $\boldsymbol{\pi P} = \boldsymbol{\pi}$ *and* $\boldsymbol{\pi e} = 1$*. Furthermore, the asymptotic throughput* $\rho_\infty = \lim_{K\to\infty} \rho(K) = 1/(\boldsymbol{\pi z})$*, where* $\boldsymbol{\pi z}$ *represents the asymptotic expected inter-completion time.*

It is interesting to see that the convergence to the stationary distribution is independent of the sequence of the workers. While we need to sequence the workers from slowest to fastest in the direction of the production flow for the deterministic bucket brigade to self balance (Bartholdi III, Eisenstein, 1996b), we do not need such a worker sequence for the stochastic bucket brigade to converge to the stationary distribution.

Figure 2.1(b) shows an example with $I = 3, J = 5, s_1 = s_2 = 0.1$, $s_3 = s_4 = 0.3$, and $s_5 = 0.2$. We consider three different scenarios: (i) $v_{1,j} = 1$,

$v_{2,j} = 2$, $v_{3,j} = 3$, (ii) $v_{1,j} = v_{2,j} = v_{3,j} = 2$, and (iii) $v_{1,j} = 3$, $v_{2,j} = 2$, $v_{3,j} = 1$. The average throughput $\rho(K)$ for each scenario converges to a constant as K increases, which is consistent with Theorem 2. Furthermore, the average throughput decreases as the worker sequence changes from slowest-to-fastest (scenario (i)) to fastest-to-slowest (scenario (iii)). Section 2.5.2 demonstrates that the slowest-to-fastest sequence is always more productive than the reverse sequence for the stochastic bucket brigade.

Using the probability distribution $\boldsymbol{\pi}^{(k)}$ of the kth hand-off station vector, we can also derive the variance of the inter-completion time $Y(k)$. Let h_i^n denote the ith component of $\boldsymbol{h}^n$.

Lemma 2. *The variance of the inter-completion time $Y(k)$ can be determined as*

$$Var\left(Y(k)\right) = \sum_{n=1}^{|\mathcal{H}|} \pi_n^{(k-1)} \sum_{j=h_{I-1}^n}^{J} \left(\frac{1}{\mu_{I,j}^{(k)}}\right)^2 + \sum_{n=1}^{|\mathcal{H}|} \pi_n^{(k-1)} \left(\sum_{j=h_{I-1}^n}^{J} \frac{1}{\mu_{I,j}^{(k)}}\right)^2 - \left(\sum_{n=1}^{|\mathcal{H}|} \pi_n^{(k-1)} \sum_{j=h_{I-1}^n}^{J} \frac{1}{\mu_{I,j}^{(k)}}\right)^2. \tag{2.4}$$

We define the *coefficient of variation* (CV) of the inter-completion time $Y(k)$ as $CV = \sqrt{Var\left(Y(k)\right)}\ /E\left[Y(k)\right]$, where $E\left[Y(k)\right]$ and $Var\left(Y(k)\right)$ are determined by Equations (2.2) and (2.4) respectively. Note that a small CV of the inter-completion time ensures a more predictable output process of the line, which facilitates planning of the downstream processes of the supply chain. Lemma 2 implies the following theorem.

Theorem 3. *If the service rates are independent of the jobs, and $\boldsymbol{\pi}$ is the stationary distribution of the hand-off station vectors. The asymptotic variance of the inter-completion time is*

$$\lim_{k\to\infty} Var\left(Y(k)\right) = \sum_{n=1}^{|\mathcal{H}|} \pi_n \sum_{j=h_{I-1}^n}^{J} \left(\frac{1}{\mu_{I,j}}\right)^2 + \sum_{n=1}^{|\mathcal{H}|} \pi_n \left(\sum_{j=h_{I-1}^n}^{J} \frac{1}{\mu_{I,j}}\right)^2 - \left(\sum_{n=1}^{|\mathcal{H}|} \pi_n \sum_{j=h_{I-1}^n}^{J} \frac{1}{\mu_{I,j}}\right)^2. \tag{2.5}$$

The asymptotic CV of the inter-completion time equals

$$\lim_{k\to\infty} \frac{\sqrt{Var\left(Y(k)\right)}}{E\left[Y(k)\right]} = \frac{1}{\boldsymbol{\pi z}} \left[\sum_{n=1}^{|\mathcal{H}|} \pi_n \sum_{j=h_{I-1}^n}^{J} \left(\frac{1}{\mu_{I,j}}\right)^2 + \sum_{n=1}^{|\mathcal{H}|} \pi_n \left(\sum_{j=h_{I-1}^n}^{J} \frac{1}{\mu_{I,j}}\right)^2 - \left(\sum_{n=1}^{|\mathcal{H}|} \pi_n \sum_{j=h_{I-1}^n}^{J} \frac{1}{\mu_{I,j}}\right)^2\right].$$

2.5 Case I: The work speeds depend only on the workers

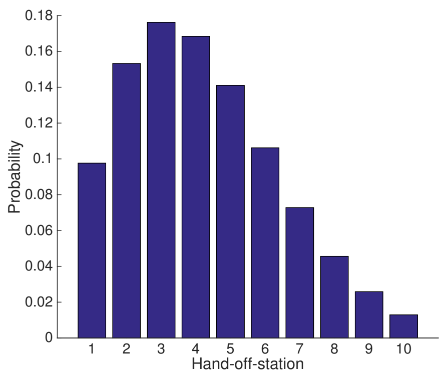
In this section, we consider a special case in which $v_{i,j}^{(k)} = v_i$, for all i, j, and k.

2.5.1 The distribution of the hand-off station vector

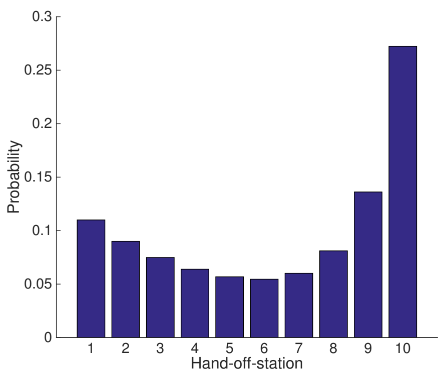
Since each worker's service time at each station is stochastic, the hand-offs between any two neighboring workers will not converge to a fixed location (station) as observed in the deterministic model studied by Bartholdi III, Eisenstein (1996b). Instead, Theorem 2 shows that the probability distribution of the kth hand-off station vector $\boldsymbol{H}^{(k)}$ always converges to a stationary distribution $\boldsymbol{\pi}$ as k goes to infinity, independent of the worker sequence.

Figure 2.2(a) shows the stationary distribution of a ten-station two-worker line with $s_j = 0.1$, $j = 1, \ldots, 10$. The workers are sequenced from slowest to fastest with $v_1 = 1$ and $v_2 = 2$. Interestingly, the stationary distribution of the hand-off station is unimodal with a single peak at station 3. This is analogous to the behavior of the deterministic model in which the hand-offs between the two workers converge to a fixed location (Bartholdi III, Eisenstein, 1996b).

Figure 2.2(b) shows the stationary distribution of the hand-off station when the workers are sequenced from fastest to slowest with $v_1 = 2$ and $v_2 = 1$. The stationary distribution has two peaks at stations 1 and 10. This is analogous to the dynamic behavior of the deterministic model studied by Bartholdi III et al. (1999) in which the hand-offs converge to a 2-cycle.



(a) $v_1 = 1, v_2 = 2$



(b) $v_1 = 2, v_2 = 1$

Figure 2.2: The stationary distributions of the hand-off station for a line with ten stations and two workers

2.5.2 Comparing with the deterministic system

To examine the effect of the random service times, we compare the stochastic system with the deterministic system with discrete work stations. We consider two workers with work speeds equal to 1 and 2. The expected work content is identical for all the stations such that $s_j = 1/J, j = 1, \ldots, J$. We compare four different settings: A deterministic system with a slowest-to-fastest worker sequence (denoted as DS), a stochastic system with a slowest-to-fastest worker sequence (denoted as SS), a deterministic system with a fastest-to-slowest worker sequence (denoted as DF), and a stochastic system with a fastest-to-slowest worker sequence (denoted as SF). We examine the asymptotic throughput of each setting as the number of jobs K approaches infinity. The asymptotic throughput of each stochastic system is determined by Theorem 2, and the asymptotic throughput of each deterministic system can be derived analytically. Figure 2.3 shows the asymptotic throughput of each setting as the number of stations J varies from 4 to 50. As J increases, the throughput difference between the deterministic and the stochastic systems becomes smaller for both worker sequences. This is consistent with the result of Bartholdi III et al. (2001). The DS setting achieves the maximum throughput $v_1 + v_2$, but the SS setting cannot always achieve the maximum throughput because of blocking.

However, the SS setting's performance approaches that of the DS setting as J increases.

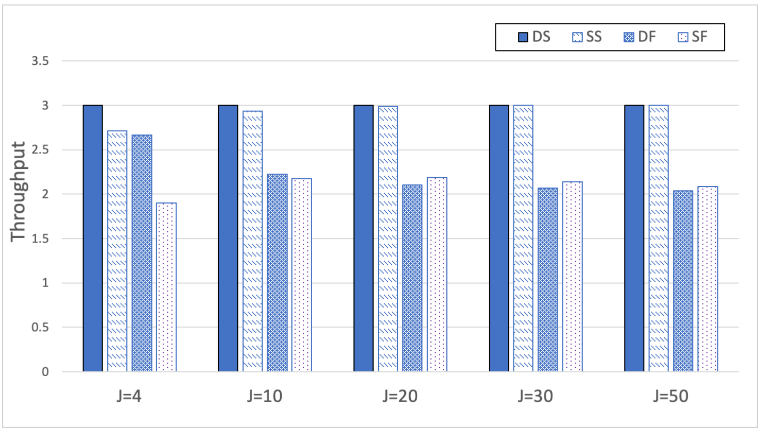


Figure 2.3: The asymptotic throughput of the deterministic and the stochastic systems for different numbers of stations

Under the fastest-to-slowest worker sequence, the asymptotic throughput of the DF setting decreases with J, whereas the asymptotic throughput of the SF setting first increases and then slightly decreases with J. Note that under the fastest-to-slowest sequence, the stochastic system (SF) may outperform the deterministic system (DF). For $J \geq 4$, the DF setting converges to a period-2 cycle as $K \to \infty$ with hand-offs alternating between $2/J$ and $(J-1)/J$. Thus, the asymptotic throughput equals $2 + 2/(J-1)$, which approaches 2 as J increases. Note that the asymptotic throughput approaches two times of the slower worker's speed as J increases because the line is becoming more like a continuous line and the faster worker upstream is constantly blocked by the slower worker downstream. In contrast, the SF setting suffers from severe blocking if J is small. Its asymptotic throughput is lower than 2 when $J = 4$. As J increases, blocking is mitigated and the SF setting can achieve a throughput greater than 2. However, as J further increases, the stochastic system resembles the deterministic system, and the throughput drops to 2.

From the above comparison between the stochastic and the deterministic models with discrete work stations, we obtain the following insights. Under the slowest-to-fastest sequence, the deterministic system is more productive than the stochastic system. This suggests that the stochastic service times cause throughput loss for a bucket brigade line with discrete work stations under the slowest-to-fastest sequence. However, under the reverse sequence, the stochastic system may outperform the deterministic system (see Figure 2.3). For both worker sequences, the performance difference between the stochastic and the deterministic systems gets closer as J increases, which is consistent with the finding of Bartholdi III et al. (2001).

Figure 2.4 shows the asymptotic throughput of each setting for different work-content distributions. We consider three stations and two workers with work speeds 1 and 2. Under each worker sequence, the deterministic system always outperforms the stochastic system.

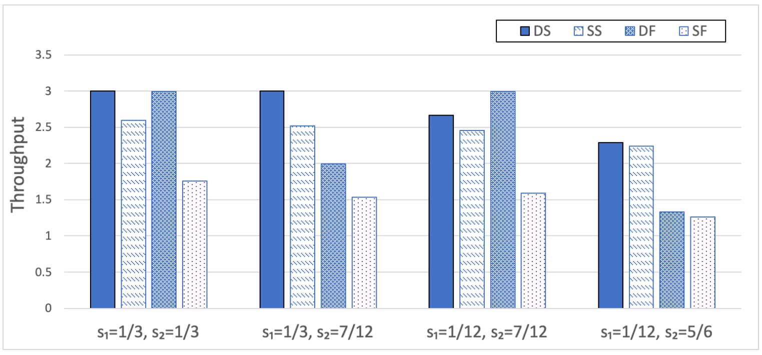


Figure 2.4: The asymptotic throughput of the deterministic and the stochastic systems for different work-content distributions

For the deterministic system, the relative performance of the slowest-to-fastest and the reverse sequences depends on the work-content distribution: If $s_1 = s_2 = 1/3$, both sequences achieve the maximum throughput 3. If $s_1 = 1/3$ and $s_2 = 7/12$, only the slowest-to-fastest sequence (DS) achieves the maximum throughput. If $s_1 = 1/12$ and $s_2 = 7/12$, only the fastest-to-slowest sequence (DF) achieves the maximum throughput. If $s_1 = 1/12$ and $s_2 = 5/6$, both sequences cannot achieve the maximum throughput. In contrast, for the stochastic system, we show in the following that the slowest-to-fastest sequence (SS) always outperforms the reverse sequence (SF) under all work-content distributions.

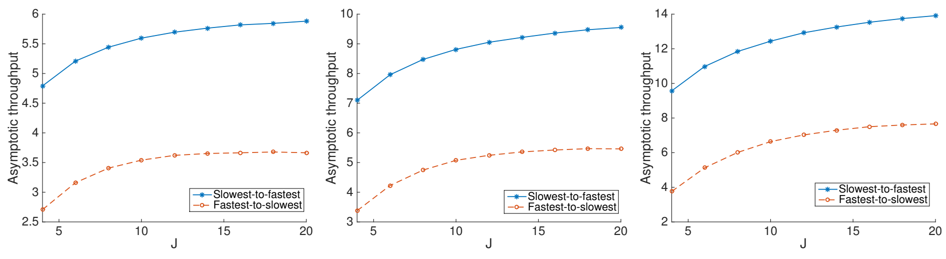
Lemma 3. *For a three-station two-worker line, if $v_1 + v_2 = c$, where c is a constant, the asymptotic throughput of the stochastic bucket brigade system increases with v_2.*

Lemma 3 implies the following corollary.

Corollary 1. *For a three-station two-worker stochastic bucket brigade system, the slowest-to-fastest sequence is always more productive than the fastest-to-slowest sequence.*

Figure 2.4 and Lim, Yang (2009) show that the slowest-to-fastest sequence can be outperformed by the reverse sequence for a *deterministic* three-station line (see Figure 6(a) of Lim, Yang (2009)). However, for a three-station line with stochastic service times, Corollary 1 shows that the slowest-to-fastest sequence is always more productive.

We further test numerically that whether the result of Corollary 1 continues to hold for a line with more work stations and workers. Figure 2.5 shows the asymptotic throughput under both worker sequences for 3, 4, and 5 workers, and the number of work stations J varies from 4 to 20. In each graph, we set the work speeds of the I workers equal $1, 2, \dots, I$. We consider evenly distributed work content such that $s_j = 1/J$, $j = 1, 2, \dots, J$. We observe that the slowest-to-fastest sequence is always more productive than the reverse sequence in these larger systems with stochastic service times.



(a) $I = 3$ (b) $I = 4$ (c) $I = 5$

Figure 2.5: The asymptotic throughput for different numbers of workers

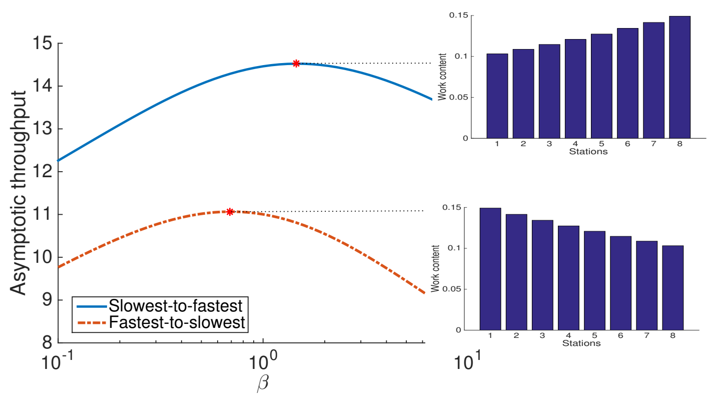
As suggested by Figures 2.3 and 2.4, if the number of stations is small, the throughput difference between the stochastic and the deterministic systems can be *quite significant* (the gap is up to 47%). Furthermore, Corollary 1 and Figure 2.5 show that the slowest-to-fastest sequence is more productive than the reverse sequence for the stochastic system. However, this result may not hold for a deterministic system (see Figure 6(a) of Lim, Yang (2009)). Thus, it is important to study the stochastic bucket brigade model especially for a line with a small number of stations.

2.5.3 Maximizing the throughput

Theorem 2 allows us to maximize the asymptotic throughput of the stochastic bucket brigade by optimizing the expected work contents $s_1, \ldots, s_J$, and the sequence of the workers. For illustration purposes, we assume $s_j/s_{j-1} = \lambda$, $j = 2, \ldots, J$. We define $\beta = s_J/s_1 = \lambda^{J-1}$. If $\beta > 1$, the work content s_j is increasing in j. If $\beta = 1$, the work content is evenly distributed over the stations. If $\beta < 1$, the work content s_j is decreasing in j.

Figure 2.6(a) shows the asymptotic throughput of a line with $J = 8$ stations and $I = 4$ workers. We assume the work speeds of the workers are 3, 4, 5 and 6. The slowest-to-fastest sequence (solid line) always outperforms the reverse sequence (dashed line). Under each worker sequence, an *overly skewed* work-content distribution over the stations (when β is too small or too large) is not productive because of severe blocking. As a result, for each worker sequence in Figure 2.6(a), the asymptotic throughput is unimodal in β. For the slowest-to-fastest sequence, the asymptotic throughput reaches the maximum when $\beta = 1.45$, corresponding to the top graph of Figure 2.6(b) where the work content s_j increases with j. For the fastest-to-slowest sequence, the maximum asymptotic throughput occurs at $\beta = 0.69$, which corresponds to the bottom graph of Figure 2.6(b) where the work content decreases from upstream to downstream. Figure 2.6(b) suggests that *to maximize the asymptotic throughput, more work content should be assigned to the stations near the faster workers.*

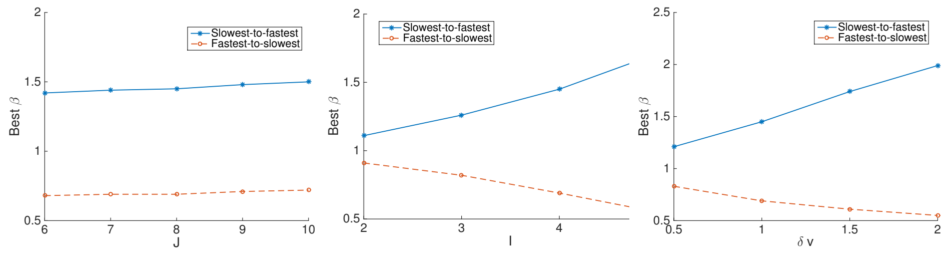
To test the robustness of the above results, we vary the number of stations, the number of workers, and the difference in the work speeds of the workers. In Figure 2.7(a), we consider the same team of four workers in Figure 2.6. We vary the number of stations J from 6 to 10. For each J, we identify the



(a) (b)

Figure 2.6: (a) Asymptotic throughput of a line with eight stations and four workers
(b) Work-content distributions with $\beta = 1.45$ (top) and $\beta = 0.69$ (bottom)

best β for both the slowest-to-fastest and the reverse sequences. We find that the best β for the slowest-to-fastest sequence is always greater than 1, implying that the work content s_j increases with j. In contrast, the best β for the reverse sequence is always less than 1, implying that s_j is decreasing in j.



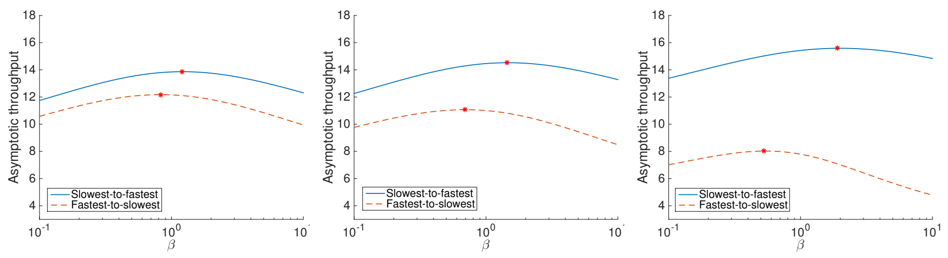
(a) Best β for different values of J (b) Best β for different values of I (c) Best β for different values of δv

Figure 2.7: Best β for maximizing the asymptotic throughput under different parameter settings

Figure 2.7(b) shows the best β by varying the number of workers I from 2 to 5. To be comparable with Figure 2.6, we consider eight stations and assume $V \equiv \sum_{i=1}^{I} v_i = 18$, where $v_i = V/I - (I+1)/2 + i$ for the slowest-to-fastest sequence, and $v_i = V/I + (I+1)/2 - i$ for the fastest-to-slowest sequence, for $i = 1, \ldots, I$. Thus, $\delta v \equiv |v_i - v_{i-1}| = 1$, $i = 2, \ldots, I$, for both sequences. We find that for each I, the best β for the slowest-to-fastest sequence is always greater than 1. Furthermore, the best β increases with I (the skewness of the work-content distribution increases with I) because the gap between the work speeds of the first and the last workers increases with I. In contrast, the best β for the fastest-to-slowest sequence is always less than 1, and the best β decreases with I because of the same reason.

Figure 2.7(c) shows the best β by varying the work-speed difference δv. We consider eight stations and four workers with $\sum_{i=1}^{4} v_i = 18$. We find that the best β for the slowest-to-fastest sequence is always greater than 1. The best β increases with δv because the gap between the work speeds of the first and the last workers increases with δv. On the other hand, the best β for the fastest-to-slowest sequence is always less than 1 and it decreases with δv because of the same reason.

To further investigate the impact of δv, Figure 2.8 shows the asymptotic throughput of each worker sequence for $\delta v = 0.5$, 1, and 2, with $J = 8$, $I = 4$, and $\sum_{i=1}^{I} v_i = 18$. We see that the gap between the asymptotic throughputs of the two sequences becomes larger as δv gets larger.

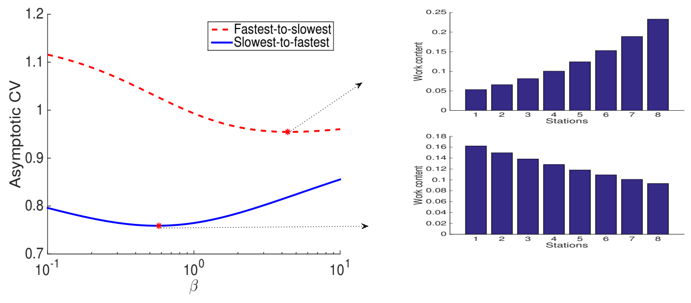


(a) $\delta v = 0.5$ (b) $\delta v = 1$ (c) $\delta v = 2$

Figure 2.8: The gap between the two sequences' asymptotic throughputs becomes larger as δv increases

2.5.4 Minimizing the CV of the inter-completion time

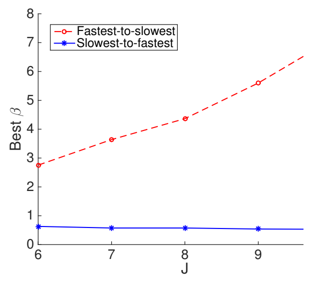
Theorem 3 enables us to minimize the CV of the inter-completion time by optimizing the work-content distribution over the stations. Based on the same definition of $\beta = s_J/s_1 = \lambda^{J-1}$ as in Section 2.5.3, Figure 2.9(a) shows the asymptotic CV of the inter-completion time under the same setting as in Figure 2.6(a). The slowest-to-fastest sequence (solid line) always has a smaller asymptotic CV of the inter-completion time than the reverse sequence (dashed line). Moreover, for each worker sequence the CV is unimodal in β. This suggests that an overly skewed work-content distribution (when β is too small or too large) causes severe blocking, resulting in a higher CV of the inter-completion time. For the fastest-to-slowest sequence, the asymptotic CV of the inter-completion time reaches the minimum when $\beta = 4.37$. This corresponds to the top graph of Figure 2.9(b), where the work content s_j increases with j. For the slowest-to-fastest sequence, the minimum asymptotic CV of the inter-completion time occurs at $\beta = 0.58$, corresponding to the bottom graph of Figure 2.9(b), where the work content decreases from upstream to downstream. Figure 2.9(b) suggests that *we can minimize the CV of the inter-completion time by allocating more work content to the stations near the slower workers.*



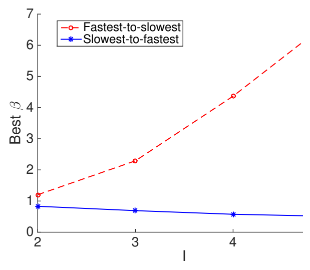
(a) (b)

Figure 2.9: (a) Asymptotic CV of the inter-completion time of a line with 8 stations and 4 workers (b) Work-content distributions with $\beta = 4.37$ (top) and $\beta = 0.58$ (bottom)

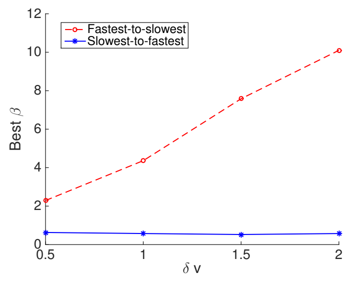
To test the robustness of the above results, we vary the number of stations, the number of workers, and the difference in the work speeds of the workers. In Figure 2.10(a), we consider the same team of four workers as in Figure 2.9. We vary the number of stations J from 6 to 10. For each J, we identify the best β for both the slowest-to-fastest and the reverse sequences. We find that the best β for the slowest-to-fastest sequence is always less than 1, and the best β decreases with J. In contrast, the best β for the fastest-to-slowest sequence is always greater than 1, and it increases with J. Similarly, Figure 2.10(b) identifies the best β by varying the number of workers I from 2 to 5. The setting is the same as Figure 2.7(b). Figure 2.10(c) shows the best β by varying the work-speed difference δv. The setting is the same as Figure 2.7(c). We find similar behavior of the best β as I or δv increases.



(a) Best β for different values of J



(b) Best β for different values of I



(c) Best β for different values of δv

Figure 2.10: Best β for minimizing the asymptotic CV of the inter-completion time under different parameter settings

Combining the results in Sections 2.5.3 and 2.5.4, we obtain the following managerial insights. Either maximizing the asymptotic throughput or minimizing the asymptotic CV of the inter-completion time, the slowest-to-fastest sequence always outperforms the reverse sequence for the stochastic bucket brigade. Furthermore, to maximize the asymptotic throughput, more work content should be assigned to the stations near the faster workers (see Figure 2.6(b)). However, to minimize the asymptotic CV of the inter-completion time, more work content should be assigned to the stations near the slower workers (see Figure 2.9(b)).

2.6 Case II: The work speeds depend only on the workers and the stations

In this section, we assume the work speed of each worker depends on the worker and the stations, but is independent of the jobs such that $v_{i,j}^{(k)} = v_{i,j}$, for all k, $i = 1, \ldots, I$, $j = 1, \ldots, J$. It is challenging to analyze this case because a worker i may not dominate another worker i' along the entire line. That is, worker i is faster at some stations, but slower at other stations than worker i'. In this situation, it is not clear how we should sequence the workers along the line.

2.6.1 Average work speeds and asymptotic expected blocked times

As the work speed of each worker may vary across the stations, we define the *average work speed* of worker i along the entire line as

$$\nu_i = \frac{\sum_{j=1}^{J} f_{i,j} s_j}{\sum_{j=1}^{J} f_{i,j} s_j / v_{i,j}}, \tag{2.6}$$

where $f_{i,j}$ represents the probability of worker i finishing his job at station j between two consecutive resets under the stationary distribution $\boldsymbol{\pi}$. Note that if $v_{i,j} = v_i$, we have $\nu_i = v_i$. Recall that h_i^n denotes the ith component of $\boldsymbol{h}^n$. For convenience, we set $h_0^n = 1$ and $h_I^n = J + 1$. Let $f_{i,j}^{(k)}$ denote the probability of worker i finishing his job at station j between the $(k-1)$st and the kth resets. The following lemma determines $f_{i,j}^{(k)}$ and $f_{i,j}$.

Lemma 4. *The probability of worker i finishing his job at station j between the $(k-1)$st and the kth resets can be obtained as*

$$f_{i,j}^{(k)} = \sum_{\substack{n=1,\dots,|\mathcal{H}| \\ h_i^n \geq j+1}} \pi_n^{(k)} - \sum_{\substack{n=1,\dots,|\mathcal{H}| \\ h_{i-1}^n \geq j+1}} \pi_n^{(k-1)}. \tag{2.7}$$

As k approaches infinity, the probability of worker i finishing his job at station j between two consecutive resets under the stationary distribution $\boldsymbol{\pi}$ can be obtained as

$$f_{i,j} = \sum_{\substack{n=1,\dots,|\mathcal{H}| \\ h_i^n \geq j+1}} \pi_n - \sum_{\substack{n=1,\dots,|\mathcal{H}| \\ h_{i-1}^n \geq j+1}} \pi_n. \tag{2.8}$$

Define the expected blocked time of worker i as the expected total time that worker i is blocked between two consecutive resets. Let B_i denote the *asymptotic expected blocked time* of worker i as K approaches infinity. Since worker I is never blocked, we have $B_I = 0$. Recall that the inter-completion time $Y(k) = T^{(k)} - T^{(k-1)}$. Let $Y_\infty = \lim_{k\to\infty} E[Y(k)]$ denote the asymptotic inter-completion time of jobs. According to Theorem 2, $Y_\infty = \boldsymbol{\pi z} = 1/\rho_\infty$.

Lemma 5. *The asymptotic expected blocked time of worker i can be expressed as*

$$B_i = Y_\infty - \sum_{j=1}^{J} \frac{f_{i,j}}{\mu_{i,j}}, \qquad i = 1, \dots, I-1. \tag{2.9}$$

The following theorem expresses the asymptotic throughput of the line as a function of the average work speeds and the asymptotic expected blocked times of the workers.

Theorem 4. *If the work speeds of the workers are independent of the jobs, the asymptotic throughput of a stochastic bucket brigade can be expressed as*

$$\rho_\infty = \sum_{i=1}^{I} \frac{Y_\infty - B_i}{Y_\infty} \nu_i. \tag{2.10}$$

Equation (2.10) can be interpreted as follows. The term $\frac{Y_\infty - B_i}{Y_\infty}$ represents the fraction of the inter-completion time that worker i is not blocked on the line, and $\frac{Y_\infty - B_i}{Y_\infty}\nu_i$ represents the effective production rate of worker i without being blocked. The asymptotic throughput of the line is the sum of the effective production rates of all the workers.

Define $\nu = \sum_{i=1}^{I} \nu_i$ as the *capacity* of the system, and define

$$\gamma = \frac{\rho_\infty}{\nu} \tag{2.11}$$

as an *efficiency ratio* that measures how much capacity of the system is effectively used for production. Note that γ is decreasing in B_i/Y_∞, for $i = 1, \dots, I-1$. As B_i/Y_∞ approaches 1 for $i = 1, \dots, I-1$, γ approaches ν_I/ν, corresponding to an extreme case in which the first $I-1$ workers are always blocked and only worker I contributes to the throughput. As B_i/Y_∞ approaches 0 for $i = 1, \dots, I-1$, γ approaches 1, corresponding to a case in which all the workers contribute fully to the throughput of the line. Thus, the efficiency ratio γ represents the degree of *no blocking* in the system.

2.6.2 Throughput determining factors: capacity and efficiency ratio

We illustrate the decisive role of the capacity and the efficiency ratio on the system's asymptotic throughput. We consider a team of workers that consists of *generalists*, who have a constant work speed 1 at all the stations, and a *specialist*, who is faster at his specialized stations but slower at other stations than the generalists. Specifically, for the specialist, we assume that the total work content of all his specialized stations is $\bar{s} < 1$. His work speed at these specialized stations is $1 + \theta$, where θ represents the *specialization intensity*. We assume his work speed at his non-specialized stations is $1 - \tau\theta$, where τ satisfies

$$\frac{\bar{s}}{1+\theta} + \frac{1-\bar{s}}{1-\tau\theta} = 1. \tag{2.12}$$

Equation (2.12) ensures that the expected time for the specialist to complete a job by himself along the line is 1, identical to that of the generalists. In contrast to the previous section, the workers do not dominate each other in terms of the expected time to complete a job along the line. This allows us to investigate the impact of worker specialization. Note that the average work speed ν_i of each generalist i is always 1, while the average work speed $\nu_{i'}$ of the specialist i' depends on how frequent he works at his specialized stations.

We use a five-station three-worker line, where there are two generalists and a specialist, to illustrate why one system configuration is more productive than another. We assume that the work content is equally distributed at all the five stations, and set the specialization intensity $\theta = 1$. Figure 2.11 shows the capacity and the efficiency ratio of different system configurations. A system configuration is defined by the specialized station of the specialist and the worker sequence. The first digit of each legend in Figure 2.11 represents the specialized station. The following letters SGG, GSG, or GGS represent the worker sequence with the letters "G" and "S" correspond to a generalist and the specialist respectively. For example, "1SGG" corresponds to a line with the specialist specializing at station 1 and being the most upstream worker.

Figure 2.11 also shows contour lines of the asymptotic throughput. A contour line that is closer to the upper-right corner has a higher asymptotic throughput. By comparing the contour lines in Figure 2.11, we can tell which system configuration is more productive than another and why. For example, the line 1SGG (far right) is more productive than 1GGS (far left). This is because the former configuration has a larger capacity ν due to the larger average work speed of the specialist. In other words, the specialist in the 1SGG configuration works more often at his specialized station, making the line more productive. Note that although the line 1SGG has more blocking (with a lower efficiency ratio γ) compared to the line 1GGS, the effect of capacity dominates such that the former is still more productive than the latter. Likewise, the line 5GGS is more productive than 5SGG because of the same reason.

In contrast, the line 3GGS is more productive than 3SGG because the former has less blocking (with a higher efficiency ratio γ). Note that in this case, although the line 3GGS has a smaller capacity ν, the negative effect of blocking in 3SGG is so large that it makes 3SGG less productive than 3GGS. Using Figure 2.11, we are able to compare the throughput of any system configurations and select the best one. We can further identify the factor (the capacity or the efficiency ratio) that causes one system configuration more productive than another.

2.7 Case III: The work speeds depend on the workers, the stations, and the jobs

In this section, we assume the work speed $v_{i,j}^{(k)}$ of each worker depends on job k. Specifically, the work speeds may increase with k as the workers learn. (We can also consider the case in which the work speeds may decrease with k as the workers get fatigued.)

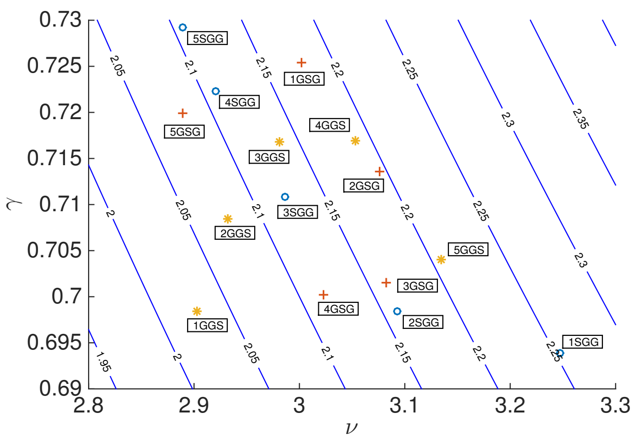


Figure 2.11: Contour lines of the throughput for a five-station three-worker line

2.7.1 A modified exponential model

We first consider an exponential model (Mazur, Hastie, 1978) to capture the changes in the work speeds. Specifically, for $i = 1, \ldots, I$, $j = 1, \ldots, J$, and $k = 1, \ldots, K$, we set

$$v_{i,j}^{(k)} = \underline{v}_{i,j} + \left(\overline{v}_{i,j} - \underline{v}_{i,j}\right)\left(1 - e^{-\alpha_{i,j} n_{i,j}^{(k)}}\right), \tag{2.13}$$

where $\underline{v}_{i,j}$ and $\overline{v}_{i,j}$ represent, respectively, the lower and upper limits of the work speed of worker i at station j, $\alpha_{i,j}$ represents the learning rate of worker i at station j, and $n_{i,j}^{(k)}$ represents the number of times worker i finishes a job at station j *prior to* working on job k. Given that the hand-off stations are random, $n_{i,j}^{(k)}$ is generally a random variable, except for $n_{I,J}^{(k)} = k - 1$ (because worker I finishes every job at the last station). Thus, the work speeds $v_{i,j}^{(k)}$ in Equation (2.13) are generally path dependent. Instead of studying the system using a large number of samples, we adopt the following *modified exponential model* to determine the work speeds:

$$v_{i,j}^{(k)} = \underline{v}_{i,j} + \left(\overline{v}_{i,j} - \underline{v}_{i,j}\right)\left(1 - e^{-\alpha_{i,j} E\left[n_{i,j}^{(k)}\right]}\right), \tag{2.14}$$

for $i = 1, \ldots, I$, $j = 1, \ldots, J$, and $k = 1, \ldots, K$.

To determine $v_{i,j}^{(k)}$, we first calculate the expected value of $n_{i,j}^{(k)}$ in Equation (2.14). Define $I_{i,j}^{(k)}$ as an indicator function that equals 1 if worker i finishes job k at station j, and equals 0 otherwise. Then, we have $n_{i,j}^{(k)} = \sum_{k'=1}^{k-1} I_{i,j}^{(k')}$. Given that the system begins with worker I working on job 1, worker $I-1$ working on job 2, and so on, worker i does not work on jobs 1 to $I-i$. Thus, we have $E\left[I_{i,j}^{(k')}\right] = 0$, for $k' = 1, \ldots, I-i$, and $E\left[I_{i,j}^{(k')}\right] = Pr\left\{\text{worker } i \text{ finishes job } k' \text{ at station } j\right\} = f_{i,j}^{(k'-I+i)}$, for $k' > I-i$. Thus, we have the following equation:

$$E\left[n_{i,j}^{(k)}\right] = \sum_{k'=1}^{k-1} E\left[I_{i,j}^{(k')}\right] = \sum_{k'=I-i+1}^{k-1} f_{i,j}^{(k'-I+i)} = \sum_{k'=1}^{k-1-I+i} f_{i,j}^{(k')}. \tag{2.15}$$

Note that $f_{i,j}^{(k')}$ is determined in Equation (2.7).

Equations (2.15) and (2.7) imply that determining $E\left[n_{i,j}^{(k)}\right]$ requires the probability distributions $\boldsymbol{\pi}^{(0)}, \ldots, \boldsymbol{\pi}^{(k-1)}$ of the hand-off station vectors. These probability distributions depend on the work speeds of the workers for the previous jobs $1, \ldots, k-1$ (see Section 2.4). We determine the work speeds for each job $k = 1, \ldots, K$ through a procedure described in Section A.7.1 of Appendix A. After obtaining the work speeds $v_{i,j}^{(k)}$, for $k = 1, \ldots, K$, we determine the expected makespan from Equation (2.3).

Figure 2.12 compares the average throughput under the modified exponential model (2.14) with that under the exponential model (2.13). We set $I = 3$, $J = 5$, $\underline{v}_{i,j} = 1$, $\overline{v}_{i,j} = 2$, and $\alpha_{i,j} = 0.01$, for $i = 1, \ldots, I$ and $j = 1, \ldots, J$. For the exponential model, we run $n = 1{,}000$ experiments to estimate the average throughput. In each experiment, for each job k, we first determine the work speeds based on Equation (2.13) and then perform a simulation using these work speeds. We observe $n_{i,j}^{(k+1)}$ in the simulation, which will be used to determine the work speeds for the next job $k+1$. The average throughput up to job k under the exponential model (dashed line in Figure 2.12) is estimated as $k/\left(\sum_{m=1}^{n} T_m^{(k)}/n\right)$, where $T_m^{(k)}$ represents the completion time of job k at the last station J in experiment m. Figure 2.12 shows that the average throughput under the modified exponential model (2.14) (solid line) almost overlaps with that under the exponential model

(2.13). This suggests that the modified exponential model (2.14) serves as a good approximation for determining the work speeds.

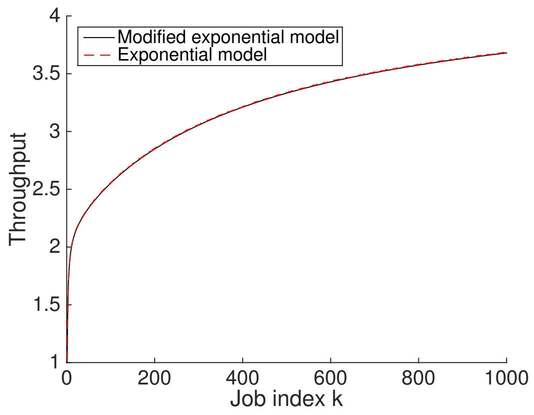


Figure 2.12: The modified exponential model (2.14) serves as a good approximation of the exponential model (2.13)

When the workers learn, is a bucket brigade with fully cross-trained workers more productive than a bucket brigade with each worker only partially cross-trained on a subset of work stations? Each worker in the former setting is more flexible but his learning effect can be diluted because of his wider work zone. In contrast, although each worker in the latter setting is constrained to a subset of stations, he can learn faster in a narrower work zone. Thus, it is non-trivial to determine which bucket brigade is more productive for a given number of jobs K. To obtain useful insights, we focus on a line with $I = 2$ workers. However, our methodology can be used to analyze a line with a general number of workers. We first analyze the bucket brigade with partially cross-trained workers, and then compare it with the bucket brigade with fully cross-trained workers.

2.7.2 A bucket brigade with partially cross-trained workers

We consider a J-station 2-worker line. Worker 1 is partially cross-trained from station 1 to station j_1, while worker 2 is partially cross-trained from station j_2 to station J. We assume $j_1 + 1 \geq j_2$ such that each station can be served by at least one worker.

Each worker carries a job and works from station to station until he hands off his job to his successor or completes the job at the end of the line. Worker 1 is *halted* if he finishes his job at station j_1 before he can hand off his job to worker 2. If worker 1 is halted, he remains idle until worker 2 takes over his job. After completing a job, worker 2 walks back upstream to take over a job from worker 1, who then initiates a new job at the beginning of the line. Worker 2 is *starved* if he walks back to station j_2 before worker 1 can hand off a job to him. If worker 2 is starved, he remains idle until he takes over a job from worker 1.

Let $\hat{H}^{(k)}$ denote the station where worker 1 is working at immediately before time $T^{(k)}$. Since worker 1 cannot work beyond station j_1, we set $\hat{H}^{(k)} = j_1 + 1$ if he finishes his job at station j_1 before time $T^{(k)}$. Thus, $\hat{H}^{(k)} \in \{1, \ldots, j_1 + 1\}$. Note that in the partially cross-trained team, worker 1 may not hand off his job to worker 2 at station $\hat{H}^{(k)}$ in the kth reset. This is because if $\hat{H}^{(k)} < j_2$, then worker 2 is starved and he will wait until worker 1 finishes a job at station $j_2 - 1$. The next hand-off occurs at station j_2. Define $\hat{\pi}_j^{(k)} = Pr\left\{\hat{H}^{(k)} = j\right\}$, for $j = 1, \ldots, j_1 + 1$. Let $\hat{\boldsymbol{\pi}}^{(k)} = \left(\hat{\pi}_1^{(k)}, \hat{\pi}_2^{(k)}, .., \hat{\pi}_{j_1+1}^{(k)}\right)$ denote the probability distribution of $\hat{H}^{(k)}$, where $\sum_{j=1}^{j_1+1} \hat{\pi}_j^{(k)} = 1$. Define the transition probability $\hat{p}_{h,h'}^{(k)} = Pr\left\{\hat{H}^{(k)} = h' | \hat{H}^{(k-1)} = h\right\}$. Let $\hat{\boldsymbol{P}}^{(k)} = \left(\hat{p}_{h,h'}^{(k)}\right)_{(j_1+1)\times(j_1+1)}$ denote the corresponding transition probability matrix.

We benchmark the partially cross-trained team against a bucket brigade with fully cross-trained workers, where the work speed of each worker on each job at each station is identical to that of the partially cross-trained team. Note that $p_{\boldsymbol{h},\boldsymbol{h}'}^{(k)}$ in Lemma 1 reduces to $p_{h,h'}^{(k)}$ when $I = 2$. We derive $\hat{p}_{h,h'}^{(k)}$ based on $p_{h,h'}^{(k)}$ in Section A.8 of Appendix A.

To derive the expected makespan of the partially cross-trained team, we first determine the inter-completion time between two successive resets. Define $\hat{z}_j^{(k)} = E\left[T^{(k)} - T^{(k-1)} | \hat{H}^{(k-1)} = j\right]$, for $j = 1, \ldots, j_1 + 1$. The following lemma determines $\hat{z}_j^{(k)}$.

Lemma 6. *The expected inter-completion time between the (k − 1)st and the kth resets, conditioned on $\hat{H}^{(k-1)} = j$ can be determined as*

$$\hat{z}_j^{(k)} = \begin{cases} \sum_{j'=j}^{j_2-1} 1/\mu_{1,j'}^{(k)} + \sum_{j'=j_2}^{J} 1/\mu_{2,j'}^{(k)}, & if\ j < j_2; \\ \sum_{j'=j}^{J} 1/\mu_{2,j'}^{(k)}, & if\ j \geq j_2; \end{cases}$$

for $j = 1, \ldots, j_1 + 1$.

Define $\hat{\boldsymbol{z}}^{(k)}$ as a (j_1+1)-dimensional vector with its jth element equals $\hat{z}_j^{(k)}$. We set $\hat{\boldsymbol{\pi}}^{(0)} = (1, 0, \ldots, 0)$. The following theorem determines the expected makespan $E\left[T^{(K)}\right]$ of the partially cross-trained team. We omit its proof because it is similar to the proof of Theorem 1.

Theorem 5. *The expected makespan of a bucket brigade with partially cross-trained workers for K jobs is*

$$E\left[T^{(K)}\right] = \sum_{k=1}^{K} \hat{\boldsymbol{\pi}}^{(0)} \hat{\boldsymbol{P}}^{(1)} \cdots \hat{\boldsymbol{P}}^{(k-1)} \hat{\boldsymbol{z}}^{(k)}. \tag{2.16}$$

2.7.3 Diversification and intensification of skills

We compare a bucket brigade with fully cross-trained workers and a bucket brigade with partially cross-trained workers. The skills of each worker in the former setting are more diversified (the workers are more flexible), but his learning effect can be diluted because of his wider work zone. In contrast, although each worker in the latter setting is constrained to a subset of stations, his learning is intensified (thus, his work speeds increase faster) in this narrower work zone. We investigate the performance of these different cross-training strategies under different initial work speeds of the workers. We first consider a case where the workers have an identical initial work speed at each station. After that we consider a line with one worker that has a smaller initial work speed than another worker at each station.

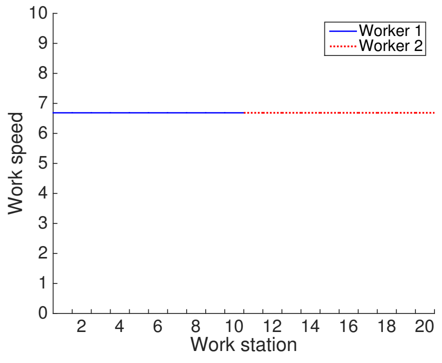
2.7.3.1 Identical workers with different cross-training strategies

We consider a twenty-station two-worker line, where the work content is evenly distributed on all the stations. We set $j_1 = j^*$ and $j_2 = J + 1 - j^*$ such that each worker is cross-trained over j^* stations. Since $j_1 + 1 \geq j_2$, we have $J/2 \leq j^* \leq J$. Specifically, for $J = 20$, we have $10 \leq j^* \leq 20$. If $j^* = 10$, the line corresponds to a *dedicated system* where workers 1 and 2 are partially cross-trained over the first and last ten stations respectively. If $11 \leq j^* \leq 19$, the work zones of the two workers overlap from station $21 - j^*$ to station j^*. If $j^* = 20$, the system becomes a bucket brigade with fully cross-trained workers. We consider three cases: $j^* = 10, 15$, and 20, which correspond to a dedicated system, a partially cross-trained system, and a fully cross-trained system respectively. We set the initial work speed of each worker at each station as 1, and his learning rate at each station in his work zone as 0.001.

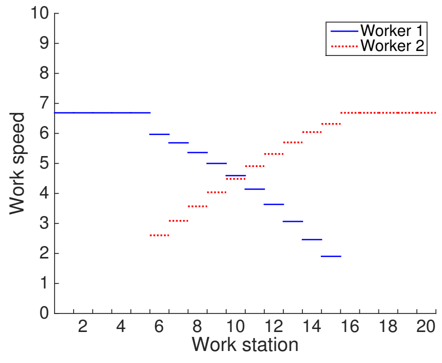
If the workers do not learn over the jobs (that is, $v_{i,j}^{(k)}$ does not change with k), then it is not surprising to see that fully cross-training the workers attains the largest average throughput. However, the results become more interesting if the workers learn over the jobs. Figures 2.13(a)-(c) show the work-speed profile of each worker on job $K = 1,000$ for $j^* = 10, 15$, and 20 respectively.

Figure 2.13(a) shows that the work speed on job $K = 1,000$ of each worker in the dedicated system ($j^* = 10$) maintains at a constant over his work zone. This is because each worker finishes every job at every station within his work zone. Given the same initial work speed and learning rate, the work speeds increase to the same value as k approaches 1,000 (see Equation (2.14)). Moreover, since $E\left[n_{i,j}^{(k)}\right]$ reaches its maximum value in the dedicated system for $K = 1,000$, the work speeds of the dedicated team are the upper bounds of the work speeds of the partially ($j^* = 15$) and the fully ($j^* = 20$) cross-trained systems.

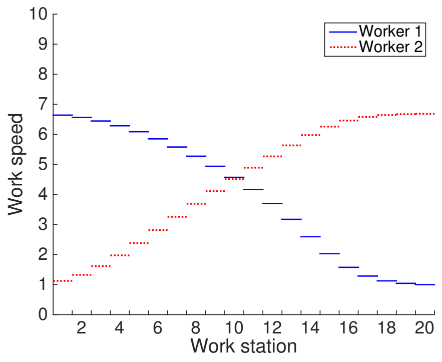
Figure 2.13(b) shows that for the partially cross-trained system ($j^* = 15$), the work speeds of worker 1 on job $K = 1,000$ maintain at the upper bound



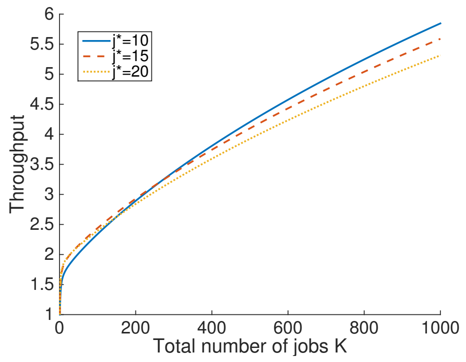
(a) $j^* = 10$



(b) $j^* = 15$



(c) $j^* = 20$



(d) The average throughput

Figure 2.13: The performance of different cross-training strategies with worker learning

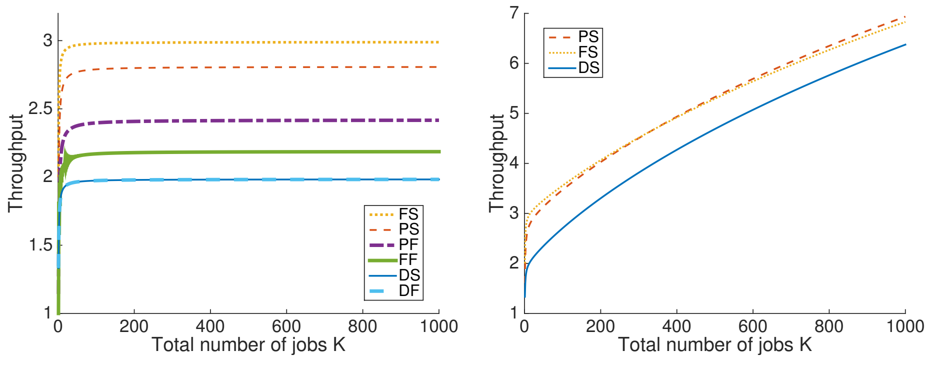
at stations 1 to 5, but decrease from station 6 to station 15 in his work zone. This is because worker 1 can be preempted by worker 2 at stations 6 to 15, and the frequency that worker 1 finishes his job at a station decreases from the upstream to the downstream in his work zone. Due to the same reason, the work speeds of worker 2 on job $K = 1,000$ increase as he moves downstream in his work zone. The work speeds of worker 2 attain the upper bound at stations 16 to 20. We observe similar work-speed profiles for the fully cross-trained system ($j^* = 20$) shown in Figure 2.13(c). Note that Figure 2.13(c) suggests that for the fully cross-trained system, the work speeds of worker 1 at stations 1 to 5 are lower than the upper bound. This is because worker 2 can take over the job from worker 1 at stations 1 to 5 in the fully cross-trained system, the frequency that worker 1 finishes his job at these stations cannot reach its maximum value for $K = 1,000$.

Figure 2.13(d) shows that for $K < 52$, the partially ($j^* = 15$) and the fully ($j^* = 20$) cross-trained systems have the largest average throughput. For $52 \leq K \leq 247$, the partially cross-trained system surpasses the fully cross-trained system to become the most productive cross-training strategy. Surprisingly, for $K > 247$, the dedicated system ($j^* = 10$) achieves the highest average throughput, followed by the partially cross-trained system, and the fully cross-trained system becomes the least productive. This is because each worker of the dedicated system focuses on his dedicated work zone, which intensifies his learning effect. This helps each worker to achieve the highest work speed (upper bound) as shown in Figure 2.13(a), and boosts the system's average throughput. In contrast, the workers' learning is not sufficiently intensified for the partially and the fully cross-trained systems (see Figures 2.13(b) and (c)). As we will see in the next section, if the workers have different initial work speeds, the dedicated system (which intensifies the workers' learning) is no longer the most productive strategy.

2.7.3.2 Non-identical workers with different cross-training strategies

We now consider a slower worker and a faster worker with initial work speeds 1 and 2, respectively, at all the stations. We define a *policy* as a combination of the cross-training strategy and the sequence of the workers. We use two capital letters to represent a policy with the first letter denotes the cross-training strategy and the second letter denotes the worker sequence. Specifically, for the first letter, let "D" represent the dedicated system, "P" represent the partially cross-trained system, and "F" represent the fully cross-trained system. For the second letter, let "S" represent the slowest-to-fastest sequence and "F" represent the fastest-to-slowest sequence based on the workers' initial work speeds. For example, the DS policy corresponds to the dedicated system with the slowest-to-fastest sequence of the workers. Likewise, the PF policy corresponds to the partially cross-trained system with the fastest-to-slowest worker sequence.

Figure 2.14(a) shows the average throughput of the six policies: DS, DF, PS, PF, FS, and FF without worker learning. The relative performance



(a) Without learning (b) With learning

Figure 2.14: The performance of different policies without and with worker learning

of the policies is as follows: $\rho_{FS} > \rho_{PS} > \rho_{PF} > \rho_{FF} > \rho_{DS} = \rho_{DF}$. Without learning, the FS policy is the most productive because the fully cross-trained system is more flexible and the slowest-to-fastest sequence has less blocking. The DS and the DF policies have the lowest average throughput because the slower worker becomes the bottleneck of the line. Under the DS policy, the faster worker at the downstream is often starved, whereas under the DF policy, the faster worker at the upstream is often halted. Interestingly, under the fastest-to-slowest sequence, the partially cross-trained system outperforms the fully cross-trained system ($\rho_{PF} > \rho_{FF}$). This is because in the fully cross-trained system, the faster worker upstream under the FF policy is constantly preempted and is subsequently blocked by the slower worker. Note that for the fully and the partially cross-trained systems, the slowest-to-fastest sequence outperforms the fastest-to-slowest sequence ($\rho_{FS} > \rho_{FF}$ and $\rho_{PS} > \rho_{PF}$) because the former sequence has less blocking. This observation still holds if the workers learn over the jobs. Thus, we focus on the slowest-to-fastest sequence when we compare the different cross-training strategies with worker learning in the following paragraphs.

Figure 2.14(b) shows the average throughput of different cross-training strategies with the slowest-to-fastest sequence when the workers learn. In contrast to Section 2.7.3.1, the dedicated system becomes the least productive for a line with a slower worker and a faster worker. This is again due to the fact that the slower worker is the bottleneck of the line. We also observe that when the number of jobs is small ($K < 431$), the fully cross-trained system has the largest average throughput, which resembles the no-learning case (see Figure 2.14(a)). In this situation, the line is more productive if the workers' skills are diversified (if the workers are fully cross-trained). However, for a large number of jobs ($K \geq 431$), the PS policy outperforms the FS policy. In this situation, the effect of the intensification of learning in the partially cross-trained system becomes dominant. Figures 2.14(a) and (b) suggest that for a line with a worker that is slower than his colleague, the productivity of the line is limited by the slower worker. One way to overcome this issue is to diversify the workers' skills by cross-training such that the faster worker can help the slower worker (see Figure 2.14(a)). However, if the workers learn over the jobs, then the intensification of learning can also boost the productivity (see Figure 2.14(b)). *Thus, an appropriate mix of the diversification of the workers' skills and the intensification of their learning becomes crucial for maximizing the productivity of the line.*

Comparing Figures 2.13(d) and 2.14(b), the relative performance of the different cross-training strategies changes with the ratio $\underline{v}_2/\underline{v}_1$, which is the relative initial work speed of worker 2 to worker 1. If $\underline{v}_2/\underline{v}_1 = 1$, the dedicated system is the most productive for a large number of jobs K (see Figure 2.13(d)). This suggests that if the workers are identical, then the effect of intensified learning dominates. In contrast, if $\underline{v}_2/\underline{v}_1 = 2$, then the partially cross-trained system becomes the most productive for a large number of jobs K (see Figure 2.14(b)). As the initial work speeds become more different, diversifying the workers' skills makes the line more productive. This is because by cross-training each worker over a wider zone, the negative effect of the bottleneck worker (the slowest worker) can be mitigated.

Figure 2.15 shows the best cross-training strategies (represented by the best j^* on the x-axis) for different values of $\underline{v}_2/\underline{v}_1$ (shown on the y-axis). If $\underline{v}_2/\underline{v}_1$ equals 1, the dedicated system is the most productive because the intensification of learning boosts the line's productivity. On the other hand, if $\underline{v}_2$ is significantly larger than $\underline{v}_1$, the fully cross-trained system ($j^* = 20$)

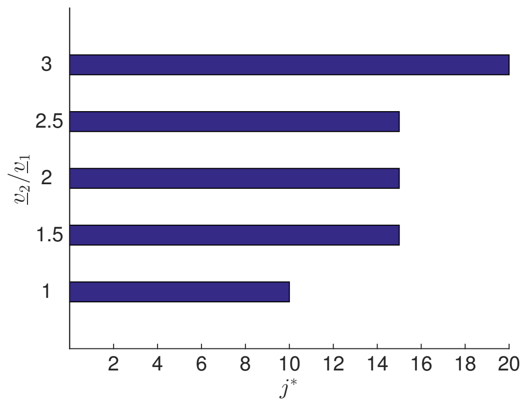


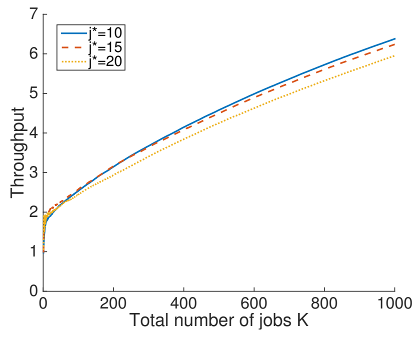
Figure 2.15: The best cross-training strategies for different relative initial work speeds

becomes the most productive. This is because diversifying the skills of the workers offers more flexibility to the line, and mitigates the negative effect of the bottleneck (slowest) worker. Lastly, if $\underline{v}_2$ is moderately larger than $\underline{v}_1$, then the partially cross-trained system ($j^* = 15$) performs the best. This is because by moderately (partially) cross-training the workers, the line achieves a balance between the diversification of the workers' skills and the intensification of worker learning. This delicate balance maximizes the productivity of the line in this situation.

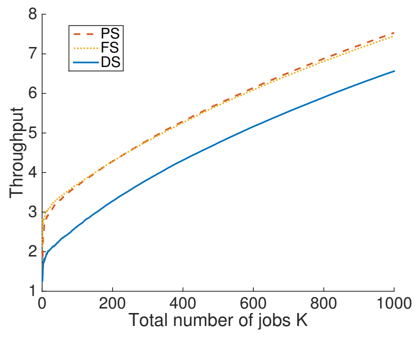
2.8 Robustness of the results

In this section, we examine the robustness of the findings in Section 2.7.3 for other settings using simulations. Figure 2.16 shows the relative performance of the different cross-training strategies for a line with uniformly-distributed service times. We use the same parameter values in Figures 2.13(d) and 2.14(b) for Figures 2.16(a) and (b) respectively. Figure 2.16 shows that the main findings in Section 2.7.3 still hold for uniformly-distributed service times. We also observe similar results for normally-distributed service times.

Likewise, Figure 2.17 shows the relative performance of the different cross-training strategies for a model in which the workers walk with finite speeds



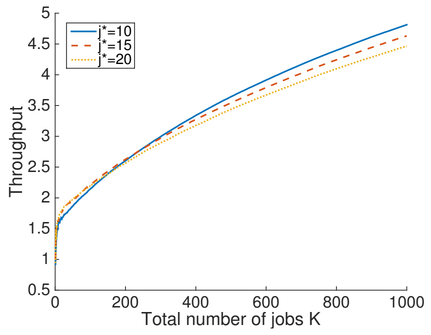
(a) Two identical workers



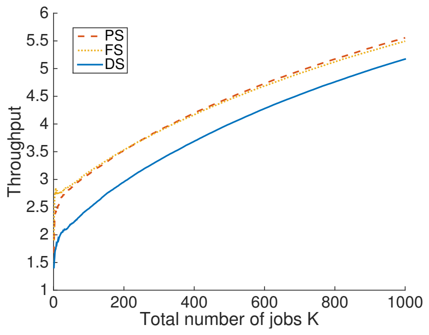
(b) Two non-identical workers

Figure 2.16: The performance of different cross-training strategies with uniformly-distributed service times

from one station to another station. We set the forward and the backward walk speeds as 20 and 40, respectively, for each worker. We use the same parameter values in Figures 2.13(d) and 2.14(b) for Figures 2.17(a) and (b) respectively. Compared to Section 2.7.3, Figure 2.17 shows that the relative performance of the different cross-training strategies remain the same when the workers walk with finite speeds.



(a) Two identical workers



(b) Two non-identical workers

Figure 2.17: The performance of different cross-training strategies with finite walk speeds

2.9 Conclusion

The literature of bucket brigades with stochastic service times is very limited. To the best of our understanding, Bartholdi III et al. (2001) is the only paper that analytically studies bucket brigades on discrete work stations with stochastic service times. The authors assume that the work speed of each worker at each station depends only on the worker. In this chapter, we extend the work of Bartholdi III et al. (2001) by assuming that the work speed of each worker at each station on a job depends not only on the worker, but also on the station and the job. Thus, the workers may not dominate each other at all the stations, and their work speeds may change with the jobs. We consider a bucket brigade line with J stations and I workers. We assume the time duration for each worker to finish a job at each station is exponentially distributed with a rate that depends on the station's expected work content and the work speed of the worker.

By observing the Markov property of the hand-off stations and analyzing the transition from one hand-off station vector to the next, we are able to derive the system's average throughput (Theorem 1) and the CV of the inter-completion time (Lemma 2). We prove that *if the work speeds of the workers are independent of the jobs, then the probability distribution of the hand-off station vector will converge to a unique stationary distribution as the number of jobs approaches infinity* (Theorem 2). Furthermore, the average throughput and the CV of the inter-completion time converge to a constant that depends on the stationary distribution (Theorems 2 and 3).

We first study a case where each worker's work speeds depend only on the worker. For a two-worker system in which one worker has a larger speed than the other at all the stations, we find that the probability distribution of the hand-off station is analogous to the behavior of the deterministic model. If the workers are sequenced from slowest to fastest, the probability distribution has a peak in the middle of the line. It is interesting to note that this result is consistent with that of the deterministic model in which the hand-offs converge to a fixed location (Bartholdi III, Eisenstein, 1996b). However, if the workers are sequenced from fastest to slowest, the probability distribution has two peaks at the two ends of the line. This result is analogous to that of the deterministic model studied by Bartholdi III et al. (1999) in which the hand-offs converge to a 2-cycle.

We further compare the throughput of our stochastic model with the deterministic model with discrete work stations. Under the slowest-to-fastest sequence, the deterministic system is more productive than the stochastic system. This suggests that the stochastic service times cause throughput loss for a bucket brigade line with discrete work stations under the slowest-to-fastest sequence. However, under the reverse sequence, the stochastic system may outperform the deterministic system. For both worker sequences, the performance difference between the stochastic and the deterministic systems gets closer as the number of stations J increases, which is consistent with the finding of Bartholdi III et al. (2001).

However, if the number of stations is small, the throughput difference between the stochastic and the deterministic systems can be *quite significant* (the gap is up to 47%). Furthermore, for a stochastic system, the slowest-to-fastest sequence is more productive than the reverse sequence (see Corollary 1 and Figure 2.5). It is worth noting that this result may not hold for a deterministic system (see Figure 6(a) of Lim, Yang (2009)). Thus, it is worthwhile and important to study the stochastic bucket brigade model especially for a line with a small number of stations.

To maximize the asymptotic throughput, the manager can optimize both the worker sequence and the work-content distribution over the stations. We demonstrate that the slowest-to-fastest sequence always outperforms the reverse sequence (Figure 2.6(a)). Furthermore, under the slowest-to-fastest sequence, the asymptotic throughput is maximized if the expected work content of the stations increases in the direction of the production flow. In contrast, under the fastest-to-slowest sequence, the asymptotic throughput is maximized if the work content decreases from upstream to downstream. These results also hold when we vary the number of stations, the number of workers, or the difference in the work speeds of the workers.

In terms of minimizing the asymptotic CV of the inter-completion time, we also find that the slowest-to-fastest sequence always outperforms the reverse sequence. For the fastest-to-slowest sequence, the asymptotic CV is the minimum when the expected work content increases from upstream to downstream. However, for the slowest-to-fastest sequence, the asymptotic CV reaches the minimum when the expected work content decreases in the direction of the production flow. The above results are robust when we vary the number of stations, the number of workers, or the difference in the work speeds of the workers.

From the above results, we obtain the following important managerial insights. *Either maximizing the asymptotic throughput or minimizing the asymptotic CV of the inter-completion time, the slowest-to-fastest sequence always outperforms the reverse sequence for the stochastic bucket brigade. Furthermore, to maximize the asymptotic throughput, more work content should be assigned to the stations near the faster workers (Figure 2.6(b)). In contrast, to minimize the asymptotic CV of the inter-completion time, more work content should be assigned to the stations near the slower workers (Figure 2.9(b)).*

We then analyze another case where the work speeds depend on the workers and the stations. Given that the workers may not dominate each other along the entire line, we define the average work speed of each worker as a weighted average of his work speeds at all the stations (Equation (2.6)). We also derive the asymptotic expected blocked time of each worker (Equation (2.9)). The asymptotic throughput of the stochastic bucket brigade can be expressed as a function of the average work speeds and the asymptotic expected blocked times of the workers, and can be interpreted as the sum of the effective production rates of all the workers (Theorem 4).

Finally, we study the general case where the work speed $v_{i,j}^{(k)}$ of worker i at station j on job k depends on the worker, the station, and the job. We assume each worker learns over the jobs such that the more often he visits a station, the faster he becomes at that station. We propose a modified exponential model and a procedure to determine the work speeds. To boost the line's productivity, a manager can diversify each worker's skills by cross-training them over more work stations. Although each worker is more flexible under this strategy, his learning effect can be diluted because of his wider work zone. In contrast, a worker that is constrained to work at a smaller set of stations, albeit less flexible, can intensify his learning. Thus, when the workers learn, it is non-trivial to determine whether a bucket brigade with fully cross-trained workers is more productive than a bucket brigade with each worker only partially cross-trained on a subset of stations. To obtain useful insights, we focus on a line with $J = 20$ stations and $I = 2$ workers. We find that if $\underline{v}_2/\underline{v}_1$ (the relative initial work speed of worker 2 to worker 1) equals 1, the dedicated system is the most productive because of intensified learning (see Figure 2.15). On the other hand, if $\underline{v}_2$ is significantly larger than $\underline{v}_1$, the fully cross-trained system becomes the most productive. This is because diversifying the skills of the workers offers more flexibility to the line, and mitigates the negative effect of the bottleneck (slowest) worker. Lastly, if $\underline{v}_2$ is moderately larger than $\underline{v}_1$, then the partially cross-trained system performs the best. This is because by moderately (partially) cross-training the workers, the line achieves a balance between the diversification of the workers' skills and the intensification of worker learning. This delicate balance maximizes the line's productivity in this situation.

Our main findings from the model with exponential service times continue to hold for uniformly- or normally-distributed service times. Furthermore, we observe similar results when the workers walk with finite speeds.

Chapter 3

Stochastic Bucket Brigades with Non-preemptible Work Content

3.1 Introduction

There are different designs of a bucket brigade line catering for different assembly or production settings, such as a serial bucket brigade (Bartholdi III, Eisenstein, 1996b,a, Bartholdi III et al., 2001), a bucket brigade on in-tree assembly networks (Bartholdi III et al., 2006), a cellular bucket brigade (Lim, 2011) and so on. Even if the design of a serial bucket brigade is chosen for the assembly or production, the management needs to consider whether the work content is *preemtpbile*. If the work content is preemptible, the work that is processed by one worker on a station can be interrupted or preempted and continues to be processed by another worker; If the work content is non-preemptible, the work on each station can not be preempted and has to be processed by the same worker.

In this chapter, we continue to study a J-station, I-worker bucket brigade line. However, we assume that the work content of any station is *non-preemptible*. As such, when one worker walks back to his predecessor, he can not take over the job immediately, but has to wait for his predecessor to finish her job at the current station, and then he continues to process the job from the next station. Similar to the assumptions in Chapter 2, we assume that the time duration for each worker to process a job at each station follows an independent exponential distribution with a rate that depends on the stations work content and the workers work speed. We also assume that the workers work speed may depend on the worker, the station, and the job. In a non-preemptible bucket brigade, a *hand-off* occurs when a worker finishes his job at a station and *at the same time* his successor is waiting to take over the job. This differs from the preemptible system introduced in Chapter 2.

3.2 Notation and assumptions

We consider a bucket brigade assembly line with work stations sequenced as $j = 1, \ldots, J$. There are I workers sequenced as $i = 1, \ldots, I$ in the direction of the production flow. We call workers $i-1$ and $i+1$ the *predecessor* and the *successor*, respectively, of worker i. There are K jobs to be processed by the assembly line. Worker i is blocked in front of station j if his job is ready to enter station j but his successor is still working at station j. When worker i is working at any station, we assume the station's work content is *non-preemptible* such that worker $i+1$ needs to *wait* for the former to finish his job at a station before he can take over the former's job.

We assume the workers spend negligible time when they move from one station to another. Let $Z_{i,j}^{(k)}$ denote the time duration for worker i to serve (or process) job k at station j. We assume $Z_{i,j}^{(k)}$ follows an exponential distribution with rate $\mu_{i,j}^{(k)}$, for $i = 1, \ldots, I$, $j = 1, \ldots, J$, and $k = 1, \ldots, K$. We define the rate $\mu_{i,j}^{(k)} = \left(s_j/v_{i,j}^{(k)}\right)^{-1}$, where s_j represents the expected work content of station j and $v_{i,j}^{(k)}$ represents the work speed of worker i at station j on job k. We normalize the total expected work content such that $\sum_{j=1}^{J} s_j = 1$. We assume that all the workers are *fully cross-trained* such that $v_{i,j}^{(k)} > 0$, for all i, j, and k.

Let $T^{(k)}$ denote the time point when worker I completes job k at the last station J. Let $H_i^{(k)}$ denote the station where worker i is working at immediately before $T^{(k)}$, for $i = 1, \dots, I-1$. If worker i is blocked in front of station j immediately before $T^{(k)}$, we set $H_i^{(k)} = j$. If worker i is waiting for his predecessor to finish his job immediately before $T^{(k)}$, we set $H_i^{(k)} = 0$.

Define $\boldsymbol{H}^{(k)} = \left(H_1^{(k)}, \dots, H_{I-1}^{(k)}\right)$ as the kth *reset vector*. We set $T^{(0)} = 0$ and $\boldsymbol{H}^{(0)} = (1, \dots, 1)$. Define $\mathcal{H}(I, J)$ as a set of all possible reset vectors for a non-preemptible bucket brigade with J stations and I workers.

Lemma 7. *An $(I-1)$-dimensional vector $\boldsymbol{h} = (h_1, h_2, \dots, h_{I-1})$ belongs to $\mathcal{H}(I, J)$ if and only if it satisfies:*

1. *$0 \le h_i \le J$ for $i = 1, 2, \dots, I-1$. $h_1 \ne 0$.*
2. *If $h_{i-1} \ne 0$, then $h_i \ge h_{i-1}$, or $h_i = 0$, for $i = 2, \dots, I-1$.*
3. *If $h_i \ne 0$, and $h_{i+1} = h_{i+2} \dots = h_{i+l} = 0$, then $h_{i+l+1} > h_i$ or $h_{i+l+1} = 0$, for $i = 1, \dots, I-3$, and $1 \le l \le I-2-i$.*

3.3 Performance measures and asymptotic behavior

In this section, we calculate the throughput of the system by calculating the expected time for completing K jobs, $E[T^{(k)}]$. Let $Y(k) = T^{(k)} - T^{(k-1)}$ denote the inter-completion time between job $k-1$ and job k. Lemma 8 shows the method of calculating $Y(k)$ by tracking the working time and waiting time of worker I.

Lemma 8. *Suppose $\boldsymbol{H}^{(k-1)} = \boldsymbol{h}$, $Y(k)$ can be calculated based on three scenarios:*

1. *If $1 \le \boldsymbol{h}_{I-1} \le J-1$, $Y(k) = Z_{I-1,h_{I-1}}^{(k)} + \sum_{j=h_{I-1}+1}^{J} Z_{I,j}^{(k)}$.*
2. *If $h_{I-1} = J$, $Y(k) = Z_{I,J}^{(k)}$.*

3. *If $h_{I-i} > 0$, and $h_{I-i+1} = h_{I-i+2} = \ldots h_{I-1} = 0$, for $2 \leq i \leq I-1$, $Y(k) = Z^{(k)}_{I-i,h_{I-i}} + \sum_{j=h_{I-i}+1}^{J} Z^{(k)}_{I,j}$.*

Lemma 8 shows that the inter-completion time can be determined by the reset vector. Thus, it is essential to study the transitions of the reset vector between two consecutive resets to calculate the inter-completion time. The transition behavior the reset vector can be determined by observing the movements of the I workers between the $(k-1)$st and kth resets.

3.3.1 State of the system

Between the $(k-1)$st and kth resets, let $X(i)$ denote the station where worker i is located, for $i = 1, \ldots, I$. If worker i is blocked in front of station j, we set $X(i) = j$. If worker i is waiting for his predecessor, we set $X(i) = 0$. We set $X(I) = J+1$ when worker I finishes his job at station J. Define a *state* of the system as $\boldsymbol{X} = (X(1), X(2), \ldots, X(I))$. We call a state with $X(I) \leq J$ a *transient state*, and a state with $X(I) = J+1$ an *absorbing state*. Define $\mathcal{X}(I,J)$ as a set of all possible states between any two resets. Lemma 9 shows the relation of the cardinality of $\mathcal{X}(I,J)$ and that of $\mathcal{H}(I,J)$.

Lemma 9. *The cardinality of $\mathcal{X}$ can be obtained by:*

$$|\mathcal{X}(I,J)| = |\mathcal{H}(I,J)| + |\mathcal{H}(I+1,J)|$$

Between the $(k-1)$st and the kth resets, instead of keeping track of the system's state continuously, we only need to consider the *epochs* when a worker finishes his job at a station. Given that the service times are exponentially distributed, the probability that more than one worker finish their jobs at their respective stations at the same time is 0. For any transient state $\boldsymbol{x}$, suppose there are $N(\boldsymbol{x})$ workers that are working. Let $l_1, \ldots, l_{N(\boldsymbol{x})}$ denote these $N(\boldsymbol{x})$ workers from upstream to downstream.

Lemma 10. *For a transient state $\boldsymbol{x}$, worker i $(1 \leq i \leq I-1)$ is working if and only if it satisfies the following two conditions,*

1. $x_i \neq 0$.

2. $x_{i+1} = 0$, *or* $x_{i+1} > x_i$.

Between the $(k-1)$st and the kth resets, let k_n denote the job that worker l_n is working on. The following lemma determines k_n.

Lemma 11. *Between the* $(k-1)$*st and the* k*th resets, for a transient state* $\boldsymbol{x}$*, the job that worker* l_n *is working on can be determined by:*

$$k_n = k + \sum_{i=l_n}^{I} \mathcal{I}\{x(i) \neq 0\}$$

Among the $N(\boldsymbol{x})$ workers that are working, one of them, say worker l_n, will finish his job in the next epoch. Let $\boldsymbol{x}_n$ denote the new state immediately after worker l_n finishes his job, we have:

Lemma 12. $\boldsymbol{x}_n$ *is determined according to four senarios:*

1. *If* $l_n = I$*, we have* $\boldsymbol{x}_n = (x(1), x(2), \ldots, x(I-1), x(I)+1)$.

2. *If* $1 \leq l_n \leq I-1$*, and* $x(l_n+1) > x(l_n)$*, we have* $x_n(l_n) = x(l_n)+1$*, and* $x_n(i) = x(i)$*, for* $i \neq l_n,\ 1 \leq i \leq I$.

3. *If* $2 \leq l_n \leq I-1$*,* $x(l_n+1) = x(l_n+2) = \ldots = x(l_n+m) = 0$*, and* $x(l_n+m+1) \neq 0$*, we have* $x_n(l_n) = 0$*,* $x_n(l_n+m) = x(l_n)+1$*, and* $x_n(i) = x(i)$*, for* $i \neq l_n,\ i \neq l_n+m,\ 1 \leq i \leq I$.

4. *If* $l_n = 1$*,* $x(2) = x(3) = \ldots = x(m) = 0$*, and* $x(m+1) \neq 0$*, we have* $x_n(1) = \ldots = x_n(m-1) = 1$*,* $x_n(m) = x(1)+1$*, and* $x_n(i) = x(i)$*, for* $m+1 \leq i \leq I$.

Between the $(k-1)$st and the kth resets, the system progresses from state $\boldsymbol{x}$ to another state $\boldsymbol{x'}$. Let $q^{(k)}_{\boldsymbol{x},\boldsymbol{x'}}$ denote a one-step transition probability from $\boldsymbol{x}$ to $\boldsymbol{x'}$. For any transient state $\boldsymbol{x}$, we have

$$q^{(k)}_{\boldsymbol{x},\boldsymbol{x'}} = \begin{cases} \mu^{(k_n)}_{l_n,x(l_n)} / \sum_{m=1}^{N(\boldsymbol{x})} \mu^{(k_m)}_{l_m,x(l_m)}, & if\ \boldsymbol{x'} = \boldsymbol{x}_n, n = 1, \ldots, N(\boldsymbol{x}), \\ 0, & otherwise. \end{cases}$$

For any absorbing state $\boldsymbol{x}$, we set

$$q^{(k)}_{\boldsymbol{x},\boldsymbol{x'}} = \begin{cases} 1, & if\ \boldsymbol{x'} = \boldsymbol{x}, \\ 0, & otherwise, \end{cases}$$

Let $\boldsymbol{Q}^{(k)} = \left(q^{(k)}_{\boldsymbol{x},\boldsymbol{x}'}\right)_{|\mathcal{X}|\times|\mathcal{X}|}$ denote the corresponding one-step transition probability matrix of the states.

3.3.2 Transition probability matrix of the reset vectors

Starting from any state $\boldsymbol{x}$, the system will take at most $I(J-1)+1$ epochs to reach an absorbing state. Let $\boldsymbol{R}^{(k)} = \left(\boldsymbol{Q}^{(k)}\right)^{I(J-1)+1}$, and $r^{(k)}_{\boldsymbol{x},\boldsymbol{x}'}$ denote the $\boldsymbol{x}$-$\boldsymbol{x'}$ entry of $\boldsymbol{R}^{(k)}$. Suppose the $(k-1)$st reset vector is $\boldsymbol{h}$. Let $\boldsymbol{x_h}$ denote the state of the system immediately after the $(k-1)$st reset. The following lemma determines $\boldsymbol{x_h}$.

Lemma 13. *$\boldsymbol{x_h}$ is determined as below:*

1. *If $h_{I-1} < J$, $\boldsymbol{x}(\boldsymbol{h}) = (\boldsymbol{h}, 0)$.*
2. *If $h_{I-m} = \ldots = h_{I-1} = J$, $h_{I-m-1} \neq J$ for $1 \leq m \leq I-2$, then $x_{\boldsymbol{h}}(I) = J$, $x_{\boldsymbol{h}}(I-m) = 0$, and $x_{\boldsymbol{h}}(i) = h_i$ for $i \neq I-m$, $1 \leq i \leq I-1$.*
3. *If $h_1 = \ldots = h_{I-1} = J$, $\boldsymbol{x_h} = (1, J, \ldots, J)$.*

As the system transitions from the reset vector $\boldsymbol{h}$ immediately after the $(k-1)$st reset to the reset vector $\boldsymbol{h'}$ at the kth reset, the system's state progresses from $\boldsymbol{x_h}$ to $(\boldsymbol{h}', J+1)$. Let $\boldsymbol{P}^{(k)} = \left(p^{(k)}_{\boldsymbol{h},\boldsymbol{h}'}\right)_{|\mathcal{H}|\times|\mathcal{H}|}$ denote the transition probability matrix of the reset vectors. We have the following lemma.

Lemma 14. *The probability of $\boldsymbol{h}'$ being the kth reset vector, conditioned on $\boldsymbol{h}$ being the $(k-1)$st reset vector is*

$$p^{(k)}_{\boldsymbol{h},\boldsymbol{h}'} = r^{(k)}_{\boldsymbol{x_h},(\boldsymbol{h}',J+1)}.$$

3.3.3 The throughput of the line

Define $\boldsymbol{z}^{(k)}$ as an $|\mathcal{H}|$-dimensional column vector with its nth component equals $E\left[Y(k)|\boldsymbol{H}^{(k-1)}=\boldsymbol{h}^n\right]$, for $n=1,\ldots,|\mathcal{H}|$.

$$E\left[Y(k)\right]=\sum_{n=1}^{|\mathcal{H}|}E\left[Y(k)|\boldsymbol{H}^{(k-1)}=\boldsymbol{h}^n\right]\pi_n^{(k-1)}=\boldsymbol{\pi}^{(k-1)}\boldsymbol{z}^{(k)}.$$

Define the *average throughput* of a bucket brigade assembly line with K jobs as $\rho(K)=K/E\left[T^{(K)}\right]$. The following theorem determines the expected makespan $E\left[T^{(K)}\right]$, and $\rho(K)$.

Theorem 6. *The expected makespan of a bucket brigade assembly line with K jobs is*

$$E\left[T^{(K)}\right]=\sum_{k=1}^{K}\boldsymbol{\pi}^{(0)}\boldsymbol{P}^{(1)}\cdots\boldsymbol{P}^{(k-1)}\boldsymbol{z}^{(k)}.$$

We have

$$\rho(K)=K\left/E\left[T^{(K)}\right]=K\right/\sum_{k=1}^{K}\boldsymbol{\pi}^{(0)}\boldsymbol{P}^{(1)}\cdots\boldsymbol{P}^{(k-1)}\boldsymbol{z}^{(k)}\,. \tag{3.1}$$

The proof of Theorem 6 is similar to Theorem 1. From Theorem 6 we see that given $\boldsymbol{\pi}^{(0)}$ and $\boldsymbol{z}^{(k)}$, we can determine the average throughput of the line using the transition probability matrices $\boldsymbol{P}^{(k)}$, for $k=1,\ldots,K-1$.

Let $\boldsymbol{e}$ denote an $|\mathcal{H}|$-dimensional unit column vector.

Theorem 7. *If the workers' service rates are independent of the jobs, then $\lim_{k\to\infty}\boldsymbol{\pi}^{(k)}=\boldsymbol{\pi}$, where $\boldsymbol{\pi}$ is a unique stationary distribution that satisfies the equations $\boldsymbol{\pi}\boldsymbol{P}=\boldsymbol{\pi}$ and $\boldsymbol{\pi}\boldsymbol{e}=1$. Furthermore, the asymptotic throughput $\rho_\infty=\lim_{K\to\infty}\rho(K)=1/(\boldsymbol{\pi}\boldsymbol{z})$, where $\boldsymbol{\pi}\boldsymbol{z}$ represents the asymptotic expected inter-completion time.*

3.4 Numerical results

We set $I = 4, J = 8, s_1 = s_2 = \ldots = s_8 = 1/8, K = 1000$, and $v_1 = 1, v_2 = 2, v_3 = 3, v_4 = 4$. The bold solid line in Figure 3.1 represents the average throughput $\rho(k)$ of a non-preemptible bucket brigade assembly line up to job k. The dashed line and the thin solid line represent two sample paths of the *actual throughput* $k/T^{(k)}$ based on simulations.

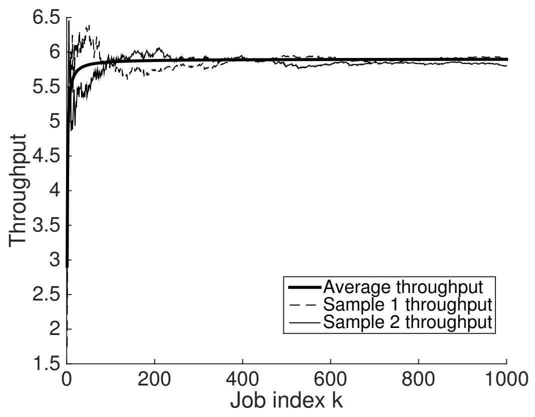
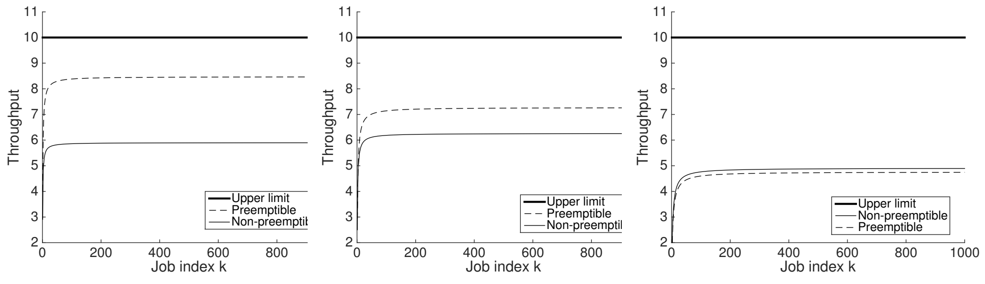


Figure 3.1: The throughput of a non-preemptible bucket brigade line with four workers.

Figure 3.2 compares the average throughput of a non-preemptible system to that of a preemptible system and their upper limit. We set $v_1+v_2+v_3+v_4 = 10$. Figure 3.2(a) depicts the case where $v_1 = 1, v_2 = 2, v_3 = 3, v_4 = 4$. That is, the workers are sequenced from slowest to fastest. We see that there is a significant throughput gap between the non-preemptible line and the preemptible line. This is because in a non-preemptible line, the faster worker at the downstream has to wait for the slower worker at upstream when the downstream worker finishes his work and walks back. It hurts the productivity of the line. Figure 3.2(b) depicts the case where there are four identical workers, each with work speed 2.5 at all the stations. The throughput gap between the two systems becomes smaller. Figure 3.2(c) depicts the case where $v_1 = 4, v_2 = 3, v_3 = 2, v_4 = 1$. That is, the workers are sequenced from fastest to slowest. We see that the non-preemptible line outperforms the preemptiblle line! This is because in a non-preemptible line, the faster worker upstream is not preempted by the slower worker downstream. The work content continues to be processed by the faster worker. However, in a preemptible line, the work content is preempted by the slower worker downstream. This may cause more blocking, thus hurting the throughput. From above, we see that there are significant differences in managing the non-preemptible and the preemptible lines. It is essential for the management to consider about the actual setting to enhance the performance of the system.



(a) Slowest-to-fastest sequence (b) Identical workers (c) Fastest-to-slowest sequence

Figure 3.2: The throughputs of a non-preemptible and a preemptible lines and their upper limit.

3.5 Conclusion

In this chapter, we study a stochastic bucket brigade line with non-preemptible work content. The main methodologies in Chapter 2 can be carried over to the analysis of this system. However, the status of a worker can be not only working or being blocked, but can also be waiting for his predecessor to finish his job at a station. As such, it complicates the analysis in a few perspectives. (1) The waiting workers need to be well defined. (2) The inter-completion times and the relation between the reset vectors and the states of the system have to be re-analyzed. (3) The transitions of the states have to be analyzed case by case. In Section 3.2 and Section 3.3, we properly denote the waiting workers, re-analyze the state of the system, and the transition probability matrix of the reset vectors. Finally, we derive the average throughput for a bucket brigade with non-preemptible work content.

In Section 3.4, we first compare the average throughput derived from Theorem 6 to the sample paths based on simulations. We see that the throughputs converge as K increases. Then we compare the throughput difference between a non-preemptible line and a preemptible line. If the workers are sequenced slowest-to-fastest, the preemptible line dominates the non-preemptible line. However, if the workers are sequenced fastest-to-slowest, the non-preemptible line can dominate the preemptible line. As such, the management needs to consider about the actual setting to enhance the performance of the system.

Chapter 4

Distributive Decision Rule in Resource Allocation Problems

4.1 Introduction

Resource allocation problems are a traditional area of Operations Research (Karp et al., 1990, Reeves, Sweigart, 1982, Riedel, 1999), in which the planner has to allocate a finite amount of resources to meet sources of demand with different requirements. Often, there is uncertainty in the demand, especially in time. The goal of the planner is to maximize a function of revenue arising from a successful allocation (*e.g.* clickthroughs in the advertising setting), or to minimize the cost of not matching (*e.g.* waiting time before allocation). Resource allocation problems have a variety of applications such as in online advertising (Jaillet, Lu, 2011), freight allocation (Spivey, Powell, 2004), and appointment scheduling (Gupta, Wang, 2008).

There is renewed interest of late in service management problems in the area of shared economy, platform services and distributed networks, such as cloud computing, online ride-hailing or ride-sharing (Tang et al., 2020, Shu et al., 2013) and last-mile delivery (Qi et al., 2018). Due to the short response time and the high variability in the state and the demand, these settings require adaptive policies that respond to the state and the uncertainty in the demand. Another key feature is the ability of the decision-maker to simultaneously control the matching between supply and demand, and the allocation of supply amongst different supply pools. For example, this occurs in the ride-hailing industry where the decision-maker has incentive to re-position vehicles (Ghosh et al., 2017).

In this paper, we are interested in studying resource allocation problems under such a context. Exhaustively, we list down the features that we are interested in:

1. Replenishable supply: When allocated, the supply is occupied for a *random* period of time, after which it is freed up. For example, in call centers, each receptionist is occupied for a random length of service, until the call ends and she becomes available again.

2. Control over allocation and supply: To increase the flexibility of the setting considered, we assume the decision-maker can control how the supply is distributed over supply sources, while simultaneously controlling the allocation of demand to supply. For example, in vehicle re-positioning in the ride-hailing context, the decision-maker can bring excess supply in one area to another, under limitations of costs and delay in re-positioning.

3. Disjoint demand and supply sources: We consider situations where demand and supply are partitioned into disjoint, but homogeneous sources. The demand has known arrival distribution, which can be estimated or inferred from data. However, demand can be non-time-homogeneous and can be highly variable (as Serna et al., 2017, discusses to be common in such contexts).

4. Demand need not be immediately fulfilled: Demand that is not immediately served, will remain in the system and can incur some known costs. This is broad enough to model lost demand, as well. For example, in a ride-hailing platform, the customer may not be served immediately, but will wait for a match with an available driver. If

after some time, the customer decides to leave, the platform incurs lost sales.

5. Known allocation value: Critical in the context that we are considering, the value of the allocation is known, be it the revenue from each allocation or the costs of failure to match. For example, in platform services, monetization occurs at the point where a match is secured; and in cloud computing, known cost is incurred in terms of delay, but where all demand must be served at some point.

The above features do permit a large class of problems, especially those pertaining to service provisions, such as online platforms of cloud computing clusters, ride-hailing and call centers. We, however, exclude settings where the reward or cost of the matching is unknown, such as exploration-exploitation in online advertising, those in which the resources are not renewable and are exhausted, such as advanced booking or scheduling, and those where the demand and/or supply are highly heterogeneous, such as personalized services. Our model is also not suitable for platforms that behave like a market wherein the allocation is decided by demand and supply forces, as opposed to the central planner, *e.g.* in the case of Airbnb.

These features describe a *doubly stochastic* nature of demand and the replenishment of the supply, which induces complex dynamics. Moreover, the state of the system is highly variable, and is potentially faced with large uncertainties in the demand and service time. The culmination is that this setting dictates the need for allocation and supply distribution policies that are adaptive and state-dependent. The challenge is to solve for such policies tractably.

4.1.1 Key approaches in literature

There are three streams of literature that address the dynamics of the problems under similar settings, namely, queuing, stochastic programming and learning.

Queuing network: Posing the problem as a queuing network is a common and natural approach to address such problems – the queue and server system can model the matching of the demand with the supply; and arrivals and service time distributions can be modelled naturally in this setting. Arguably, this is the most commonly seen approach. For example, Armony, Ward (2010) propose a threshold policy to determine server pool priority in a call center with server pools of different skill levels. Véricourt, Jennings (2011) studies the nurse staffing problem by constructing a $M/M/s//n$ model, and compared their policy against a fixed nurse-to-patient policy. Yom-Tov, Mandelbaum (2014) studies a queueing network with reentrant customers which they call as Erlang-R.

To arrive at a decision, we observe one of two methods in the literature. In the first instance, the steady state dynamics of the queueing network is derived as a function of the decisions, upon which the performance metrics are optimized. For example, Puha, Ward (2019) studies a multi-class many-server queueing network with impatient customers, and use the restricted fluid model to approximately solve the scheduling control problem. Most related to our work is that by Chan et al. (2021), who examine the dynamic assignment of servers in a multiclass queueing system. By assuming Poisson arrivals and exponential service times, the authors construct a deterministic discrete and finite time fluid control approximation, which is asymptotically optimal.

We find it challenging to apply this method to our problem setting because, one, our problem is fundamentally posed in the transient setting (and which later, in our numerical experiments, we shall provide the evidence to justify that there are significant consequences in ignoring the transient nature of the problem), and two, in some cases, the fluid approximation does not lead to state-dependent solutions.

In the second instance, to derive state-dependent solutions, the authors often models the problem in a dynamic programming framework. For example, Martonosi (2011) studies whether dynamically reassigning servers to parallel queues in response to queue imbalances can reduce average waiting time in those queues by approximate dynamic programming method. They observe that dynamically reallocating servers can substantially reduce waiting time in some circumstances. Most relevant to our work is that by Dai, Shi (2019), who study the inpatient overflow problem to decide how patients should be assigned to non-primary wards when their original primary ward are fully occupied. The authors model the problem as a multiclass, multipool parallel-server queuing system in a Markov decision process framework, and also considered the interplay between patient length-of-stay and the discharge process. They eventually employ an approximate dynamic programming approach to solve the model.

Stochastic programming: There is another stream of literature that models the problem under a stochastic programming framework. Most relevant is the work by Lyu et al. (2019), who address the multi-objective online ride-matching problem by online convex optimization techniques. Setting the platform revenue, pick-up distance, and service quality as the multiple objectives, the authors prove that their policy can achieve an optimal solution that is closest to any pre-determined multi-objective target. Also, Guo et al. (2018) study the virtual machine scheduling problem in cloud computing systems, and propose several online algorithms that they illustrate to yield good performance. Jaillet, Lu (2014) examine the online stochastic bipartite matching problem for advertising applications. Based on the solution of simple linear programs of maximum flow problems, the online algorithm in the paper is shown to be computationally efficient with better bounds for the competitive ratio. Correia et al. (2018) consider a two-stage modeling framework for the multi-period stochastic capacitated multiple allocation hub location problem. Works like Youssef et al. (2021) and Rostami et al. (2021) also examine a similar context.

Learning: The other stream that is gaining prominence of late is the learning approach. This can be broadly described in two parts, static learning and online learning. For the static learning literature, we see work such as Dai, Gluzman (2020), who employ the advanced policy gradient policy framework in deep reinforcement learning to a large-size multiclass queueing network with parallel servers. The algorithm consistently generates control policies that outperform state-of-art heuristics. He et al. (2019) develop a data-driven robust framework to study the patient scheduling problem in an emergency department balancing off the patients' door-to-provider times and lengths of stay. Their hybrid robust-stochastic approach is computationally efficient. Some works, such as Batt et al. (2019), who study the effects of work shifts of care providers on the performance of emergency departments using regression models, focus specifically on understanding and correctly modelling the dynamics of the system.

The online learning approach is also seen in the literature as it is able to make use of the sequentially available data to make here-and-now decisions. Johari et al. (2021) consider the problem faced by a service platform that needs to match limited supply with demand, where the demand type is unknown. The planner needs to learn the type of the demand over time to maximize the payoff. Fershtman, Pavan (2015) study the dynamics of mediated many-to-many matching allocation in a two-sided market, in which agents' preferences evolve over time. However, they do not consider the stochastic entry or departure of agents. Özkan, Ward (2020) propose matching policies for ride-sharing, based on a continuous linear program that accounts for the differing arrival rates of customers and drivers in different areas of the city, the customers' and the drivers' willingness to wait. They consider time-varying factors. When pricing affects customer and driver arrival rates and parameters are time homogeneous, they show that asymptotically optimal joint pricing and matching decisions lead to fully utilized drivers under mild conditions.

Summary: In a broad sense, our approach lies within the intersection of the queueing and the stochastic approaches. We shall apply the Pipeline Queue paradigm (Bandi, Loke, 2018), which can be viewed as using robust optimization and stochastic techniques to solve for specially constructed discrete queueing dynamics. The learning literature is less relevant to our work as in the problem setting we have described, we assume that the decision-maker possesses sufficient data and moreover, the value of the allocation is known, along with the source of the demand or the supply.

4.1.2 Approach and contributions

In this paper, to deal with the complex dynamics of the state and the uncertainties arising from the double stochastic nature of demand and supply, and also the need to solve for state-dependent policies in the transient setting, we appeal to the Pipeline Queue paradigm introduced in (Bandi, Loke, 2018). There, the authors propose a model that sought to reduce waiting times in queueing networks via an optimization model that decides on the flow of jobs within the network. This is applicable in our context as the flow of jobs here is represented by the allocation of demand to supply. Moreover, in Bandi, Loke (2018), their model allows capacity to enter into the decision variables, and this in our context, would allow us to optimize for the distribution of supply across sources. This paradigm has seen subsequently works such as Zhou et al. (2021) and Tang et al. (2020) in the areas of intraday patient scheduling and vehicle re-positioning respectively. These works indicate good out-of-sample performance against traditional methodologies such as sample average approximations (SAA) and fluid approximations.

Unfortunately, the Pipeline Queue model cannot be directly applied to the resource allocation setting. This is primarily due to the way the Pipeline Queue model defines the allocation between demand and supply as a static variable. In our later simulations, we numerically illustrate the dangers of directly applying the Pipeline Queue model to our problem. As such, we shall attempt, in this paper, to exploit the structure of the demand and supply to construct a new framework for the resource allocation problem under the context we defined, while preserving core ideas in the Pipeline Queue paradigm.

Key contributions

In this paper, we present a wholly new model that simultaneously decides on the allocation of incoming demand to existing server pools, and the distribution of supply across the server pools, under a doubly stochastic demand and service time. Our policy is adaptive, which we term the 'distributive decision rule', and can be tractably solved. In the numerical section, we illustrate insights on how the optimal policy changes as service distribution, initial conditions, temporal fluctuations in demand, resource availability, and network connectivity are varied. Our model turns out to be very dynamic and responses to these varying conditions. We then proceed to compare our model against the original Pipeline Queue model and a state-of-the-art model proposed in Chan et al. (2021). We find our model dominating in the practical settings we defined above.

Aside from the key contribution of this paper to the resource allocation setting, the paper also advances the growing literature in Pipeline Queues, by introducing new methodologies. In particular, it is the first illustration of the pipeline decision rule as applied on queues, as opposed to servers. The additional benefit is that it allows some independence requirements, assumed in Bandi, Loke (2018), to be avoided.

Organization of the paper

We first formally introduce the problem in Section 4.2 and describe the model that we propose. This will be accompanied by proofs detailing the model reformulation. In Section 4.3, we shall perform numerical experiments by exploring how the model varies under different conditions in the service time distribution, arrivals, initial conditions, and other factors. We also compare our model against other models in the literature. Finally, we conclude with some remarks in Section 4.4.

Notation. We use the calligraphic font (*e.g.*, $\mathcal{A}$) to denote sets, and boldface lowercase letters for vectors (*e.g.*, $\boldsymbol{\theta}$). We use $[N]$ to denote the running index $\{1, 2, 3, \ldots, N\}$ for a known integer N. We adopt the convention that $\inf \emptyset = +\infty$, where $\emptyset$ is the empty set. We also use $\text{Bin}(n, p)$ to refer to the Binomial random variable of n independent trials, each with probability p.

4.2 An distributive model for allocation problems

We consider a formulation for the allocation problem as follows: Given a finite time horizon $\mathcal{T} := \{0, 1, 2, \ldots, T\}$, define a bipartite graph $\mathcal{G}$, as shown in Figure 4.1, with two partitions, representing a demand set $\mathcal{I} := \{1, 2, \ldots, I\}$ and a supply set $\mathcal{J} := \{1, 2, \ldots, J\}$ respectively, and edges (i, j) for $i \in \mathcal{I}$ and $j \in \mathcal{J}$ depicting that supply source j can serve demand type i.

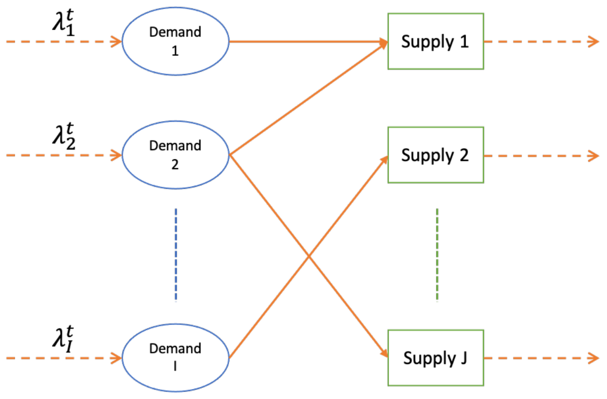


Figure 4.1: The bipartite structure of allocation problems.

At the start of every time period, new jobs arrive at each demand node i. This can be deterministic, stochastic, or a combination of both. They then enter the queue. Each supply node represents a team of finite number of servers. Each job can be serviced by one server. The time duration that each server in a team finishes a job can be deterministic, stochastic, or a combination of both. As capacity is freed up in the supply nodes, the decision-maker decides which of the supply nodes should service the demand from which demand node. As such, the decisions are the flows on the edges (i, j) across all time $t \in \{1, 2, \ldots, T\}$. The decision-maker might also have the flexibility to decide the capacity on the supply nodes, subject to some constraints (such as budget).

In what ensues, we shall attempt to imbue into this model the five distinctive contexts of the problem class that we want to consider:

1. *Replenishable supply*: Supply is modelled as servers uncertain service times. They are freed up after a random period of time.
2. *Control over allocation and supply*: We shall define decisions on both the edges linking demand to supply and the capacity.
3. *Disjoint demand and supply sources*: This is modelled by the different demand and supply nodes.
4. *Demand need not be immediately fulfilled*: We shall model this as waiting cost in the queue.
5. *Known allocation value*: We shall define known costs in the edges.

Discrete time model

Before introducing our model formally, we present a simpler version to motivate our model formulation. Let x_i^t denote the number of jobs of demand of type i that has not yet been dispatched to any supply nodes at time t. This can be interpreted as the queue for jobs of type i. Similarly, let y_j^t denote the number of units that are currently in service at supply node j at time t. At the start of each time period t, λ_i^t (drawn from some distribution Λ_i^t) units of new jobs arrive at demand node i. The decision-maker then decides to dispatch some $w_{i,j}^t$ of the demand from demand node i into supply node j. Certainly, $w_{i,j}^t = 0$ if (i, j) is not an edge in $\mathcal{G}$. As such, the dynamics in the demand nodes i are as follows:

$$x_i^{t+1} = x_i^t + \lambda_i^t - \sum_{j \in \mathcal{J}} w_{i,j}^t. \tag{4.1}$$

Simultaneously, some jobs in each supply node are completed and leave the system. Let the number of them be z_j^t for supply node j at time t. The dynamics in the supply nodes j are as follows:

$$y_j^{t+1} = y_j^t + \sum_{i \in \mathcal{I}} w_{i,j}^t - z_j^t. \tag{4.2}$$

While this formulation is simple, the key difficulty is in modelling z_j^t accurately. Objectively, we would want $z_j^t := z_j(y_j^t)$ to be some random function of y_j^t. However, at this point, we have not tracked any information regarding the current time under service of each unit. As such, it is difficult to define z_j^t accurately. Indeed, suppose a server in supply node j takes exactly 5 units of time to complete a job. Then z_j^t will vary drastically depending on how long the units in supply node j have spent, specifically whether or not they have spent 5 units of time in node j or not.

Such difficulties in modelling the service time distribution is, arguably, one of the drawbacks of the queuing approach in modelling transient systems such as these. Certainly, z_j^t can be modelled by a stationary distribution in the steady-state as in the fluid approximation approach. However, our problems are fundamentally posed in the transient setting, and this would not be tenable. Moreover, although this formulation is able to track the queue lengths in the demand nodes, it can not tell how many periods each job has waited for, which is related to the queuing cost.

Introducing present delay s

The Pipeline Queues paradigm (Bandi, Loke, 2018) attempts to directly solve this problem by defining the dynamics with the additional dimension of time spent in the node. Reusing earlier notation, let $x_i^{t,s}$ denote the number of units of demand of type i that at time t, have been awaiting dispatch for s amount of time since they arrived. We can think of s as tracking the currently experienced waiting time, or *present delay*, for these demand units. Upon this base, $w_{i,j}^{t,s}$ jobs that have waited in demand node i for s periods are routed at time t to supply node j. Similarly, $y_j^{t,s}$ denotes the number of units that have already been in service at supply node j for s amount of time at time period t. Hence, for the servers, s represents the currently experienced service time. As before, λ_i^t gives the new demand of type i. This leads to the dynamics in the demand nodes i, for all times

$t \geq 0$:

$$x_i^{t+1,0} = \lambda_i^{t+1} \tag{4.3}$$

$$x_i^{t+1,s+1} = x_i^{t,s} - \sum_{j \in \mathcal{J}} w_{i,j}^{t,s}, \qquad \forall s \geq 0. \tag{4.4}$$

To accurately define the dynamics on job completion, let $q_j^{t,s}$ denote the probability that a unit, having, at time t, been in service in supply node j for s periods, will not complete in period t.

Assumption 1. *The probability of job completion of any job, conditioned on the amount of time that they have been served, is independent of any other job (and their completion).*

Independence here could be understood as supply node j being modelled as a group of identical parallel servers. The capacity of this supply node is interpreted as the number of parallel servers. This interpretation is particular intuitive in the setting of vehicle repositioning, say, where each parallel server represents a vehicle. As such, the number of job completions at each supply node follows a Binomial distribution. Indeed, this can represent any general discrete time distribution on job completion (Dai, Shi, 2017). Then, the dynamics in the supply node j is:

$$y_j^{t+1,0} = \sum_{i \in \mathcal{I}} \sum_{s} w_{i,j}^{t,s} \tag{4.5}$$

$$y_j^{t+1,s+1} = \text{Bin}(y_j^{t,s}, q_j^{t,s}) \qquad \forall s \geq 0, \tag{4.6}$$

where the Binomial distribution models the completion of units as a trial on $y_j^{t,s}$ units with success probability $1 - q_j^{t+1,s+1}$. Denote $x_i^{0,s}$ (for $i \in \mathcal{I}$ and $s \geq 0$) and $y_j^{0,s}$ (for $j \in \mathcal{J}$ and $s \geq 0$) as the initial conditions at period 0. The dynamics (4.6) simplifies to

$$y_j^{t+1,s+1} = \begin{cases} \text{Bin}(y_j^{t-s,0}, p_j^{t+1,s+1}), & t > s \\ \text{Bin}(y_j^{0,s-t}, p_j^{t+1,s+1}), & t \leq s \end{cases}, \qquad \text{where} \quad p_j^{t+1,s+1} = \begin{cases} q_j^{t-s,0}, \ldots, q_j^{t,s}, & t > s \\ q_j^{0,s-t}, \ldots, q_j^{t,s}, & t \leq s \end{cases} \tag{4.7}$$

with $p_j^{t,s}$ representing the cumulative survival probability of not completing service before s time periods at time t.

Assumption 2. *Demand arrivals are exogenous with finite support. Furthermore, the arrivals at different time points $t = 1, \dots, T$ and to different demand nodes $i = 1, \dots, I$ are also independent of each other.*

Ostensibly, one would have a well-defined dynamics at this point. However, the key difficulty lies in modelling the decision variables $w_{i,j}^{t,s}$. There are two core difficulties. The first is with regards to the static or adaptive nature of $w_{i,j}^{t,s}$. If one chooses to model $w_{i,j}^{t,s}$ as static, the model could potentially be conservative with longer time horizon T. Moreover, they would not be adaptive. However, if one demands that $w_{i,j}^{t,s} := w_{i,j}^{t,s}(\boldsymbol{x}^t, \boldsymbol{y}^t)$, then the solution space is potentially very large and the resulting model is likely intractable.

The second difficulty lies in how to enforce constraints such as $\sum\limits_{j \in \mathcal{J}} w_{i,j}^{t,s} \leq x_i^{t,s}$, where the right-hand side is a random variable. The solution to the second difficulty was proposed in the Pipeline Queue framework by utilizing a proportion-based decision rule. This also partially addresses the first difficulty as it reduces the solution space sharply. However, as it stands, Pipeline Queues do not permit such decisions rules to be applied on queues. Instead, 'dispatch variables', which represented flow out of queues, were modelled as static variables. For example, in Tang et al. (2020), the authors used static variables to model the assignment of passengers to cars in a vehicle repositioning problem, which falls under the umbrella of resource allocation problems that we are considering. Critically, if these dispatch variables were forcibly modelled as proportion-based decision rules, then to the best of our knowledge, there is no way to maintain the tractability of the model, which is by far, one of the most attractive features of the Pipeline Queues framework. Indeed, even for the servers, on which the proportion-based decision rules were used, there are strict conditions, for example, the nodes that these edges are adjacent to may only be recipient to maximally one of such flows. These conditions ensured that their construction remained tractable.

4.2.1 Model and formulation

The key insight is to realize that the proportion-based decision rule can in fact be adapted to the resource allocation problem, as we have defined, due to its unique structure as a bipartite graph. Nonetheless, this would require some new techniques to be worked through in the reformulation process, that eventually arrives at a wholly new framework. Here, we present our fully adaptive, but tractable formulation for the resource allocation problem.

Variables, dynamics and constraints

We retain all of the previous notation of $\boldsymbol{x}$, $\boldsymbol{y}$, $\boldsymbol{\lambda}$, and $\boldsymbol{q}$ (resp. $\boldsymbol{p}$) to denote the state variables for the demand nodes, supply nodes, the arrival of new demand and the completion probabilities (resp. cumulative probabilities) for the supply nodes. Instead, we transform the additive dynamics $x_i^{t+1,s+1} = x_i^{t,s} - \sum\limits_{j \in \mathcal{J}} w_{i,j}^{t,s}$ via the distributive decision rule:

$$w_{i,j}^{t,s} = x_i^{t,s} \frac{\alpha_{i,j}^{t+1,s+1}}{\beta_i^{t,s}}. \tag{4.8}$$

This is akin to simply just the multiplicative rule $w_{i,j}^{t,s} = x_i^{t,s} \gamma_{i,j}^{t,s}$, except that we have expressed $\gamma_{i,j}^{t,s}$ as a fraction for purposes of tractability. It also affords us the flexibility of normalizing $\beta_i^{t,0}$ and $\beta_i^{0,s}$ at a later convenient point. For now, we shall only assume for the latter:

$$\beta_i^{0,s} = x_i^{0,s}. \tag{4.9}$$

This has the effect of transforming the dynamics to the following form:

$$y_j^{t+1,0} = \sum_{i \in \mathcal{I}} \sum_{s} x_i^{t,s} \frac{\alpha_{i,j}^{t+1,s+1}}{\beta_i^{t,s}}, \tag{4.10}$$

Consequently, the dynamics on the outflow from the demand nodes i would be expressed as

$$x_i^{t,s} - x_i^{t+1,s+1} = x_i^{t,s} \sum_j \alpha_{i,j}^{t+1,s+1} / \beta_i^{t,s}. \tag{4.11}$$

This preserves the flow balance between the demand and supply nodes, *i.e.*, ensuring that the total jobs dispatched from the demand nodes equate to the total jobs dispatched into the supply nodes. This is not always desirable. For example, it may be desirable to reject some jobs when the system becomes overloaded. Alternatively, it may also be desirable to commit to doing more jobs than there are at present in the system, as this creates flexibility.

With this in mind, we adopt the following representation with the option to relate $\boldsymbol{\alpha}$ with $\boldsymbol{\beta}$:

$$x_i^{t+1,s+1} = x_i^{t,s} \frac{\beta_i^{t+1,s+1}}{\beta_i^{t,s}}, \tag{4.12}$$

There are three possible options for relating $\boldsymbol{\alpha}$ to $\boldsymbol{\beta}$ to describe the different forms of flow balance:

a) <u>Exact balance</u>: If $\sum_{j \in \mathcal{J}_i} \alpha_{i,j}^{t+1,s+1} + \beta_i^{t+1,s+1} = \beta_i^{t,s}$, exact flow balance is maintained. In this option, however, it is not clear whether the ensuing formulation will be convex.

b) <u>Overloading commitment</u>: If $\sum_{j \in \mathcal{J}_i} \alpha_{i,j}^{t+1,s+1} + \beta_i^{t+1,s+1} \geq \beta_i^{t,s}$, then the sum of jobs entering the server nodes is greater than the number of jobs released from the demand node. This is equivalent to committing to doing more jobs than necessary. Nonetheless, overcommitting jobs to the servers will only increase the chance of their capacity being exceeded. As such, in the optimal case, this constraint ought to be tight. Thus, this can be viewed as a convex relaxation of the exact balance case. Notice that it is necessary to accompany this constraint with $\beta_i^{t+1,s+1} \leq \beta_i^{t,s}$, to ensure that no new jobs are created within the demand node itself.

c) Controlled rejection: If $\sum_j \alpha_{i,j}^{t+1,s+1} + \beta_i^{t+1,s+1} \leq \beta_i^{t,s}$, then there is a loss of jobs from the system. This can be viewed as a decision to reject jobs (see Jaillet et al., 2019, for an analogous example on firing employees). It would be necessary, however, to control the number of jobs rejected. This might take the form of the following constraint:

$$\sum_{i \in \mathcal{I}} \sum_s \frac{\beta_i^{t,s} - \sum_{j \in \mathcal{J}} \alpha_{i,j}^{t+1,s+1} - \beta_i^{t+1,s+1}}{\beta_i^{t,s}} x_i^{t,s} u_i^s \leq U^t, \tag{4.13}$$

where the left-hand side represents the total cost of rejecting the demand units at time t, scaled by some u_i^s that depends on the time duration s they have waited for before being rejected and the demand type i.

As we shall soon see, the nature of allocation problems being posed as a bipartite graph allows this distributive rule to be tractable, despite the mixture of additive and multiplicative forms in Equation 4.10. The totality of variables defined are summarized in Table 4.1.

		Parameters and known quantities
$\mathcal{T}$	:	Time horizon $\{1, \ldots, T\}$
$\mathcal{I}$	:	The set of all the types of demands, $\{1, \ldots, I\}$
$\mathcal{J}$	:	The set of all the types of supplies, $\{1, \ldots, J\}$
$\mathcal{I}_j$	:	The subset of all the demand types that can be served by team j.
$\mathcal{J}_i$	:	The subset of all the supply types that are capable of serving demand type i
$\mathcal{K}^t$	:	The capacity of all the supply teams at time t, $(\mathcal{K}_1^t, \ldots, \mathcal{K}_J^t)$
λ_i^t	:	The new arrivals of demand type i at the beginning of period t
$q_j^{t,s}$	:	The survival probability of a job at supply node j which has been served for s periods at time t.
$p_j^{t,s}$	:	The accumulative survival probability.
		State and decision variables
$\mathcal{X}_i^{t,s}$	:	The set of demand of type i that have been awaiting dispatch for s periods at time t.
$\mathcal{Y}_j^{t,s}$	:	The set of the jobs that have stayed in supply node j for s periods at time t
$x_i^{t,s}$	:	The number of units of demand of type i that have been awaiting dispatch for s periods at time t.
$y_j^{t,s}$	:	The number of demands that have stayed in supply node j for s periods at time t
$\alpha_{i,j}^{t,s}$	:	The fraction of $\mathcal{X}_i^{t,s}$ that is dispatched to supply node j at period t
$\beta_i^{t,s}$	:	The fraction of $\mathcal{X}_i^{t,s}$ that are retained in the demand node i at period t.
$w_{i,j}^{t,s}$	:	The number of jobs in $\mathcal{X}_i^{t,s}$ that is allocated to team j in period t.

Table 4.1: List of Parameters and Variables

With the dynamics, we can now describe the constraints. In particular, there are four types of constraints that we shall define.

$$\sum_s y_j^{t,s} \leq \mathcal{K}_j^t, \qquad \forall j \in \mathcal{J}, \forall t > 0 \tag{Capacity}$$

$$\sum_s b_i^{t,s} x_i^{t,s} \leq C_i^t \sum_s x_i^{t,s}, \qquad \forall i \in \mathcal{I}, \forall t > 0 \qquad \text{(Cost)}$$

$$\sum_i \sum_j \sum_s a_{i,j}^{t,s} x_i^{t,s} \frac{\alpha_{i,j}^{t+1,s+1}}{\beta_i^{t,s}} \leq A^t, \qquad \forall t \geq 0 \qquad \text{(Allocation)}$$

$$\beta_i^{t,s} \lesseqqgtr \beta_i^{t+1,s+1} + \sum_j \alpha_{i,j}^{t+1,s+1}, \qquad \forall i \in \mathcal{I}, \forall t, s > 0 \qquad \text{(Flow balance)}$$

Capacity constraints limit the supply $\mathcal{K}_j^t$ on each of the supply nodes j at time t. We allow this capacity to vary with time t. If it is desired that $\mathcal{K}_j^t$ is a decision variable, then additional constraints limiting $\mathcal{K}_j^t$, for example $\sum_j \mathcal{K}_j^t u_j \leq \mathcal{K}$ describing a cost constraint on capacity allocation, may be required. The second type of constraint pertains to cost constraints, here phrased as an average cost constraint. In particular, this is very helpful in defining service level guarantees, such as average waiting time (see Bandi, Loke, 2018, for more details on the options available and the expressibility of this constraint). The third type of constraint refers to the revenue or cost associated with making an allocation at time t. $a_{i,j}^{t,s}$ represents the unit cost of dispatching one job from demand node i that has waited for s periods to supply node j at time t, and A^t is the total budget at time t (or $-a_{i,j}^{t,s}$ and $-A^t$ in the case of revenue). Finally, as described earlier, there is a choice on how the decision-maker would like the flow balance to be modelled.

Model and Reformulation

The pipeline queues paradigm proceeds by transforming these constraints under the entropic value of risk. This procedure, akin to risk-correcting the constraints, yields probabilistic guarantees against constraint violation. The resultant model would turn out to be tractable as long as the constraints under the entropic value of risk operator have convex reformulations. In the ensuing discussion, we emulate this procedure to describe our model and to obtain its equivalent reformulation under the pipeline framework.

Notice first that the capacity and cost constraints, including the rejection constraint (Inequality 4.13), contain random variables $\boldsymbol{x}$ and $\boldsymbol{y}$, but are linear in these variables. As such, consider the following versions of these constraints under the entropic value at risk:

$$k \log \mathbb{E} \exp \left(\left(\sum_s y_j^{t,s} - \mathcal{K}_j^t \right) \Big/ k\theta_{1,j}^t \right) \le 0, \qquad \forall j \in \mathcal{J}, \forall t > 0 \qquad \text{(Capacity}'\text{)}$$

$$k \log \mathbb{E} \exp \left(\left(\sum_s (b_i^{t,s} - C_i^t)\, x_i^{t,s} \right) \Big/ k\theta_{2,i}^t \right) \le 0, \qquad \forall i \in \mathcal{I}, \forall t > 0 \qquad \text{(Cost}'\text{)}$$

$$k \log \mathbb{E} \exp \left(\left(\sum_i \sum_j \sum_s a_{i,j}^{t,s} x_i^{t,s} \frac{\alpha_{i,j}^{t+1,s+1}}{\beta_i^{t,s}} - A^t \right) \Big/ k\theta_3^t \right) \le 0, \qquad \forall t \ge 0. \qquad \text{(Allocation}'\text{)}$$

where $\boldsymbol{\theta}$ are parameters to tighten the risk aversion idiosyncratic to each constraint. For example, capacity constraints should never be violated and are hard constraints, whereas cost constraints are soft constraints that represent targets to be aimed for.

These constraints yield probabilistic guarantees against constraint violation:

Proposition 1. *If k obeys $k \log \mathbb{E} \exp (X/k\theta) \le 0$, for any $\phi > 0$, we have $\mathbb{P}(X \ge \phi) \le \exp(-\phi/k\theta)$.*

Proof. Proof $k \log \mathbb{E} \exp (X/k\theta) \le 0$ is equivalent to $\mathbb{E} [\exp(X/k\theta)] \le 1$. By Markov's inequality, we have

$$\begin{aligned} \mathbb{P}[X \ge \phi] &= \mathbb{P} [\exp(X/k\theta) \ge \exp(\phi/k\theta)] \\ &\le \mathbb{E} [\exp(X/k\theta)] / \exp(\phi/k\theta) \\ &\le \exp(-\phi/k\theta). \end{aligned} \qquad (4.14)$$

□

As such, the decision-maker may choose to minimize these guarantees against constraint violation by minimizing the risk level k. From Proposition 1 above, we can see that as the bounds on the chance of violation are monotone in k. This leads to the proposed model:

$$
\begin{array}{llr}
\text{minimize} & k & (4.15)\\
\text{subject to} & k \log \mathbb{E} \exp \left(\left(\sum_s y_j^{t,s} - \mathcal{K}_j^t \right) \Big/ k\theta_{1,j}^t \right) \le 0, & \forall j \in \mathcal{J}, \forall t > 0\\
& k \log \mathbb{E} \exp \left(\left(\sum_s \left(b_i^{t,s} - C_i^t \right) x_i^{t,s} \right) \Big/ k\theta_{2,i}^t \right) \le 0, & \forall i \in \mathcal{I}, \forall t > 0\\
& k \log \mathbb{E} \exp \left(\left(\sum_i \sum_j \sum_s a_{i,j}^{t,s} x_i^{t,s} \frac{\alpha_{i,j}^{t+1,s+1}}{\beta_i^{t,s}} - A^t \right) \Big/ k\theta_3^t \right) \le 0, & \forall t \ge 0\\
& + \text{ Choice of flow balance constraint} & \\
& \beta_i^{t,s} \le \beta_i^{t-1,s-1} & \\
& \beta_i^{t,s} \ge 0,\ \alpha_{i,j}^{t,s} \ge 0 & \\
\text{with dynamics} & x_i^{t,0} = \lambda_i^t, & \forall t > 0\\
& x_i^{t,s} = x_i^{t-1,s-1} \beta_i^{t,s} / \beta_i^{t-1,s-1}, & \forall t, s > 0\\
& y_j^{t,0} = \sum_{i \in \mathcal{I}_j} \sum_s x_i^{t-1,s} \frac{\alpha_{i,j}^{t,s+1}}{\beta_i^{t-1,s}}, & \forall t > 0\\
& y_j^{t,s} \sim \text{Bin}(y_j^{t-1,s-1}, q_j^{t-1,s-1}), & \forall t, s > 0
\end{array}
$$

The crux of the problem now is whether the three entropic value-at-risk constraints (Capacity′), (Cost′) and (Allocation′) have convex reformulations. We start with the cost reformulation first:

Proposition 2 (Cost Reformulation)**.** *Let $g_i^t(\omega) = \log \mathbb{E}[e^{\omega \lambda_i^t}]$ represent the log-moment generating function of Λ_i^t. Then,*

i. For a given t, the state variables in set $\{x_i^{t,s} : i \in \mathcal{I}, s\}$ are independent.

ii. Let $\tilde{b}_i^{t,s} = b_i^{t,s} - C_i^t$, and the cost constraint may be reformulated as follows:

$$
k \log \mathbb{E} \exp \left[\left(\sum_s \tilde{b}_i^{t,s} x_i^{t,s} \right) \Big/ k\theta_{2,i}^t \right] = k \sum_{s=0}^{t-1} g_i^{t-s} \left(\frac{\tilde{b}_i^{t,s} \beta_i^{t,s}}{\beta_i^{t-s,0} k\theta_{2,i}^t} \right) + \sum_{s \ge t} \tilde{b}_i^{t,s} \beta_i^{t,s} \Big/ \theta_{2,i}^t \quad (4.16)
$$

Proof. Proof of Proposition 2.

i. For a given t, we have:

$$
x_i^{t,s} = \begin{cases} x_i^{t-1,s-1} \dfrac{\beta_i^{t,s}}{\beta_i^{t-1,s-1}} = \ldots = \lambda_i^{t-s} \dfrac{\beta_i^{t,s}}{\beta_i^{t-s,0}} & s < t \\ x_i^{t-1,s-1} \dfrac{\beta_i^{t,s}}{\beta_i^{t-1,s-1}} = \ldots = x_i^{0,s-t} \dfrac{\beta_i^{t,s}}{\beta_i^{0,s-t}} & s \ge t \end{cases}
$$

As we assume the demand arrivals are independent for all t and $\mathcal{I}$, $x_i^{0,s}$ are constants for all s and $\mathcal{I}$, and all $\beta_i^{t,s}$, $\beta_i^{t,s}$ are decision variables, we can conclude that $\{x_i^{t,s} : i \in \mathcal{I}, s\}$ are independent.

ii. As $x_i^{t,s}$ are independent across all s, we get that:

$$\begin{aligned}
k \log \mathbb{E} \exp \left[\left(\sum_s \tilde{b}_i^{t,s} x_i^{t,s} \right) \Big/ k\theta_{2,i}^t \right] &= \sum_s k \log \mathbb{E} \exp \left(\tilde{b}_i^{t,s} x_i^{t,s} \big/ k\theta_{2,i}^t \right) \\
&= \sum_{s=0}^{t-1} k \log \mathbb{E} \exp \left(\frac{\tilde{b}_i^{t,s} \beta_i^{t,s}}{\beta_i^{t-s,0} k\theta_{2,i}^t} \lambda_i^{t-s} \right) + \sum_{s \geq t} \frac{\tilde{b}_i^{t,s} \beta_i^{t,s}}{\beta_i^{0,s-t} \theta_{2,i}^t} x_i^{0,s-t} \\
&= k \sum_{s=0}^{t-1} g_i^{t-s} \left(\frac{\tilde{b}_i^{t,s} \beta_i^{t,s}}{\beta_i^{t-s,0} k\theta_{2,i}^t} \right) + \sum_{s \geq t} \frac{\tilde{b}_i^{t,s} \beta_i^{t,s}}{\theta_{2,i}^t}
\end{aligned}$$

□

Here, it is most critical to notice that the expression in Equation 4.16 is *jointly* convex in the decision variables β and k. Note first that $\beta_i^{t-s,0}$ is not part of the decision variables, and is a degree of freedom afforded to us by the definition of the distributive decision rule. One may set $\beta_i^{t-s,0}$ as any value here, but in practice, the empirical mean of the arrivals is chosen (see Remark 4.1 later). Next, notice that $g_i^{t-s}(\,\cdot\,)$ is a convex function – indeed, the logarithm of any moment generating function is convex. Hence, the expression $k \cdot g_i^{t-s}(\zeta \beta_i^{t,s}/k)$ is jointly convex in both k and $\beta_i^{t,s}$ for any ζ. This convexity will enable us to solve for Equation 4.16 efficiently later, as we discuss in the next subsection.

Remark 4.1. The expression $g_i^{t-s}\left(\frac{\tilde{b}_i^{t,s} \beta_i^{t,s}}{\beta_i^{t-s,0} k\theta_{2,i}^t} \right)$ in Equation 4.16 might appear to be unwieldy. If the family of distribution of Λ_i^{t-s} is known, then there are potential simplifications. Indeed, if Λ_i^{t-s} is Poisson-distributed with mean μ_i^{t-s}, then we have that

$$g_i^{t-s} \left(\frac{\tilde{b}_i^{t,s} \beta_i^{t,s}}{\beta_i^{t-s,0} k\theta_{2,i}^t} \right) = \log \mathbb{E} \exp \left(\frac{\tilde{b}_i^{t,s} \beta_i^{t,s}}{\beta_i^{t-s,0} k\theta_{2,i}^t} \lambda_i^{t-s} \right) = \mu_i^{t-s} \cdot \exp \left(\frac{\tilde{b}_i^{t,s} \beta_i^{t,s}}{\beta_i^{t-s,0} k\theta_{2,i}^t} - 1 \right).$$

Here, if we set $\beta_i^{t-s,0} = \mu_i^{t-s}$, then notice that the expression becomes a perspective in μ_i^{t-s}! This means that it is possible to allow even μ_i^{t-s} to enter the decision variables. In some problem settings, this might be useful, for example, if the decision-maker is able to control the inflow with some cost. A similar situation also occurs for other distributions such as the Gamma distribution.

A similar reformulation exists for the (Allocation′) constraint:

Proposition 3 (Allocation Reformulation)**.** *The allocation constraint may be reformulated as follows, and is jointly convex in k and decision variables $\alpha_{i,j}^{t,s}$:*

$$k \log \mathbb{E} \exp \left(\left(\sum_i \sum_j \sum_s a_{i,j}^{t,s} x_i^{t,s} \frac{\alpha_{i,j}^{t+1,s+1}}{\beta_i^{t,s}} \right) \Big/ k\theta_3^t \right) \tag{4.17}$$
$$= k \sum_{s=0}^{t-1} \sum_i g_i^{t-s} \left(\sum_j \frac{a_{i,j}^{t,s} \alpha_{i,j}^{t+1,s+1}}{\beta_i^{t-s,0} k\theta_3^t} \right) + \sum_{s \geq t} \sum_i \sum_j a_{i,j}^{t,s} \alpha_{i,j}^{t+1,s+1} \Big/ \theta_3^t$$

Proof. Proof of Proposition 3. The proof is analogous to Proposition 2 and is omitted for brevity. □

The reformulation of (Capacity′) is not straightforward. In particular, unlike the derivation in Proposition 2, $y_j^{t,s}$ are not independent across s for fixed t and j. The jobs arriving in the same cohort may be dispatched to the same supply node j at different time periods. As a result, the jobs coming from the same demand node may have been served for different time duration at the same supply node j at time t. This is the key difference where our proposed formulation departs from pipeline queues. It turns out, that (Capacity′) can nonetheless still be evaluated.

Theorem 4.1 (Capacity Reformulation)**.** *The general capacity constraint on the supply nodes j has the following reformulation:*

$$k \log \mathbb{E} \exp \left[\left(\sum_s y_j^{t,s} - \mathcal{K}_j^t \right) \Big/ k\theta_{1,j}^t \right]$$
$$= \sum_{s: s \geq t} k r_j^{t,s} (k\theta_{1,j}^t) y_j^{0,s-t} + k \sum_{i \in \mathcal{I}_j} \sum_{t'=1}^{t-1} g_i^{t'} \left(\sum_{\tau=1}^{t-t'} \frac{r_j^{t,t-t'-\tau} (k\theta_{1,j}^t) \alpha_{i,j}^{t'+\tau,\tau}}{\beta_i^{t',0}} \right)$$
$$+ \sum_{i \in \mathcal{I}_j} \sum_{s=0}^{t-1} \sum_{\tau \geq t-s} k r_j^{t,s} (k\theta_{1,j}^t) \alpha_{i,j}^{t-s,\tau} - \mathcal{K}_j^t / \theta_{1,j}^t \tag{4.18}$$

where $r_j^{t,s}(\,\cdot\,) = \log\left(1 - p_j^{t,s} + p_j^{t,s}\exp\left(1/\,\cdot\,\right)\right)$*. Moreover, (4.18) is convex in the decision variables* $\boldsymbol{\alpha}$*.*

Proof. Proof of Theorem 4.1.

$$\begin{aligned}
&k\log\mathbb{E}\exp\left[\left(\sum_s y_j^{t,s} - \mathcal{K}_j^t\right)\Big/k\theta_{1,j}^t\right]\\
=&k\log\mathbb{E}\exp\left[\sum_{s=0}^{t-1} y_j^{t,s}\Big/k\theta_{1,j}^t)\right] + k\log\mathbb{E}\exp\left[\sum_{s\geq t} y_j^{t,s}\Big/k\theta_{1,j}^t\right] - \mathcal{K}_j^t/\theta_{1,j}^t
\end{aligned}\tag{4.19}$$

Equation (4.19) is because if $s \geq t$, $y_j^{t,s}$ is the number of jobs that are still in service at time t coming from $\mathcal{Y}_j^{0,s-t}$. Since $y_j^{0,s-t}$ is the initial state of the system, $\sum\limits_{s=0}^{t-1} y_j^{t,s}$ and $\sum\limits_{s\geq t} y_j^{t,s}$ are thus independent.

For the second term in (4.19), we have:

$$\begin{aligned}
&k\log\mathbb{E}\exp\left[\sum_{s\geq t} y_j^{t,s}\Big/k\theta_{1,j}^t\right]\\
=&k\log\mathbb{E}\exp\left[\sum_{s\geq t}\operatorname{Bin}\left(y_j^{0,s-t}, p_j^{t,s}\right)\Big/k\theta_{1,j}^t\right]\\
=&\sum_{s\geq t} k\log\mathbb{E}\exp\left[\operatorname{Bin}(y_j^{0,s-t}, p_j^{t,s})\Big/k\theta_{1,j}^t\right]\\
=&\sum_{s\geq t} r_j^{t,s}(k\theta_{1,j}^t)y_j^{0,s-t}
\end{aligned}$$

For the first term in (4.19) we have:

$$\begin{aligned}
&k\log\mathbb{E}\exp\left[\sum_{s=0}^{t-1} y_j^{t,s}\Big/k\theta_{1,j}^t\right]\\
=&k\log\mathbb{E}\left[\mathbb{E}\left[\exp\left[\left(\sum_{s=0}^{t-1}\operatorname{Bin}(y_j^{t-s,0}, p_j^{t,s})\right)\Big/k\theta_{1,j}^t\right]\middle|\, y_j^{t-s,0}, s=1,\dots,t-1\right]\right]\\
=&k\log\mathbb{E}\left[\prod_{s=0}^{t-1}\mathbb{E}\left[\exp\left(\operatorname{Bin}(y_j^{t-s,0}, p_j^{t,s})\Big/k\theta_{1,j}^t\right)\middle|\, y_j^{t-s,0}, s=1,\dots,t-1\right]\right] \qquad (4.20)\\
=&k\log\mathbb{E}_{\left\{y_j^{t',0}\right\}_{t'=1}^t}\left[\prod_{s=0}^{t-1}\exp\left(r_j^{t,s}(k\theta_{1,j}^t)y_j^{t-s,0}\right)\right]
\end{aligned}$$

$$=k\log\mathbb{E}_{\{\lambda_i^{t'}\}_{t'=1}^{t}}\left[\exp\left[\sum_{s=0}^{t-1} r_j^{t,s}(k\theta_{1,j}^t)\,y_j^{t-s,0}\right]\right] \tag{4.21}$$

$$=k\log\mathbb{E}\left[\exp\left[\sum_{s=0}^{t-1}\left(r_j^{t,s}(k\theta_{1,j}^t)\sum_{i\in\mathcal{I}_j}\left(\sum_{\tau=1}^{t-s-1}\lambda_i^{t-s-\tau}\frac{\alpha_{i,j}^{t-s,\tau}}{\beta_i^{t-s-\tau,0}}+\sum_{\tau\geq t-s}x_i^{0,\tau-s+t}\frac{\alpha_{i,j}^{t-s,\tau}}{\beta_i^{0,\tau-s+t}}\right)\right)\right]\right]$$

$$=\sum_{i\in\mathcal{I}_j}k\log\mathbb{E}\left[\exp\left[\sum_{s=0}^{t-1}\sum_{\tau=1}^{t-s-1}r_j^{t,s}(k\theta_{1,j}^t)\lambda_i^{t-s-\tau}\frac{\alpha_{i,j}^{t-s,\tau}}{\beta_i^{t-s-\tau,0}}\right]\right]+\sum_{i\in\mathcal{I}_j}\sum_{s=0}^{t-1}\sum_{\tau\geq t-s}kr_j^{t,s}(k\theta_{1,j}^t)\alpha_{i,j}^{t-s,\tau}. \tag{4.22}$$

The critical part of this derivation is the conditioning on the inflows $\left\{y_j^{t',0}\right\}_{t'=1}^{t}$, which we have emphasized for salience. This results in (4.20) – indeed, each Binomial term is a function of the inflows $y_j^{t',0}$, which is mutually *conditionally* independent, as such, the sum within the exponential can be decomposed into a product of expectations. The second critical step is in (4.21), where it is important to realize that the σ-algebras generated by $\left\{y_j^{t',0}\right\}_{t'=1}^{t}$, and by $\{\lambda_i^{t'}\}_{t'=1}^{t}$, conditional on all decisions $\alpha_{i,j}^{t,s}$ and $\beta_i^{t,s}$, and initial conditions, coincide. This is because the former is a linear function of the latter, as a result of the distributive decision rule (as illustrated in Figure 4.2). As such, this allows the subsequent re-representation of the dynamics of the inflow in the server, in terms of the dynamics of the queue.

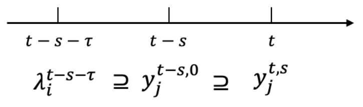

$t-s-\tau$
$t-s$
t
$\lambda_i^{t-s-\tau} \supseteq y_j^{t-s,0} \supseteq y_j^{t,s}$

Figure 4.2: The timeline of the allocation process

For the first term in (4.22) we have:

$$\begin{aligned}
&k\log\mathbb{E}\left[\exp\left[\sum_{s=0}^{t-1}\sum_{\tau=1}^{t-s-1}r_j^{t,s}(k\theta_{1,j}^t)\lambda_i^{t-s-\tau}\alpha_{i,j}^{t-s,\tau}/\beta_i^{t-s-\tau,0}\right]\right]\\
=&k\log\mathbb{E}\left[\exp\left[\sum_{t'=1}^{t-1}\sum_{\tau'=1}^{t'}r_j^{t,t'-\tau'}(k\theta_{1,j}^t)\lambda_i^{t-t'}\alpha_{i,j}^{t-t'+\tau',\tau'}/\beta_i^{t-t',0}\right]\right]\\
=&\sum_{t'=1}^{t-1}k\log\mathbb{E}\left[\exp\left[\lambda_i^{t-t'}\sum_{\tau'=1}^{t'}r_j^{t,t'-\tau'}(k\theta_{1,j}^t)\alpha_{i,j}^{t-t'+\tau',\tau'}/\beta_i^{t-t',0}\right]\right]\\
=&k\sum_{t'=1}^{t-1}g_i^{t'}\left(\sum_{\tau=1}^{t-t'}\frac{r_j^{t,t'-\tau}(k\theta_{1,j}^t)\alpha_{i,j}^{t-t'+\tau,\tau}}{\beta_i^{t-t',0}}\right)
\end{aligned}$$

$$=k\sum_{t'=1}^{t-1} g_i^{t'}\left(\sum_{\tau=1}^{t-t'} \frac{r_j^{t,t-t'-\tau}(k\theta_{1,j}^t)\alpha_{i,j}^{t'+\tau,\tau}}{\beta_i^{t',0}}\right).$$

Similar to Proposition 2, the expression for the reformulation (4.18) is also jointly convex in k, α and β for the same reasons – $\beta_i^{t',0}$ being a degree of freedom and $g_i^{t'}$ being convex, allowing its perspective to be jointly convex in the arguments. □ □

Theorem 4.2. *The model (4.15) can be solved via a sequence of convex programs.*

Proof. Proof of Theorem 4.2. Notice that each of the constraint reformulations in Proposition 2 and Theorem 4.1 is monotone in k. As such, we can solve (4.15) via interval bisection on k, where each sub-problem evaluates the feasibility of the constraints under that given k. By Proposition 2 and Theorem 4.1, these constraints are jointly convex in the decision variables α and β. □

This is a critical tractability result. Propositions 2, 3 and Theorem 4.1 indicate that every entropic constraint can be reformulated into just one convex constraint. In other words, (4.15) reduces to a convex program with $O(|\mathcal{I}||\mathcal{J}|T)$ constraints. This is interesting for a few reasons. One, it does not grow exponentially in time window T, and hence avoids the curse of dimensionality that is faced by many other methodologies that consider multi-period problems. Additionally, the complexity does not depend on the maximum queue length or the server capacity, as might one expect in a Dynamic Programming framework, due to the need to solve for the policy as a function of the state. The culmination of these factors allows our model to be far more scalable than other existing methodologies.

Solving for the log moment generating function term

While in Remark 4.1, we have discussed the situation where the family of distribution of the arrivals is known, in general, that is not the case.

Instead, what the decision-maker might have is a dataset of past observations on the arrivals $\mathcal{D}_i^t := \{\lambda_{i,l}^t\}_{l=1}^N$. In this case, we propose that the log-mgf terms be evaluated via Sample Average Approximations (SAA). More specifically, we evaluate

$$k \cdot g_i^{t-s}(\zeta\beta_i^{t,s}/k) \approx k \log\left(\sum_{l=1}^{N} \exp\left(\zeta\frac{\beta_i^{t,s}}{k}\lambda_{i,l}^{t-s}\right)\right).$$

At this point, this expression is still jointly convex in $\beta_i^{t,s}$ and k. Nonetheless, it is still not easy to optimize for $\beta_i^{t,s}$ (notice that it is less critical for k as we are already solving for it using interval bisection, so in each feasibility problem, k is fixed). As such, we propose a successive cutting plane algorithm to estimate this. Suppose the dataset is ordered: $\lambda_{i,1}^t \leq \ldots \leq \lambda_{i,N}^t$. Then we can initiate the problem by replacing this term by its two asymptotes, $\zeta\lambda_{i,1}^t\beta_i^{t,s}$ and $\zeta\lambda_{i,N}^t\beta_i^{t,s}$, where the direction depends on the sign of ζ. Subsequently, we declare a threshold for the accuracy of the estimation of this term. Whenever the feasibility problem is soluble and the threshold is not met, a cutting plane can be generated at the present solution of the problem, proceeding till either the threshold is satisfied or the problem is shown to be infeasible. This computational methodology as applied in the Pipeline Queue setting can be similarly referenced in Bandi, Loke (2018) and Jaillet et al. (2019).

Here, we make a quick remark that despite this, SAA can be a poor estimate for the log moment generating function. Indeed, if g_i^{t-s} corresponds to a Poisson distribution, then in Remark 4.1, we know that $k \cdot g_i^{t-s}(\zeta\beta_i^{t,s}/k) = k(\exp(\zeta\beta_i^{t,s}/k) - 1)$, which cannot possibly be estimated by a function that is asymptotically linear. Hence, insofar as the distributional family is known, it should be used instead, and other convex solution methodologies appealed to, such as exponential conic solvers in this particular case.

4.3 Numerical experiments

In this section, we present the results of our numerical experiments. This is done in three parts. In the first subsection, we shall introduce the core features of our model applied on a simple network to build basic intuition about how our model works and to draw some quick insights. In the second stream, we look specifically at partially connected networks, and extend to the question of (flexible) capacity allocation. In the third subsection, we perform comparisons in performance of our model against some benchmarks.

4.3.1 Understanding the dynamics of our model

In this section, we first start with a simple network with just one demand node and three supply nodes, all of which can satisfy the demand. The structure is shown in Figure 4.3.

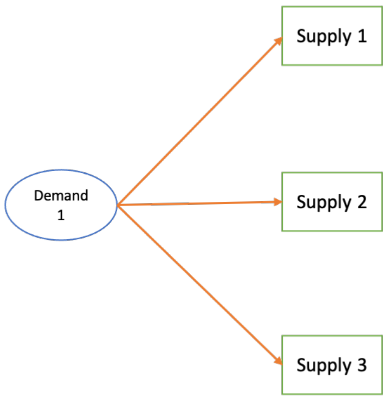


Figure 4.3: One demand node and three supply nodes

We consider a modelling horizon of $T = 12$. In the most basic set-up, we assume that the inflow of the arrivals at the demand node is time-homogeneous and follows a scaled Beta distribution $\Lambda_1^t \sim 30\beta(3,3), \forall t$. This amounts to a mean of 15 jobs per time period. We assume that the capacity of the three supply nodes are 20 each, and for now, we shall assume that their service distributions are homogeneous and follow an Geometric distribution with a probability of 1/3 completing a job at any point in time.

Finally, we assume that all nodes are initially empty at time period 0. In what ensues, we shall sequentially alter the service time distribution of the supply nodes, the initial conditions, the inflow pattern, and the capacity of the servers; and examine their impact on the optimal policy.

To obtain the optimal policy, we first generate data for the inflow distribution. This would help us to estimate the moment generating function of the arrivals using the SAA methodology explained at the end of §3. For the service completion probabilities $1 - q_j^{t,s}$, we used the true assumed distribution, which we shall vary in the subsequent analysis. From here, we solve the problem (4.15) where the choice of the flow balance constraint is taken as overloading commitment. The cost constraint is specified as an average waiting time constraint with average $C_i^t = 1.2^t$, for $t = 1, \dots, T$. The raising of the power to t is to simulate a discounting in time. We did not include the allocation constraint for the purposes of this modelling. To solve (4.15), the convex part, which arises wholly out of the log moment generating function term $g_i^t(\,\cdot\,)$ are evaluated in sample. We employ successive cutting plane algorithm to sequentially estimate the accuracy of the constraint till a fixed tolerance is met. On the overall, the model was solved on a MacBook Pro Early 2015 with 2.7 GHz Dual-Core Intel Core i5 processor, programmed with Gurobi in Python / Jupyter Notebook. On average, each instance of the model solved in around 90 seconds. Where we conduct simulations, we generate sample paths of data size 1000 and evaluate the performance of policies over this data set.

The effects of different service time distribution

In this section, we test the impact of the service time distributions of the supply nodes on the dispatch decisions. We shall conduct 4 experiments in this section:

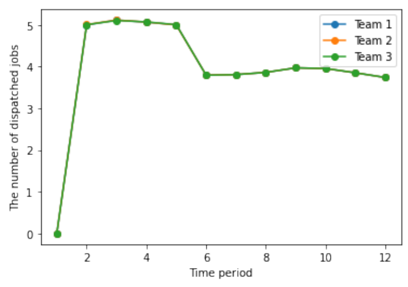
a) Geometric: Service distribution in all supply nodes are homogeneous and obey a geometric distribution with probability of 1/3 of job completion.

b) Deterministic (one-point): All supply nodes complete jobs in exactly 3 time periods.

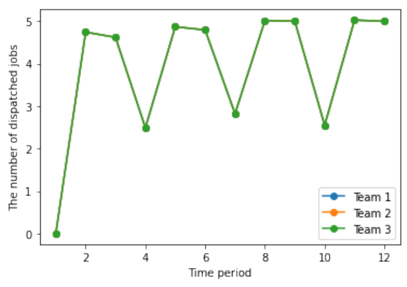
c) Two-point distribution: All supply nodes have a 50% chance of completing jobs in exactly 1 and 5 time periods.

d) Mixed: The three supply nodes are heterogeneous and have service distributions corresponding to each of the above distributions.

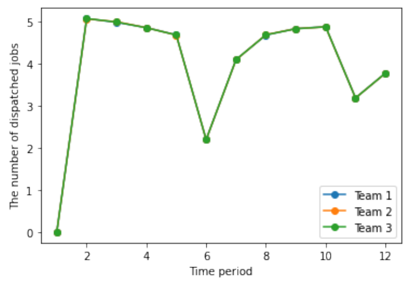
Most importantly, the average service time in all these cases are identically 3. In Figure 4.4, we plot the expected number of jobs that are dispatched to each team for each time period, according to the optimal policy that is solved by our model.



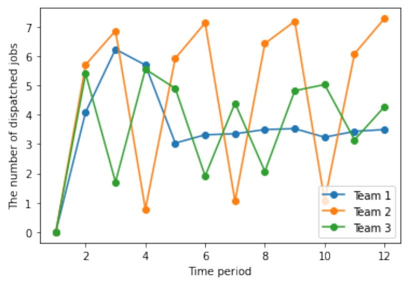
(a) Geometric



(b) Deterministic (one-point)



(c) Two-point



(d) Mixed

Figure 4.4: The dispatch decisions under different service time distributions

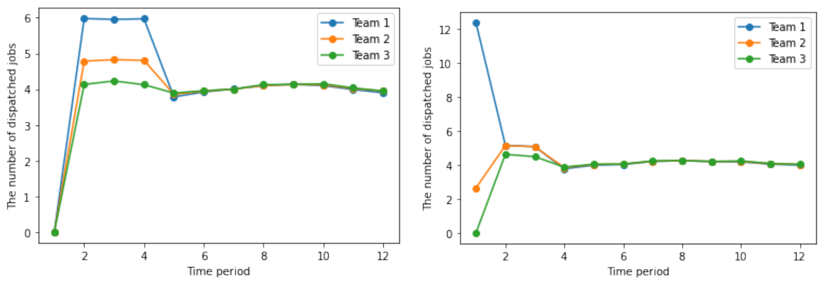
Not surprisingly, we find the number of jobs allocated to each team is identical in Figures 4.4 (a)–(c)). This is because the teams are homogeneous. Nonetheless, the allocation policy differs with service time distribution. Specifically, the geometric distribution (Figure 4.4 (a)) records a constant policy. Because the probability of job completion is the same no matter how long the job has been processed for, a constant policy ensures a constant inflow of jobs into the server, which balances with the rate of job completion. There is nonetheless a dip in the policy between periods 5 and 6, corresponding to the empty initial conditions of the supply nodes so more jobs are served at the start without waiting. For the deterministic case (Figure 4.4 (b)), the periodic nature of the allocation policy is due to the periodic nature of job completion. In fact, the period of 3 coincides with the service time. For the two-point distribution (Figure 4.4 (c)), a much more chaotic behaviour is observed. There is a loose 5-period cycle, which may potentially coincide with 5 being the longest service time required for job completion. Most notably, these three optimal policies differ significantly despite possessing the same mean service time, indicating that mean service time alone is insufficient to characterize the optimal policy.

In the last instance (Figure 4.4 (d)), where the supply nodes are heterogeneous, we observe that each node retains its previous structure – the one with geometric distribution eventually stabilizes into a constant policy; the one with the deterministic service time retains its periodic nature. Despite retaining its structure, however, the optimal policy for each of the distributions take different parameters from before – the constant policy now stabilizes at 3 jobs per period instead of 4; the periodic allocation fluctuates between 1 and 7 jobs now, instead of 2 and 5 previously.

In summary, we have identified that (i) service distributions significantly alter the structure of the optimal policy and considering mean service time is insufficient, (ii) the optimal policy in some cases can be chaotic, and (iii) with server heterogeneity, it is possible for the structure of the optimal policy to be preserved, but the actual values taken will depend on the other supply nodes. These insights are interesting for a few reasons. First, this identifies the existence of optimal policies that do not converge (for example, the periodic policy), and hence can never be obtained under certain methodologies, *e.g.* steady-state or state-independent policies. Second, there are complex interactions between service time distributions. These interactions are potentially complicated to the point that it may be difficult to analytically solve for, and might only be gleaned through the solution of an optimization problem.

The effects of different initial conditions

In this experiment, we examine the impact of different initial conditions on the optimal policy. In particular, we make the supply nodes homogeneous (all possessing a geometric distribution with the same probability). However, we initiate the system with the three supply nodes being fully occupied, partially occupied and fully unoccupied at time 0 respectively. In Figure 4.5 below, we plot the optimal policy over time for two instances, where in Figure 4.5 (a), the demand nodes are initiated empty, and in Figure 4.5 (b), the demand nodes have existing jobs (15) in the queue.



(a) Queue is empty at period 0. (b) Queue is partially packed at period 0.

Figure 4.5: Different initial conditions with geometric service time distribution

In this case, we can see two distinct epochs, a transient epoch and a steady state epoch. In the steady state, the shape of the optimal policy from Figure 4.4 is recovered. In the transient state, the allocation of the jobs depends both on the availability of spare capacity in the supply nodes, which is evident in both instances, and the current accumulation of jobs in the queue at time 0, affecting the urgency of the jobs to be dispatched in the first few time periods.

In the next experiment, we change the service distribution in the supply nodes to one with deterministic service times. The initial conditions that we prescribe on the demand and supply nodes are as follows: First, the demand nodes is initiated with (15) jobs waiting to be served. Second, for the three supply nodes, they are fully occupied at time 0. However, the time spent by the present jobs in each of the nodes are 0, 1, 2 periods respectively. The results are presented below in Figure 4.6. What we see is that the optimal policy reacts to the initial distribution, by preserving its periodic pattern, but are staggered and out-of-phase from each other, as they are reacting to the expected time of job completion of the existing jobs.

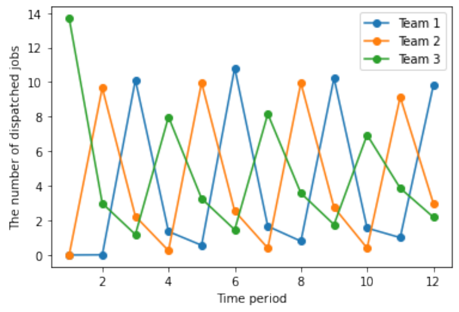


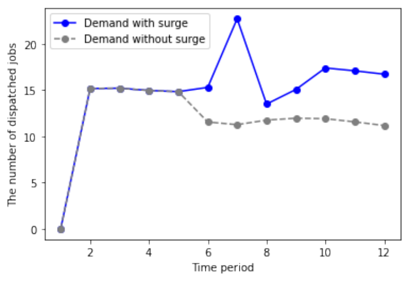
Figure 4.6: Different initial conditions with deterministic service time distribution

In summary, what we have seen here is that the initial conditions can have both a short-term impact on the transient nature of the optimal policy (as in the first case) and long-term impact on the optimal policy, such as its structure (as in the second case). As such, models that fundamentally do not react to the transient nature of the problem or are unable to adapt to the changing initial conditions may potentially be sub-optimal.

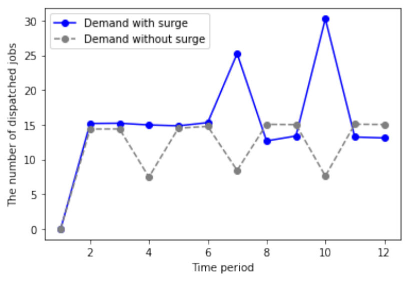
The effects of time-varying demand

In this experiment, we examine the impact of time-varying demand on the optimal policy. Specifically, we return the model to its initial set-up (geometric service distribution, zero initial conditions), but consider a non-time-homogeneous arrival distribution at the demand nodes.

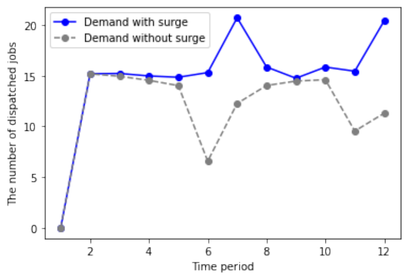
In the first case, we examine a surge in the demand. Specifically, we model a surge at time period 6, amount to increasing the expected number of cases by three times. The arrival then falls back to the usual rate after the surge. Notice that as the arrival distribution is a model input, information of the surge is privy to the model. We present the results in Figure 4.7. In the Figure, we took the results from Figure 4.4 and aggregated the total jobs across the three servers into the grey broken line. This is then compared against the total jobs under the model with the surge (blue line). Detailed breakdowns are available in Appendix C.1 (Figure C.1).



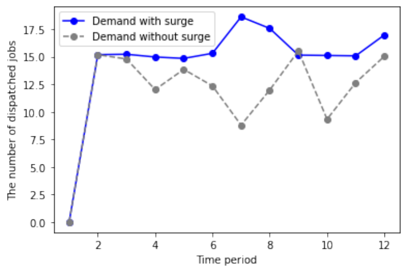
(a) Geometric



(b) Deterministic (one-point)



(c) Two-point

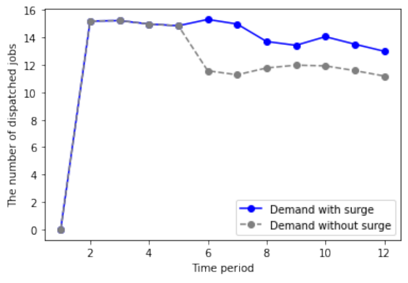


(d) Mixed

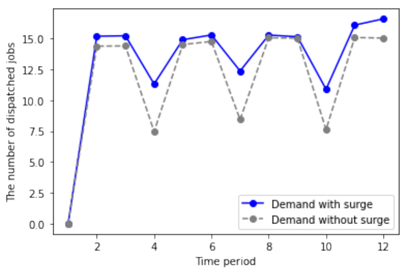
Figure 4.7: The dispatch decision where there is a surge of demand at period 6

We observe that where previously the model has avoided dispatching the maximum number of jobs into the supply nodes, which is 15 jobs per time period after around period 4 or 5, the model now does so, in anticipation of the surge, which will kick in at time period 6 and effected at time period 7. Moreover, in some cases, features of the original optimal policy are still seen, for example, the deterministic case having a 3 period pattern after the surge, and to a less extent, for the two-point distribution.

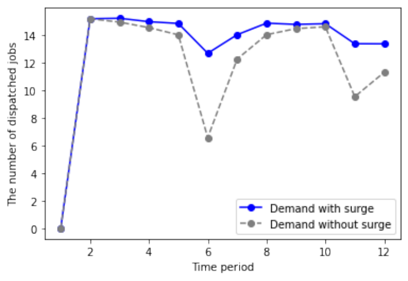
In the next case, we replicate the above result, except that we consider an arrival pattern where there is a significant increase in arrivals consistently after period 6. Specifically, the average demand is raised by 1.25 of the



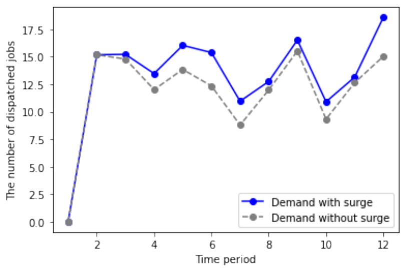
(a) Geometric



(b) Deterministic (one-point)



(c) Two-point



(d) Mixed

Figure 4.8: The dispatch decision where there is a lift of demand from period 6

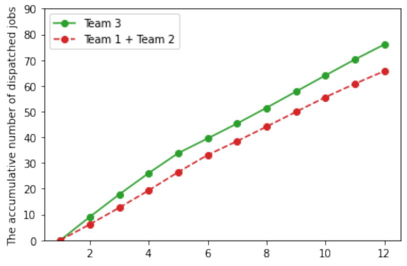
original mean of 15 from period 6 onwards. The same figure is drawn in Figure 4.8 where the grey broken line is the same comparison against the initial policy in Figure 4.4. Once again, detailed breakdowns are available in Appendix C.1 (Figure C.2). Here, the structure of the original optimal policy is preserved more strongly, with only some uplifting. This is most salient in Figures 4.8(b) and 4.8(d). Moreover, this uplifting occurs far earlier than the expected time of the increase of demand, as earlier as period 3.

In conclusion, the above identifies that our model is anticipatory of expected surges or increased demand. However, the way this takes shape in the optimal policy is not *a priori* known.

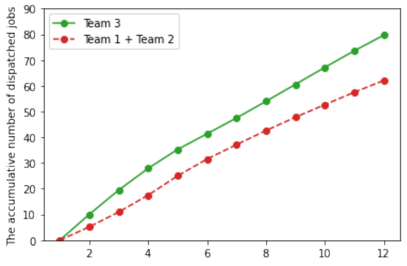
The effects of different supply capacities

In this last experiment on the basic model, we alter the capacities in the supply nodes. Here, we increase the capacity of the final supply node 40, specifically, where $\mathcal{K}_1 = 20$, $\mathcal{K}_2 = 20$, $\mathcal{K}_3 = 40$. Notice that $\mathcal{K}_1 + \mathcal{K}_2 = \mathcal{K}_3$.

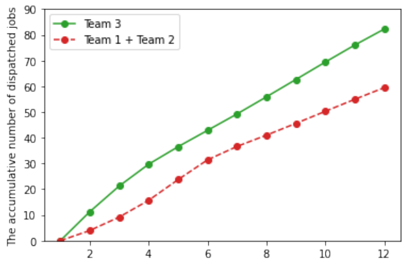
If we consider only expectations, then the allocation of jobs to the first two nodes should sum to the third. In what ensues, we will vary $\mathcal{K}_1$ and $\mathcal{K}_2$ but keep their sum the same. The detailed plots of the optimal policy is deferred to Appendix C.1 (Figure C.3). Instead, we plot the cumulative number of jobs that have been allocated to the supply nodes in Figure 4.9 below. Jobs to the first two nodes are summed as a comparison against the third.



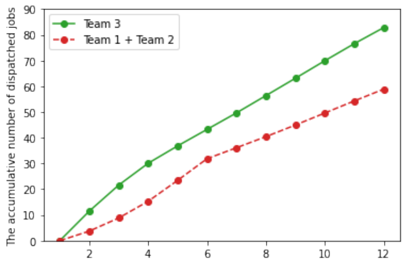
(a) $\mathcal{K}_1 = 5$, $\mathcal{K}_2 = 35$.



(b) $\mathcal{K}_1 = 10$, $\mathcal{K}_2 = 30$.



(c) $\mathcal{K}_1 = 15$, $\mathcal{K}_2 = 25$.



(d) $\mathcal{K}_1 = 20$, $\mathcal{K}_2 = 20$.

Figure 4.9: Capacity pooling

The primary observation is the risk-pooling effect, where the total allocated jobs to the third supply node is larger than the sum of allocated jobs to the first two nodes, despite having the same total capacity. Second, this risk-pooling effect increases as the differential in the capacities between the first and second supply nodes decreases. This is due to the convex nature of the risk measure considered – that is, the total amount of risk that can be borne by a pair of nodes with capacity $\mathcal{K}_1 = 15$ and $\mathcal{K}_2 = 25$ is larger than that with capacity $\mathcal{K}_1 = \mathcal{K}_2 = 20$.

Summary

What we have seen in the above four experiments is that the service distribution, the initial conditions, the arrival distribution and the capacity can change the structure of the optimal policy in very dynamic ways, that are intuitive, but difficult to ascertain theoretically. In the new context of demand and supply uncertainty, such problems can be very difficult to solve, especially in practical settings. Failure to consider any of these aspects could easily lead to sub-optimality. In this regard, this also justifies why we constructed our model to encompass all these features.

4.3.2 Insights on partially connected networks

In this subsection, we consider bipartite but incomplete graphs, in order to study the implications of a network that is not fully connected. We hope to be able to illustrate insights about capacity planning and allocation policies within these settings, so as to advise on how networks are designed.

Managing spare capacity

We consider a graph with two demand nodes and three supply nodes, as in Figure 4.10 below. Such a graph represents two dedicated supply nodes, and one node (Supply 2 in the Figure) that represents a spare capacity that can flexibly serve either of the demand pools. Such a setting may be more reasonably seen in service operations, or healthcare operations, and much has been studied about them (Yan et al., 2018). In many of these settings, there may be higher costs for utilizing the spare capacity, but we shall not take that into consideration in this experiment. As before, we return to the geometric service time and scaled beta arrivals setting.

In what ensues, we will slowly increase the capacity of the spare node (Supply 2) while keeping the total capacity across the three supply nodes unchanged. In Figure 4.11, we plot on the left the largest queuing cost

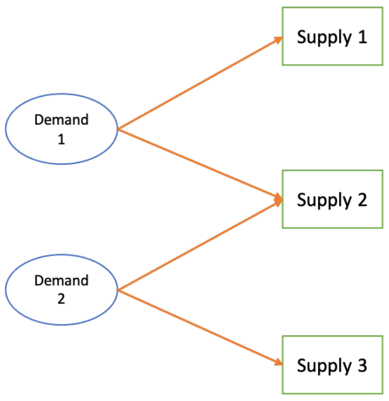
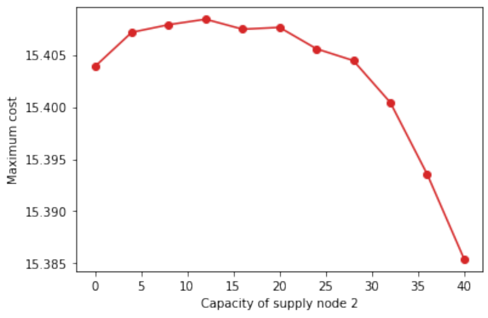
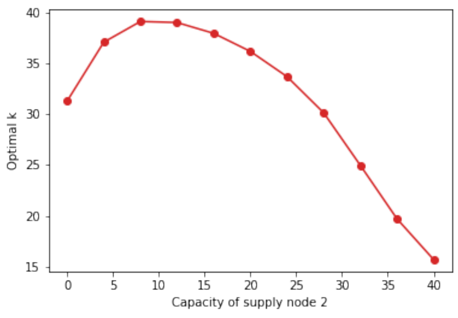


Figure 4.10: Network with two demand and three supply nodes

observed at any one point in time in any demand node, which is aggregated over $1,000$ simulations, and on the left, we plot the risk level k of the optimal policy. The two charts should relate to each other, which is a result of Proposition 1.



(a) Queuing cost



(b) Optimal risk level

Figure 4.11: Different capacity of supply node 2

Surprisingly, the chart does not yield a monotonically decreasing trend. As one might expect, the greater the spare capacity, the more flexible the allocation of jobs may be and hence the lower the risk. However, what is observed here is that the risk first increases, before it starts to decrease. The reasoning behind this might be that when the spare capacity is small, it cannot reasonably handle a large number of jobs. As the total capacity is fixed, this effectively is a diversion of capacity away from both of the dedicated supply nodes, thus reducing their ability to serve their respective demand. In other words, there is a critical point where the gains from possessing spare capacity is sizeable compared to the loss in service at the dedicated nodes.

Load sharing

Let us look at another structure as shown in Figure 4.12. Here, with three demand nodes and only two supply nodes, there will exist a non-trivial allocation issue, especially for jobs from demand source 2. Specifically, this models the setting where both supply nodes have a dedicated demand that they need to serve, and a common work pool arising out of demand source 2 that they need to load share. In this context, we seek the best allocation of capacity between the servers. To this end, we stick to the geometric service time setting with scaled beta arrivals. However, we let the arrivals be such that $\lambda_1 = \lambda_3 \sim n\beta(3,3)$, and $\lambda_2 \sim (30-2n)\beta(3,3)$. Here, altering n would not change the overall expected demand, but will skew the relative ratio of dedicated to shared work load. We further assume $\mathcal{K}_1 + \mathcal{K}_2 = 60$.

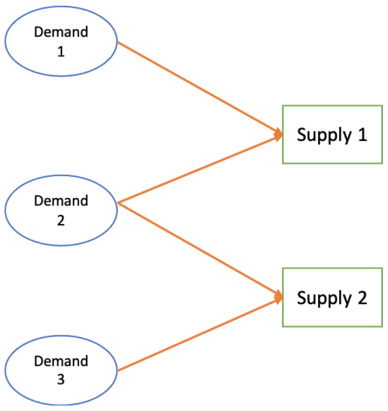
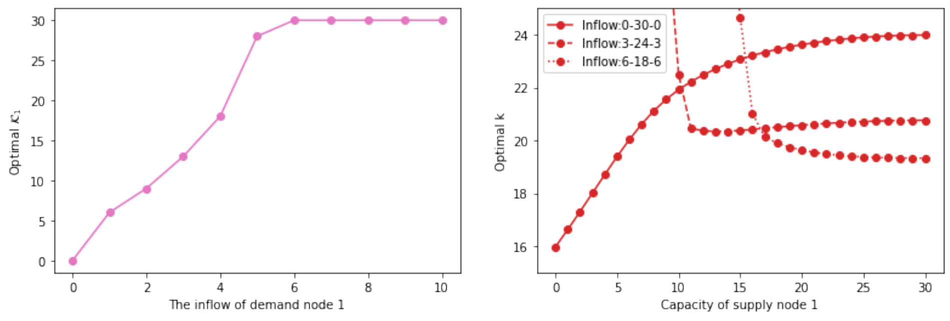


Figure 4.12: Network with 3 demand and 2 supply nodes

Figure 4.13(a) shows the optimal capacity of supply node 1, as demands 1 and 3, *i.e.* the dedicated jobs, are increased. By definition, the optimal capacity of supply node 2 is 60 less this number. Here, the optimal capacity allocation between supply nodes is distributed more and more even as the expected dedicate jobs increase. Figure 4.13(b) examines more closely how the optimal capacity is obtained in Figure 4.13(a), by examining how the level of risk k against different choices of capacity for three specific configurations of inflow, $n = 0, 3$ and 6. When $n = 0$, risk k is monotone, as splitting the capacity between two servers decreases the risk-pooling effect. When $n = 3$, k first decreases and then increases with capacity; here, the trade-off between having sufficient bandwidth to deal with its own dedicated jobs and enough risk-pooling to handle the shared jobs is evident. Eventually the latter overwhelms the former and this is what is observed for $n = 6$.



(a) Optimal capacity of supply node 1

(b) Optimal k for different inflows

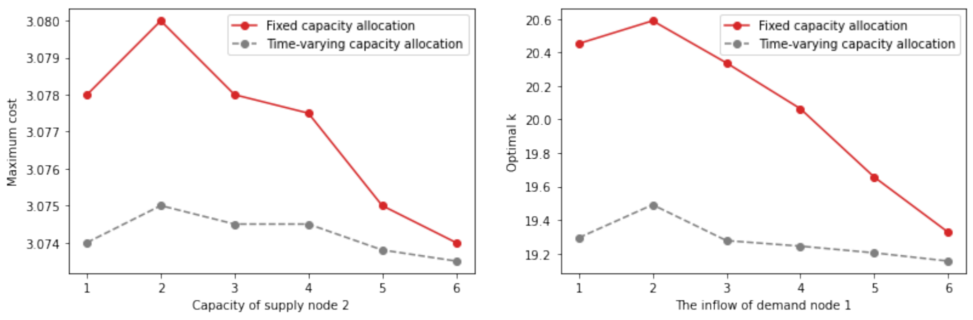
Figure 4.13

Taking this experiment further, we now allow the capacity allocation to change with time, that is, not just having $\mathcal{K}_1$ and $\mathcal{K}_2$ as decision variables, but allowing $\mathcal{K}_1^t, \mathcal{K}_2^t$ to both be decided at every time period with constraint $\mathcal{K}_1^t + \mathcal{K}_2^t = 60$. Figure 4.14 plots the savings obtained from doing so, in terms of the lower risk levels, which translates at the end of the day to shorter waiting times.

Further work we did on network connectivity, for the sake of brevity, is deferred to Appendix C.1.

4.3.3 Comparisons

In this subsection, we examine the comparisons we performed against other policies. First, we conducted a comparison against the static decision rule, under the same Pipeline Queue framework. This is done for two purposes.



(a) Queuing cost (b) Optimal risk level

Figure 4.14: Maximum cost and optimal k for fixed and time-varying capacity allocation

One, it illustrates the gains obtained from having an adaptive policy. Two, works in the literature, such as Tang et al. (2020) and Zhou et al. (2021), primarily utilize the static decision policy and already perform substantially better than the policies which they compared against. As such, the static decision rule is a reasonable benchmark for comparison. For the sake of brevity, we have placed the derivation of the static model, the set-up of the comparison and detailed discussion of the comparison in Appendix C.3.

To compare the performance of the static decision with the adaptive decision rules, we return to the original simple network of Figure 4.3, under geometric service times. We summarize the comparison in Table 4.2. For actual cumulative histograms, please refer to Table C.1 in Appendix C.3. In terms of the cost, the adjusted static model experiences a higher probability of violation and level of violation as evidenced by the median and 90th percentile violations than both the vanilla static model and our proposed adaptive model. Between our adaptive model and the vanilla static model, while they appear to have similar probabilities of constraint violation, the extent of the violation is on different scales, *i.e.* if violations occur, the latter will experience violations of around $30 - 40$ times larger violations. Moreover, while the vanilla static model appears to have better performance in terms of cost minimization than its adjusted static counterpart, the compensation it pays is in terms of the violation of the 'no-underflow constraint', which is reflected as the auxiliary constraint in the Table. In fact, around two-thirds of the time the model dictates dispatching more jobs than there are available! The adjusted static model does not violate this constraint *in simulation*. For the adaptive model, this is guaranteed, by design.

Metric		Adaptive	Vanilla Static	Adjusted Static
Auxiliary	$\mathbb{P}$[violation]	–	65.5%	0
constraint	Median violation	–	0.908	0
violation	90th percentile	–	0.908	0
Cost	$\mathbb{P}$[violation]	40.8%	45.8%	90.1%
constraint	Median violation	0.158	5.008	6.677
violation	90th percentile	0.338	12.888	13.835

Table 4.2: Comparison between adaptive and static decision rules

In summary, our proposed fully adaptive model enjoys the best of both worlds. It is able to achieve low violations in the cost constraints, while averting the unsavoury potential outcome of dispatching more jobs than there are available, which arguably is the Achilles' Heel of the static model.

Comparison against a fluid model

In what ensues we shall compare the performance of our model against a state-of-the-art model in the literature. Specifically, we examine Chan et al. (2021), where the authors study a capacity allocation problem. For brevity, we shall refer to their model as the 'benchmark model' from here on. The setting is posed as follows: Suppose there are I different types of jobs, and correspondingly, there are $J = I$ teams of workers who are to be assigned to work on these jobs. Within a finite time horizon of T, the decision-maker can reallocate the workers between the teams at fixed time points (shifts) $t = \ell\tau$ for $\ell = 1, 2, \ldots$, which form the decision variables, so as to minimize the total queue length in all of the queues over the horizon. To solve this problem, the authors considered Poisson arrivals and Exponential service times, and constructed a deterministic fluid approximation for the problem. As such, given a capacity policy, the dynamics between every decision point $\ell\tau$ and $(\ell + 1)\tau$ reduces to an ordinary differential equation, which they provide explicit solutions for. Based on this, the authors are able to compute the state and the cost at every decision point $\ell\tau$ as a function of the previous state and the capacity policy. This reduces to a dynamic programming formulation over T where each time step is each decision point $\ell\tau$. The authors showed that their proposed solution is asymptotically optimal, that is, the deterministic fluid approximation and its proposed policy is optimal for the original stochastic problem.

Here, we set up the comparison as follows: We considered $T = 12$, with shift lengths $\tau = 3$. For simplicity, we considered $I = J = 2$ types of jobs, with a total capacity of 60 to be allocated between the two teams. The arrival rates for the jobs were both fixed at 10 per period and their average service times were 3.3 and 2.5 periods. Under our model, the above setting would be modelled by a two demand and server complexes representing two dedicated queues and servers (see Figure 4.15), but where the sum of their capacities is constrained under the total capacity: $\mathcal{K}_1 + \mathcal{K}_2 \leq 60$. We only use the capacity decisions when testing our model. For further details about how the parameters of the constraints and the tightening θ-parameters were fixed, please refer to our more detailed write-up in Appendix C.3. As before, to compute the simulated results, we generated $1,000$ sample paths for the system and simulated the policies over these sample paths. The models are solved over a rolling horizon basis. Note that this does not change the solution of the the benchmark model as it is the solution of a dynamic program.

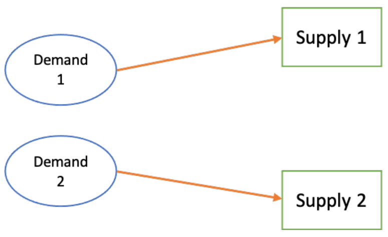


Figure 4.15: Network with 2 demand and 2 supply nodes

In the Table 4.3 below, we present the comparative results between our model and the benchmark under two situations. In the first case, we use the original setting of Poisson arrivals and Exponential service times. In the second case, we keep the mean arrival rate and service time the same, but change both of their distributions to two-point distributions. Since the benchmark model uses only the parameters of mean arrival rate and mean service time, the policy obtained is the same.

	Queue length	Original setting		Two-point distribution	
		Our model	Benchmark	Our model	Benchmark
Queue 1	Average	11.51(−21%)	14.59	25.54(−14%)	29.73
	90th percentile	30(−19%)	37	56(−26%)	76
Queue 2	Average	8.03(+16%)	6.94	9.64(−6%)	10.29
	90th percentile	24(+20%)	20	25(−7%)	27
Total	Average	19.54(−9%)	21.54	35.18(−12%)	40.02
	90th percentile	47(−10%)	52	76(−15%)	89

Table 4.3: Comparison between our model and benchmark

We noticed that our model reduces overall queue lengths by around 9% in the original setting. This does, however, come at the cost of having longer queue lengths in one of the queues. The reason why our model outperforms in this case is likely due to the spreading of the risk out between the two queues. In the benchmark model, only total service time is computed, and this might have an effect of generating longer service times in one queue over the other. Moreover, our model also uses information about the present delay, *i.e.* how long the job has waited in the queue and more critically, how long the job has already been served for in the server. Instead, the benchmark model only uses information about the number of jobs present in the queue. The reduction is larger (12%) when the Poisson arrival and Exponential service time distribution assumptions are not satisfied. In this case, we even see shortening of queue lengths in both queues. This indicates that the distributional assumptions were critical to the benchmark model. However, it is immediately clear if the benchmark model can be easily extended to general distributions, since it does depend on critical statistics like the mean arrival rate and service time.

4.4 Conclusion

In this paper, we address the resource allocation problem where the uncertainties come from demand arrivals, service times and service availability.

Here, we proposed a tractable and adaptive model for the resource allocation problem posed in this setting. Our model is an adaptation of the earlier Pipeline Queue paradigm (Bandi, Loke, 2018), and involves new techniques and derivations not seen previously. In this setting, the service distribution, the initial conditions, the arrival distribution and the capacity can all be very dynamically evolving. Our numerical simulations further illustrate that neglecting them would easily lead to sub-optimality.

In future work, we intend to pursue applications of this framework on actual problem settings, to better understand itself effectiveness in the face of real data. Theoretically, we are interested to examine two aspects of our model. First, the definition of the dynamics permits the modelling of drop-outs in the demand nodes via a similar Binomial distributed dynamics as what happens in the servers. The entropic constraints should still be reformulable and this should not alter the tractability and the convex nature of the problem, though there might be challenges introduced into how to proceed with the solution of the convex problem. Second, the relatively simpler structure of our model *vis-à-vis* the Pipeline Queue framework allows an easier analysis of the duality of the model, which might be able to point to the shadow costs of capacity decisions. This might help in explaining the structure of the risk-based results that we saw in the numerical section, involving the capacity of the system.

Chapter 5

Conclusion

This thesis investigates the flexible resource allocation in service and production systems. Chapters 2 and 3 study the stochastic bucket brigades with *preemptible* and *non-preemptible* work content respectively. Chapter 4 studies the resource allocation problem in a service system under both supply and demand uncertainty.

The literature of bucket brigades with stochastic service times is very limited. To the best of our understanding, Bartholdi III et al. (2001) is the only paper that analytically studies bucket brigades on discrete work stations with stochastic service times. In Chapters 2 and 3, we extend the work of Bartholdi III et al. (2001) by assuming that the work speed of each worker at each station on a job depends not only on the worker, but also on the station and the job. We consider a bucket brigade line with J stations and I workers. We assume the time duration for each worker to finish a job at each station is exponentially distributed with a rate that depends on the station's expected work content and the work speed of the worker. By observing the Markov property of the hand-off stations (reset vector) and analyzing the transition from one hand-off station vector (reset vector) to the next, we are able to derive the system's average throughput. We prove that if the work speeds of the workers are independent of the jobs, then the probability distribution of the hand-off station vector (reset vector) will converge to a unique stationary distribution as the number of jobs approaches infinity. The throughput will also converge to a value that depends on the stationary distribution.

Chapter 4 addresses the online resource allocation problem where the uncertainties come from demand arrivals, service times and service availability. We decide upon the resource allocation as well as the dynamic capacity allocation. This model has broad applications in ride-sharing, service scheduling and last-mile delivery. We introduce an *distributive decision rule*, which decides on the proportion of jobs awaiting dispatch to each of the possible resource supply pools. Our model has a convex reformulation that can be solved efficiently. Our model is an adaptation of the earlier Pipeline Queue paradigm (Bandi, Loke, 2018), and involves new techniques and derivations not seen previously. In this setting, the service distribution, the initial conditions, the arrival distribution and the capacity can all be very dynamically evolving. Our numerical simulations further illustrate that neglecting them would easily lead to sub-optimality. The benchmarking justifies the adaptivity of the proposed distributive decision rule and shows the robustness of our model to different settings.

Appendix A

Appendix for Chapter 2

A.1 Proof of Theorem 1

The expected makespan for K jobs can be obtained by summing up $E\left[T^{(k)} - T^{(k-1)}\right]$ from $k = 1$ to $k = K$:

$$\begin{aligned} E\left[T^{(K)}\right] &= \sum_{k=1}^{K} E\left[T^{(k)} - T^{(k-1)}\right] \\ &= \sum_{k=1}^{K} \boldsymbol{\pi}^{(k-1)} \boldsymbol{z}^{(k)} \\ &= \sum_{k=1}^{K} \boldsymbol{\pi}^{(0)} \dots \boldsymbol{P}^{(k-1)} \boldsymbol{z}^{(k)}. \end{aligned}$$

So we have:

$$\rho(K) = K \Big/ E\left[T^{(K)}\right] = K \left/ \sum_{k=1}^{K} \boldsymbol{\pi}^{(0)} \boldsymbol{P}^{(1)} \cdots \boldsymbol{P}^{(k-1)} \boldsymbol{z}^{(k)} \right. .$$

$\square$

A.2 Proof of Theorem 2

If the processing rates of the workers are independent of the jobs, then from Lemma 1, $p^{(k)}_{\boldsymbol{h},\boldsymbol{h}'} = p_{\boldsymbol{h},\boldsymbol{h}'}$, for $\boldsymbol{h}, \boldsymbol{h}' \in \mathcal{H}$. This implies that $\{\boldsymbol{H}^{(k)}, k = 0, 1, 2, \ldots\}$ is an irreducible aperiodic homogeneous Markov chain with a finite number of states. Since there is only a finite number of states and all of them communicate, all the states are positive recurrent. Recall that we set $\boldsymbol{\pi}^{(0)} = (1, 0, ..., 0)$. According to the definition of $\pi^{(k)}_n$, we have

$$\begin{aligned}\lim_{k\to\infty} \pi^{(k)}_n &= \lim_{k\to\infty} Pr\left\{\boldsymbol{H}^{(k)} = \boldsymbol{h}^n\right\} \\ &= \lim_{k\to\infty} \sum_{m=1}^{|\mathcal{H}|} \pi^{(0)}_n Pr\left\{\boldsymbol{H}^{(k)} = \boldsymbol{h}^n | \boldsymbol{H}^{(0)} = \boldsymbol{h}^m\right\} \\ &= \lim_{k\to\infty} Pr\left\{\boldsymbol{H}^{(k)} = \boldsymbol{h}^n | \boldsymbol{H}^{(0)} = \boldsymbol{h}^1\right\} \\ &= \pi_n\end{aligned}$$

The last equality is due to Theorem 4.3.3 of Ross et al. (1996), which also guarantees that $\boldsymbol{\pi} = \left(\pi_1, \ldots, \pi_{|\mathcal{H}|}\right)$ is the only stationary distribution of the Markov chain. This stationary distribution can be obtained by solving the equations $\boldsymbol{\pi}\boldsymbol{P} = \boldsymbol{\pi}$ and $\boldsymbol{\pi}\boldsymbol{e} = 1$.

Thus, we have

$$\begin{aligned}\lim_{K\to\infty} \frac{1}{\rho(K)} &= \lim_{K\to\infty} \frac{E\left[T^{(K)}\right]}{K} \\ &= \lim_{K\to\infty} \frac{\sum_{k=1}^{K} E\left[T^{(k)} - T^{(k-1)}\right]}{K} \\ &= \lim_{K\to\infty} \frac{\sum_{k=1}^{K} \boldsymbol{\pi}^{(k-1)}\boldsymbol{z}}{K} \\ &= \boldsymbol{\pi}\boldsymbol{z}\end{aligned}$$

The last equality is implied by Section 3 of Chapter 1 of Lyusternik, Yanpol'Skii (1965). Therefore, $\lim_{K\to\infty} \rho(K) = 1/(\boldsymbol{\pi}\boldsymbol{z})$. □

A.3 Proof of Lemma 2

We denote the cumulative distribution function (CDF) of $Y(k)$ as $F^{(k)}(\,\cdot\,)$. Let $Y_n(k)$ denote the inter-completion time between the $(k-1)$st and the kth resets, conditioned on $\boldsymbol{H}^{(k-1)} = \boldsymbol{h}^n$. Let $F_n^{(k)}(\,\cdot\,)$ denote the CDF of $Y_n(k)$. Recall that $\boldsymbol{\pi}^{(k-1)} = \left(\pi_1^{(k-1)}, \ldots, \pi_{|\mathcal{H}|}^{(k-1)}\right)$ is the probability distribution of the $(k-1)$st hand-off station vector $\boldsymbol{H}^{(k-1)}$. The CDF of $Y(k)$ can be expressed as a mixture distribution $F^{(k)}(\,\cdot\,) = \sum_{n=1}^{|\mathcal{H}|} \pi_n^{(k-1)} F_n^{(k)}(\,\cdot\,)$. According to Proposition 5.2 of Ross (2014), the variance of $Y(k)$ can be obtained as follows:

$$\mathrm{Var}\left(Y(k)\right) = \sum_{n=1}^{|\mathcal{H}|} \pi_n^{(k-1)} \mathrm{Var}\left(Y_n(k)\right) + \sum_{n=1}^{|\mathcal{H}|} \pi_n^{(k-1)} \left(E\left[Y_n(k)\right]\right)^2 - \left(\sum_{n=1}^{|\mathcal{H}|} \pi_n^{(k-1)} E\left[Y_n(k)\right]\right)^2. \tag{A.1}$$

Note that $Y_n(k)$ is the sum of the processing times of worker I on job k from station h_{I-1}^n to station J. Since these processing times are independent exponential random variables, we have

$$E\left[Y_n(k)\right] = \sum_{j=h_{I-1}^n}^{J} E\left[Z_{I,j}^{(k)}\right] = \sum_{j=h_{I-1}^n}^{J} 1/\mu_{I,j}^{(k)}, \tag{A.2}$$

and

$$Var\left(Y_n(k)\right) = \sum_{j=h_{I-1}^n}^{J} Var\left(Z_{I,j}^{(k)}\right) = \sum_{j=h_{I-1}^n}^{J} \left(1/\mu_{I,j}^{(k)}\right)^2. \tag{A.3}$$

Substituting Equations (A.2) and (A.3) into Equation (A.1), we obtain Lemma 2. □

A.4 Proof of Lemma 3

We assume $v_1 + v_2 = c$, where c is a constant. We have

$$\begin{aligned} 1/\rho_\infty \;\; = \;\; & \big(v_1^7 s_2^2 s_3^5 + 4v_1^6 v_2 s_1 s_2^2 s_3^4 + 2v_1^6 v_2 s_1 s_2 s_3^5 + 2v_1^6 v_2 s_2^3 s_3^4 + 2v_1^4 v_2^3 s_1^3 s_2^2 s_3^2 + \\ & 3v_1^4 v_2^3 s_1^2 s_2^3 s_3^2 + 3v_1^4 v_2^4 s_1^2 s_2^2 s_3^3 + 6v_1^4 v_2^2 s_1^2 s_2 s_3^4 + v_1^4 v_2^2 s_1^2 s_3^5 + \end{aligned}$$

$$
\begin{aligned}
&2v_1^4v_2^2s_1s_2^4s_3^2 + 5v_1^4v_2^2s_1s_2^3s_3^3 + 4v_1^4v_2^2s_1s_2^2s_3^4 + v_1^4v_2^2s_2^5s_3^2 + \\
&2v_1^3v_2^3s_1^3s_3^4 + 2v_1^3v_2^3s_1^2s_2^4s_3 + 6v_1^3v_2^3s_1^2s_2^3s_3^2 + 8v_1^3v_2^3s_1^2s_2^2s_3^3 + \\
&2v_1^3v_2^3s_1^2s_2s_3^4 + 2v_1^3v_2^3s_1s_2^5s_3 + 2v_1^3v_2^3s_1s_2^4s_3^2 + v_1^2v_2^4s_1^5s_3^2 + \\
&2v_1^2v_2^4s_1^4s_2^2s_3 + 3v_1^2v_2^4s_1^3s_2^3s_3 + 3v_1^2v_2^4s_1^3s_2^2s_3^2 + 3v_1^2v_2^4s_1^3s_2s_3^3 + \\
&v_1^2v_2^4s_1^2s_2^5 + 4v_1^2v_2^4s_1^2s_2^4s_3 + 2v_1v_2^5s_1^5s_2s_3 + 2v_1v_2^5s_1^3s_2^4 + v_2^6s_1^5s_2^2\big) \, / \\
&[7v_2\left(v_1^3s_2s_3^2 + 2v_1^2v_2s_1s_2s_3 + v_1^2v_2s_1s_3^2 + v_1^2v_2s_2^2s_3 + v_1v_2^2s_1^2s_3 + \right. \\
&\left. v_1v_2^2s_1s_2^2 + v_1v_2^2s_1s_2s_3 + v_2^3s_1^2s_2\right)].
\end{aligned}
$$

Taking the derivative of $1/\rho_\infty$ with respect to v_1, we have

$$
\begin{aligned}
\frac{d(1/\rho_\infty)}{dv_1} \quad = \quad &\big(v_1^6s_2^2s_3^5 + 4v_1^5v_2s_1s_2^2s_3^4 + 2v_1^5v_2s_1s_2s_3^5 + 2v_1^5v_2s_2^3s_3^4 + 2v_1^4v_2^2s_1^3s_2^2s_3^2 + \\
&3v_1^4v_2^2s_1^2s_2^3s_3^2 + 3v_1^4v_2^2s_1^2s_2^2s_3^3 + 6v_1^4v_2^2s_1^2s_2s_3^4 + v_1^4v_2^2s_1^2s_3^5 + \\
&2v_1^4v_2^2s_1s_2^4s_3^2 + 5v_1^4v_2^2s_1s_2^3s_3^3 + 4v_1^4v_2^2s_1s_2^2s_3^4 + v_1^4v_2^2s_2^5s_3^2 + \\
&2v_1^3v_2^3s_1^4s_2s_3^2 + 2v_1^3v_2^3s_1^3s_2^3s_3 + 2v_1^3v_2^3s_1^3s_2^2s_3^2 + 2v_1^3v_2^3s_1^3s_2s_3^3 + \\
&2v_1^3v_2^3s_1^3s_3^4 + 2v_1^3v_2^3s_1^2s_2^4s_3 + 6v_1^3v_2^3s_1^2s_2^3s_3^2 + 8v_1^3v_2^3s_1^2s_2^2s_3^3 + \\
&2v_1^3v_2^3s_1^2s_2s_3^4 + 2v_1^3v_2^3s_1s_2^5s_3 + 2v_1^3v_2^3s_1s_2^4s_3^2 + v_1^2v_2^4s_1^5s_3^2 + \\
&2v_1^2v_2^4s_1^4s_2^2s_3 + 3v_1^2v_2^4s_1^3s_2^3s_3 + 3v_1^2v_2^4s_1^3s_2^2s_3^2 + 3v_1^2v_2^4s_1^3s_2s_3^3 + \\
&v_1^2v_2^4s_1^2s_2^5 + 4v_1^2v_2^4s_1^2s_2^4s_3 + 2v_1v_2^5s_1^5s_2s_3 + 2v_1v_2^5s_1^3s_2^4 + v_2^6s_1^5s_2^2\big) \, / \\
&[v_2^2\left(v_1^3s_2s_3^2 + 2v_1^2v_2s_1s_2s_3 + v_1^2v_2s_1s_3^2 + v_1^2v_2s_2^2s_3 + v_1v_2^2s_1^2s_3 + \right. \\
&\left. v_1v_2^2s_1s_2^2 + v_1v_2^2s_1s_2s_3 + v_2^3s_1^2s_2\right)^2].
\end{aligned}
$$

Since $d(1/\rho_\infty)/dv_1 > 0$, as v_2 increases (v_1 decreases), $1/\rho_\infty$ decreases, which means ρ_∞ increases. □

A.5 Proof of Lemma 4

Recall that $f_{i,j}^{(k)}$ denote the probability of worker i finishing his job at station j between the $(k-1)$st and kth resets. For convenience, we set $H_0^{(k)} = 1$, and $H_I^{(k)} = J+1$, for all k. We have

$$\begin{aligned} f_{i,j}^{(k)} &= Pr\left\{H_{i-1}^{(k-1)} \le j, H_i^{(k)} \ge j+1\right\} \\ &= Pr\left\{H_i^{(k)} \ge j+1\right\} - Pr\left\{H_{i-1}^{(k-1)} \ge j+1, H_i^{(k)} \ge j+1\right\} \\ &= Pr\left\{H_i^{(k)} \ge j+1\right\} - Pr\left\{H_{i-1}^{(k-1)} \ge j+1\right\} \\ &= \sum_{\substack{n=1,\dots,|\mathcal{H}| \\ h_i^n \ge j+1}} \pi_n^{(k)} - \sum_{\substack{n=1,\dots,|\mathcal{H}| \\ h_{i-1}^n \ge j+1}} \pi_n^{(k-1)}. \end{aligned} \tag{A.4}$$

Note that the third equality holds because $H_{i-1}^{(k-1)} \ge j+1$ implies $H_i^{(k)} \ge j+1$. Equation (2.8) in Lemma 4 is obtained by setting k to infinity in Equation (A.4). □

A.6 Proof of Lemma 5

Recall that $Z_{i,j}^{(k)}$ denotes the time duration for worker i to process job k at station j, which follows an exponential distribution with rate $\mu_{i,j}^{(k)} = \left(s_j/v_{i,j}^{(k)}\right)^{-1} = (s_j/v_{i,j})^{-1} = \mu_{i,j}$. Let $F(\cdot)$ denote the CDF of $Z_{i,j}^{(k)}$. Define $D_{i,j}^{(k)}$ as the time duration from the time point when worker i starts working at station j on job $k+I-i$ after the $(k-1)$st reset to the time point $T^{(k)}$. Between the $(k-1)$st and kth resets, we set $D_{i,j}^{(k)} = 0$ if worker i does not work at station j on job $k+I-i$. Let $G^{(k)}(\cdot)$ denote the CDF of $D_{i,j}^{(k)}$. Between the $(k-1)$st and kth resets, the time duration that worker i spends working at station j on job $k+I-i$ is $\min\left(Z_{i,j}^{(k+I-i)}, D_{i,j}^{(k)}\right)$. Condition on $D_{i,j}^{(k)} = t$, we have

$$\begin{aligned} E\left[\min\left(Z_{i,j}^{(k+I-i)}, D_{i,j}^{(k)}\right) | D_{i,j}^{(k)} = t\right] &= \int_0^t x\mu_{i,j} e^{-\mu_{i,j}x} dx + t\int_t^\infty \mu_{i,j} e^{-\mu_{i,j}x} dx \\ &= -te^{-\mu_{i,j}t} + \int_0^t e^{-\mu_{i,j}x} dx + te^{-\mu_{i,j}t} \\ &= \frac{1}{\mu_{i,j}} \int_0^t \mu_{i,j} e^{-\mu_{i,j}x} dx \\ &= \frac{1}{\mu_{i,j}} F(t). \end{aligned}$$

So, we have

$$\begin{aligned} E\left[\min\left(Z_{i,j}^{(k+I-i)}, D_{i,j}^{(k)}\right)\right] =& E_{D_{i,j}^{(k)}}\left[E\left[\min\left(Z_{i,j}^{(k+I-i)}, D_{i,j}^{(k)}\right)|D_{i,j}^{(k)}\right]\right] \\ =& E_{D_{i,j}^{(k)}}\left[F\left(D_{i,j}^{(k)}\right)\right] \\ =& \frac{1}{\mu_{i,j}}\int_0^\infty F\left(D_{i,j}^{(k)}\right)\, dG^{(k)}\left(D_{i,j}^{(k)}\right) \\ =& f_{i,j}^{(k)}/\mu_{i,j}, \end{aligned}$$

where $f_{i,j}^{(k)}$ denotes the probability of worker i finishing his job at station j between the $(k-1)$st and kth resets. The expected inter-completion time $E[Y(k)]$ between the $(k-1)$st and kth resets equals the sum of $\sum_{j=1}^{J} f_{i,j}^{(k)}/\mu_{i,j}$ and the expected blocked time of worker i between the $(k-1)$st and kth resets. Setting k to infinity, we obtain Equation (2.9). $\square$

A.7 Proof of Theorem 4

We can obtain the following equation

$$
\begin{aligned}
\sum_{i=1}^{I} \nu_i \left(Y_\infty - B_i\right) &= \sum_{i=1}^{I} \left(\frac{\sum_{j=1}^{J} f_{i,j} s_j}{\sum_{j=1}^{J} f_{i,j} s_j / v_{i,j}} \left(Y_\infty - B_i\right) \right) \\
&= \sum_{i=1}^{I} \left(\frac{\sum_{j=1}^{J} f_{i,j} s_j}{\sum_{j=1}^{J} f_{i,j} / \mu_{i,j}} \left(Y_\infty - B_i\right) \right) \\
&= \sum_{i=1}^{I} \left(\sum_{j=1}^{J} f_{i,j} s_j \right) \\
&= \sum_{j=1}^{J} \left(s_j \sum_{i=1}^{I} f_{i,j} \right) \\
&= \sum_{j=1}^{J} \left(s_j \sum_{i=1}^{I} \left(\sum_{\substack{n=1,\ldots,|\mathcal{H}| \\ h_i^n \geq j+1}} \pi_n - \sum_{\substack{n=1,\ldots,|\mathcal{H}| \\ h_{i-1}^n \geq j+1}} \pi_n \right) \right) \\
&= \sum_{j=1}^{J} \left(s_j \left(\sum_{\substack{n=1,\ldots,|\mathcal{H}| \\ h_I^n \geq j+1}} \pi_n - \sum_{\substack{n=1,\ldots,|\mathcal{H}| \\ h_0^n \geq j+1}} \pi_n \right) \right) \\
&= \sum_{j=1}^{J} \left(s_j \cdot 1 \right) \\
&= 1.
\end{aligned} \tag{A.5}
$$

The third equality is due to Equation (2.9). The fifth equality is due to Equation (2.8). Substituting $1 = \rho_\infty Y_\infty$ into Equation (A.5), we have Theorem 4. □

A.7.1 Procedure to determine $v_{i,j}^{(k)}$

1. Set $v_{i,j}^{(I-i+1)} = \underline{v}_{i,j}$, for $i = 1, \ldots, I$ and $j = 1, \ldots, J$. Set $\boldsymbol{\pi}^{(0)} = (1, 0, \ldots, 0)$.

2. For $k = 1, 2, \ldots, K-1$:

(a) Determine $\boldsymbol{P}^{(k)}$ with $v_{i,j}^{(I-i+k)}$ for $i = 1, \ldots, I$ and $j = 1, \ldots, J$. Compute $\boldsymbol{\pi}^{(k)} = \boldsymbol{\pi}^{(k-1)}\boldsymbol{P}^{(k)}$, and compute $f_{i,j}^{(k)}$ using Equation (2.7).

(b) Compute

$$v_{i,j}^{(I-i+k+1)} = \underline{v}_{i,j} + (\overline{v}_{i,j} - \underline{v}_{i,j})\left(1 - e^{-\alpha_{i,j}\sum_{k'=1}^{k} f_{i,j}^{(k')}}\right), \quad \text{(A.6)}$$

for $i = 1, \ldots, I$ and $j = 1, \ldots, J$.

A.8 Derivation of the transition probability $\hat{p}_{h,h'}^{(k)}$

Lemma 15. *The transition probability $\hat{p}_{h,h'}^{(k)}$ can be determined as*

$$\hat{p}_{h,h'}^{(k)} = \begin{cases} p_{j_2,h'}^{(k)}, & h < j_2, h' \leq j_1; \\ \sum_{j=j_1+1}^{J} p_{j_2,j}^{(k)}, & h < j_2, h' = j_1 + 1; \\ p_{h,h'}^{(k)}, & h \geq j_2, h' \leq j_1; \\ \sum_{j=j_1+1}^{J} p_{h,j}^{(k)}, & h \geq j_2, h' = j_1 + 1. \end{cases} \quad \text{(A.7)}$$

Proof. If $h < j_2$, then worker 2 is starved until he takes over the job of worker 1 at station j_2 in the $(k-1)$st reset. The probability of transitioning from $H^{(k-1)} = h$ to $H^{(k)} = h'$ is the same as the probability of transitioning from $H^{(k-1)} = j_2$ to $H^{(k)} = h'$, for $h' = 1, \ldots, j_1 + 1$. That is, $\hat{p}_{h,h'}^{(k)} = \hat{p}_{j_2,h'}^{(k)}$, for $h < j_2$. There are two cases:

1. If $h' \leq j_1$, then the transition probability of the partially cross-trained team is equivalent to that of the fully cross-trained team. That is, $\hat{p}_{h,h'}^{(k)} = \hat{p}_{j_2,h'}^{(k)} = p_{j_2,h'}^{(k)}$, for $h < j_2$ and $h' \leq j_1$.

2. If $h' = j_1 + 1$, then worker 1 is halted before the kth reset. This corresponds to the case where $h' > j_1$ for the fully cross-trained team. That is, $\hat{p}_{h,h'}^{(k)} = \hat{p}_{j_2,h'}^{(k)} = \sum_{j=j_1+1}^{J} p_{j_2,j}^{(k)}$, for $h < j_2$ and $h' = j_1 + 1$.

If $h \geq j_2$, then worker 2 takes over the job of worker 1 at station h in the $(k-1)$st reset. There are two cases:

1. If $h' \leq j_1$, then the transition probability of the partially cross-trained team is equivalent to that of the fully cross-trained team. That is, $\hat{p}^{(k)}_{h,h'} = p^{(k)}_{h,h'}$, for $h \geq j_2$ and $h' \leq j_1$.
2. If $h' = j_1 + 1$, then worker 1 is halted before the kth reset. This corresponds to the case where $h' > j_1$ for the fully cross-trained team. That is, $\hat{p}^{(k)}_{h,h'} = \sum_{j=j_1+1}^{J} p^{(k)}_{h,j}$, for $h \geq j_2$ and $h' = j_1 + 1$.

$\square$

Appendix B

Appendix for Chapter 3

B.1 Proof of Lemma 7

Condition 1 means that all the front $I-1$ workers are working at, blocked or waiting at the stations, while worker 1 does not wait for any worker because he is at the beginning of the line and has no predecessor. Condition 2 means that if the predecessor of worker i is working or blocked at a station, worker i should be working or blocked at the same station or a downstream station, or waiting for his predecessor. Condition 3 means that if the predecessor of worker i is waiting, worker i should be working or blocked at a downstream station, or waiting for his predecessor. □

B.2 Proof of Lemma 8

If worker $I-1$ is working at station 1 to $J-1$ when worker I completes job $k-1$ at the last station, worker I will walk back to wait for worker $I-1$ to finish job k at station h_{I-1} and then continue to work on job k from station $h_{I-1}+1$ until the last station. If worker $I-1$ is blocked in front of the last station when worker I completes job $k-1$ at the last station, worker I will take over job k from worker $I-1$ and completes the job at the last station. If worker $I-1$ is waiting when worker I completes job $k-1$ at the last station, there will be consecutive workers from worker $i+1$ to worker $I-1$ that wait for worker i. Worker I will also walk back to wait for worker i to finish job k at station h_i. Then worker i hands off the job to worker i, and worker i hands off the job to worker $i+1$ ..., until worker I take over the job and work on job k from station h_i+1 to station J. $\square$

B.3 Proof of Lemma 9

Assume the set of all transient states of a J-station I-worker line as $\mathcal{X}_t(I,J)$, and the set of all absorbing states as $\mathcal{X}_a(I,J)$. Then we have $\mathcal{X}(I,J) = \mathcal{X}_a(I,J) \bigcup \mathcal{X}_t(I,J)$. For any absorbing state $\boldsymbol{x}$, $x(I) = J+1$, $(x(1), \ldots, x(I-1))$ can be seen as an element in $\mathcal{H}(I,J)$. So $|\mathcal{X}_a(I,J)| = |\mathcal{H}I,J|$. Similarly, for any transient state $\boldsymbol{x}$, it can be seen as a reset vector for a J-station $(I+1)$-worker line. So $|\mathcal{X}_t(I,J)| = |\mathcal{H}(I+1,J)|$. $\square$

B.4 Proof of Lemma 10

For any transient state, a worker is working if and only if he is not waiting or blocked. By definition, Condition 1 implies that worker i is not waiting. Condition 2 considers two scenarios based on the location of worker $i+1$. If $x_{i+1} = 0$, worker $i+1$ is waiting; If $x_{i+1} \neq 0$, worker $i+1$ is also working, he does not block worker i if and only if $x_{i+1} > x_i$. $\square$

B.5 Proof of Lemma 11

After the $(k-1)st$ reset, the workers work on job k onwards from downstream to upstream. Note that both the blocked workers and working workers are carrying jobs. So the job that worker l_n is working on should count all the non-waiting workers from the downstream. $\square$

B.6 Proof of Lemma 12

Scenarios 1 and 2 correspond to the case that there are no successors of worker l_n waiting in front of him. So after worker l_n finishes his job at station $x(l_n)$, he walks to the next station, while the other workers keep their current states. Scenario 3 and 4 corresponds to case where there are successors of worker l_n waiting in front of him. If $l_n = 2, \ldots, I-1$, the job will eventually hand off to the last worker who is waiting for him and he walks back to wait for his predecessor. If $l_n = 1$, the job will also eventually hand off to the last worker who is waiting for him, and then he walks back to take a new job and hands it off to the second-to-last worker. All the predecessors of this worker will be blocked at station 1. $\square$

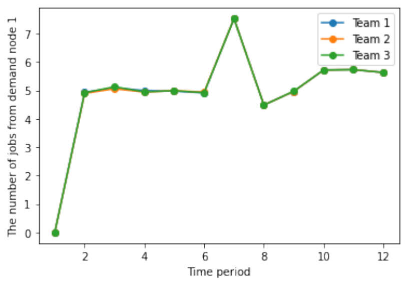
B.7 Proof of Lemma 13

If $h_{I-1} < J$, worker $I-1$ is working. Worker I has to wait for him after he completes job $k-1$. If $h_{I-m} = \ldots = h_{I-1} = J$, $h_{I-m-1} \neq J$ for $1 \leq m \leq I-2$, worker $I-m, \ldots, I-1$ are waiting in front of the last station. They will hand off their jobs to their predecessor, and worker worker $I-m$ will walk back to wait for worker $I-m-1$. $\square$

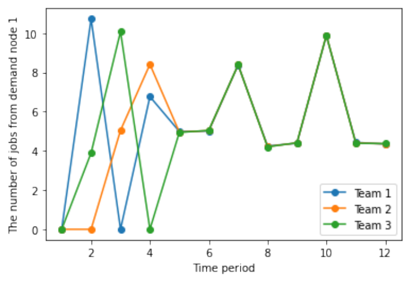
Appendix C

Appendix for Chapter 4

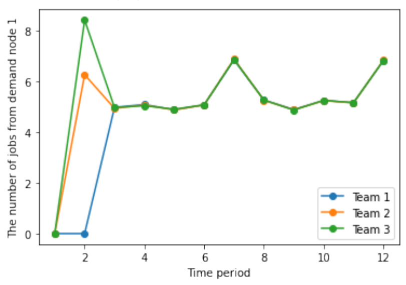
C.1 More simulation results



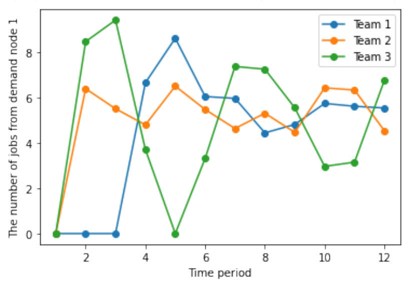
(a) Geometric



(b) Deterministic (one-point)

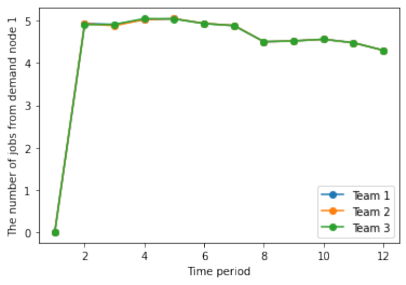


(c) Two-point

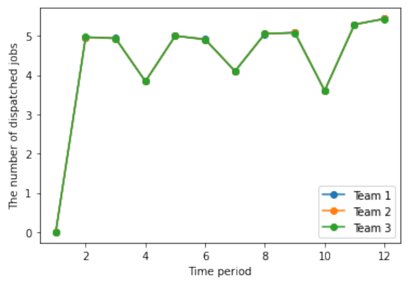


(d) Mixed

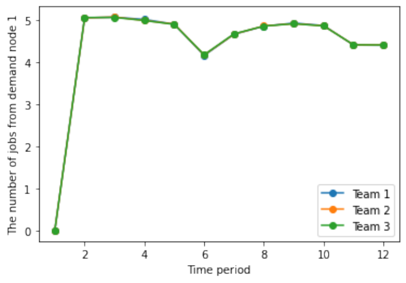
Figure C.1: The dispatch decision where there is a surge of demand at period 6



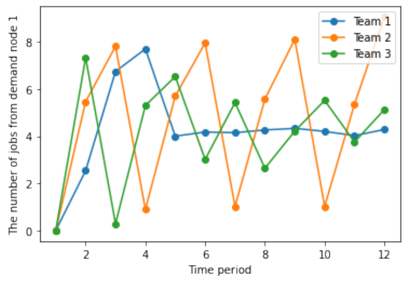
(a) Geometric



(b) Deterministic (one-point)



(c) Two-point



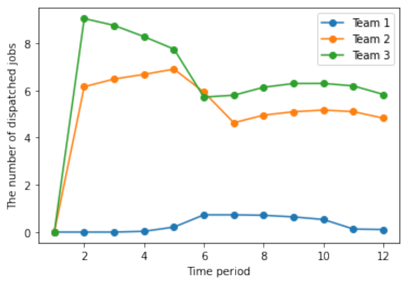
(d) Mixed

Figure C.2: The dispatch decision where there is a lift of demand from period 6

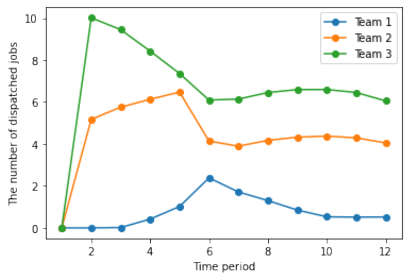
C.2 Additional discussion on network connectivity

In this next analysis, we consider a network as shown in Figure C.4. The service time distributions in all the supply nodes are geometric. In this case, the capacities of all the supply nodes are equal, however, for the arrivals at the demand nodes are not. Specifically, $\Lambda_1^t \sim 18.33\beta(3,3)$, $\Lambda_2^t \sim 8.33\beta(3,3)$, and $\Lambda_3^t \sim 3.33\beta(3,3)$. These constants are chosen in such a way such that, on average, we should expect each supply node to receive the same number of jobs from each demand node that it may be allocated jobs from.

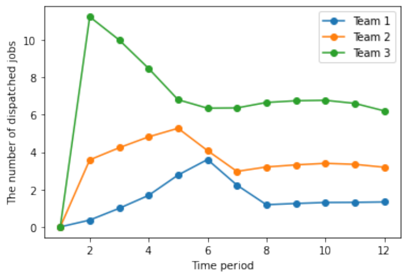
In Figure C.5, we plot the optimal policy in terms of the total number of jobs allocated to each of the supply nodes in (a). To make the analysis clearly, we further broke down the number of jobs arriving at each of the supply nodes to their source demand node in (b) – (d).



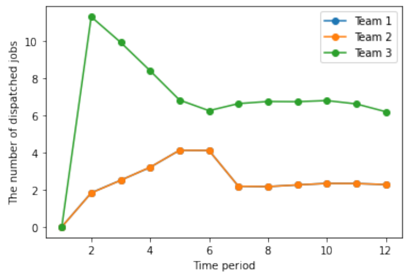
(a) $\mathcal{K}_1 = 5$, $\mathcal{K}_2 = 35$.



(b) $\mathcal{K}_1 = 10$, $\mathcal{K}_2 = 30$.



(c) $\mathcal{K}_1 = 15$, $\mathcal{K}_2 = 25$.



(d) $\mathcal{K}_1 = 20$, $\mathcal{K}_2 = 20$.

Figure C.3: Capacity pooling and allocation decisions

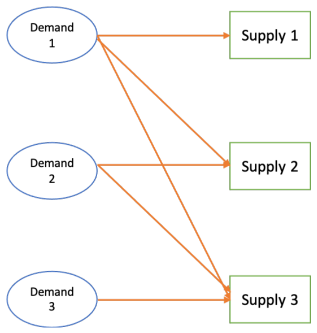
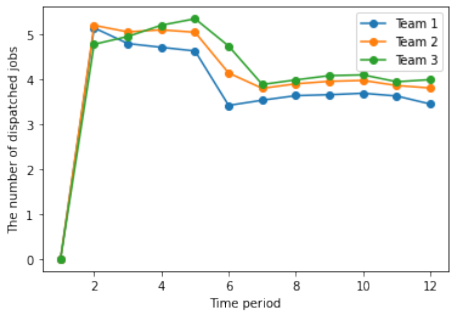
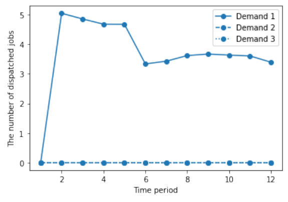


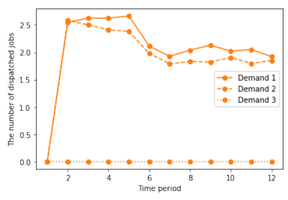
Figure C.4: Network Structure

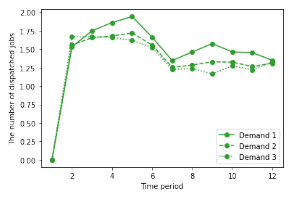
Observe that the number of jobs allocated to each team on the overall is almost the same, as earlier explained. This is confirmed when we examine the source of the allocated jobs in Figure C.5 (b)–(d). However, supply node 3, being the most connected of the three supply nodes is allocated marginally more than supply node 2, and in turn, supply node 1. In other words, the model is able to encode and take into consideration the reduction in the risk as a result of the flexibility in the network structure.



(a) Total number of jobs.







(b) Supply node 1. (c) Supply node 2. (d) Supply node 3.

Figure C.5: The numbers of dispatched jobs for a partially flexible structure.

C.3 Comparison against static policy

Here, we first present the derivation of the static policy model under the Pipeline Queue paradigm.

Most primarily, the key change is to modify our model (4.15) to the dynamics written using $w_{i,j}^{t,s}$, and setting them as the decision variables. In such a case, the no-underflow constraints, *i.e.* ensuring that no more jobs are dispatched out of the demand nodes than there are actual jobs in the nodes, is no longer guaranteed (previously, this was guaranteed as we were always considering a fraction of jobs from the demand nodes):

$$\sum_j w_{i,j}^{t,s} - x_i^{t,s} \leq 0 \quad \forall i \in \mathcal{I}, t \geq 0, s \geq 0$$

If $t \leq s$, this constraint can be reformulated as a linear constraint by writing $x_i^{t,s} = x_i^{0,s-t} - \sum_j \sum_{t'=0}^{t-1} w_{i,j}^{t',s-t+t'}$. However, if $t \leq s$, $x_i^{t,s}$ is a random variable.

We can transform it as an entropy constraint, that is:

$$k \log \mathbb{E} \exp \left[\left(\sum_j w_{i,j}^{t,s} - x_i^{t,s} \right) \Big/ k\theta_{3,i}^{t,s} \right] \leq 0$$

The overall optimization model for the static decision rule can be written as follows:

$$\begin{aligned}
&\text{minimize} && k && && \text{(C.1)}\\
&\text{subject to} && k \log \mathbb{E} \exp \left(\left(\sum_s y_j^{t,s} - \mathcal{K}_j^t \right) \Big/ k\theta_{1,j}^t \right) \leq 0, && \forall j \in \mathcal{J}, \forall t > 0\\
& && k \log \mathbb{E} \exp \left(\left(\sum_s \left(b_i^{t,s} - C_i^t \right) x_i^{t,s} \right) \Big/ k\theta_{2,i}^t \right) \leq 0, && \forall i \in \mathcal{I}, \forall t > 0\\
& && k \log \mathbb{E} \exp \left[\left(\sum_j w_{i,j}^{t,s} - x_i^{t,s} \right) \Big/ k\theta_{3,i}^{t,s} \right] \leq 0, && \forall i \in \mathcal{I}, \forall t > s\\
& && \sum_j \sum_{t'=0}^{t} w_{i,j}^{t',s-t+t'} - x_i^{0,s-t} \leq 0, && \forall i \in \mathcal{I}, \forall t \leq s\\
& && w_{i,j}^{t,s} \geq 0, && \\
&\text{with dynamics} && x_i^{t,0} = \lambda_i^t, && \forall t > 0\\
& && x_i^{t,s} = x_i^{t-1,s-1} - \sum_j w_{i,j}^{t-1,s-1}, && \forall t, s > 0\\
& && y_j^{t,0} = \sum_{i \in \mathcal{I}_j} \sum_s w_{i,j}^{t-1,s}, && \forall t > 0\\
& && y_j^{t,s} \sim \text{Bin}(y_j^{t-1,s-1}, q_j^{t-1,s-1}), && \forall t, s > 0
\end{aligned}$$

If $t > s$, the no-underflow constraint can be reformulated as:

$$k \log \mathbb{E} \exp \left[\left(\sum_j w_{i,j}^{t,s} - x_i^{t,s} \right) \Big/ k\theta_{3,i}^{t,s} \right] = \sum_j \sum_{t'=0}^{s} w_{i,j}^{t-s+t',t'} + k g_i^{t-s} \left(-\frac{1}{k\theta_{3,i}^{t,s}} \right)$$

We can see that the constraint is linear in the decision variables $w_{i,j}^{t,s}$. Similarly, the capacity constraints and the queuing cost constraints can be reformulated to linear expressions in $w_{i,j}^{t,s}$.

Detailed results

To compare the performance of the static decision with the adaptive decision rules, we return to the original simple network of Figure 4.3, under geometric service times. We changed the arrival pattern to $\Lambda^t \sim (15 - n) + 2n\beta(3, 3)$, in other words, its mean is 15, and its support is $[15 - n, 15 + n]$. The parameter n here, adjusts the variance in the arrivals; if $n = 0$, the arrival is deterministic, and if $n = 15$, the arrival recovers the scaled beta distribution we had previously used.

Here, we shall compare the performance of three models, the first being the adaptive model (4.15). For the static models, note first that there is a new constraint, termed the 'no-underflow constraints' (see Appendix C.3 for more details), which is a hard constraint. The hardness of this constraint is controlled by the θ-parameter, specifically, $\theta_{3,i}^{t,s}$ at all times t and s. Here, we attempt two models. The first uses $\theta_{3,i}^{t,s} = 1$, thereby effectively treating the constraint as a soft constraint and having no guarantees against dispatching more jobs than there are. We term this the 'vanilla static' model. The second uses $\theta_{3,i}^{t,s} = 0.1$, which models it as a hard constraint, but in exchange, would increase the overall risk of the model, since the model now needs to work harder to ensure constraint satisfaction. We term this the 'adjusted static' model. We plot the resultant risk levels k for different choices of parameter n in Figure C.6.

From Figure C.6, we see that for all models, the risk level scales up with variability in demand n. Moreover, both static models experience much higher risk levels k than our proposed fully adaptive model. The core reason for this is because it is increasingly difficult to prevent constraint violation the longer the modelling time with purely static decisions. Also, explained earlier, the tighter constraint requirement for the Adjusted static model results in a higher risk level k than its Vanilla static counterpart.

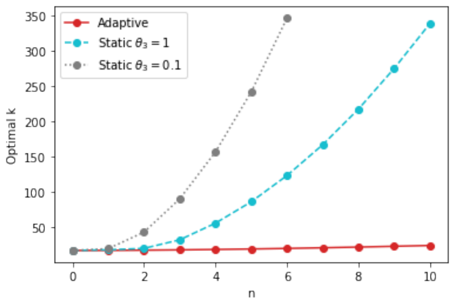


Figure C.6: Optimal value of k for adaptive and static decisions

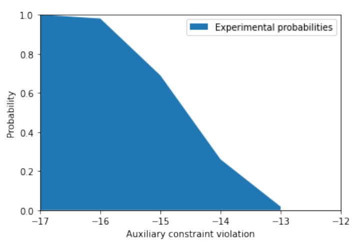
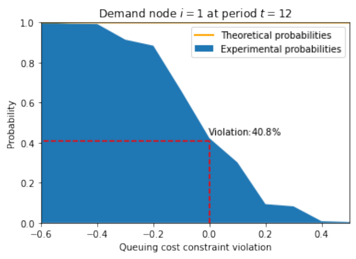
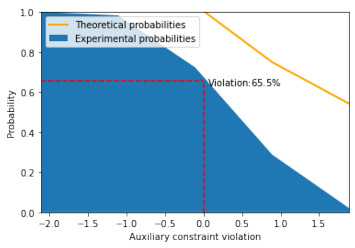
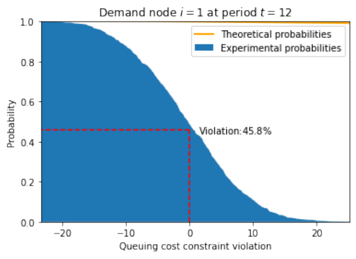
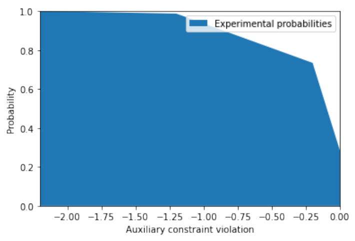
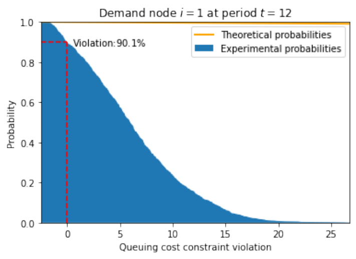
	Auxiliary constraint violation	Cost constraint violation
Adaptive decision		
Static decision		
Adjusted static decision		

Table C.1: Comparison between adaptive and static decision rules

Bibliography

Ahn Hyun-Soo, Duenyas Izak, Zhang Rachel Q. Optimal control of a flexible server // Advances in Applied Probability. 2004. 139–170.

Andradóttir Sigrún, Ayhan Hayriye. Throughput maximization for tandem lines with two stations and flexible servers // Operations Research. 2005. 53, 3. 516–531.

Andradóttir Sigrún, Ayhan Hayriye, Down Douglas G. Server assignment policies for maximizing the steady-state throughput of finite queueing systems // Management Science. 2001. 47, 10. 1421–1439.

Andradóttir Sigrún, Ayhan Hayriye, Down Douglas G. Dynamic server allocation for queueing networks with flexible servers // Operations Research. 2003. 51, 6. 952–968.

Andradóttir Sigrún, Ayhan Hayriye, Down Douglas G. Compensating for failures with flexible servers // Operations Research. 2007a. 55, 4. 753–768.

Andradóttir Sigrún, Ayhan Hayriye, Down Douglas G. Dynamic assignment of dedicated and flexible servers in tandem lines // Probability in the Engineering and Informational Sciences. 2007b. 21, 4. 497.

Armbruster Dieter, Gel Esma S. Bucket brigades revisited: Are they always effective? // European Journal of Operational Research. 2006. 172, 1. 213–229.

Armony Mor, Chan Carri W, Zhu Bo. Critical care capacity management: Understanding the role of a step down unit // Production and Operations Management. 2018. 27, 5. 859–883.

Armony Mor, Ward Amy R. Fair dynamic routing in large-scale heterogeneous-server systems // Operations Research. 2010. 58, 3. 624–637.

Bandi C., Loke G.G. Exploiting Hidden Convexity for Optimal Flow Control in Queueing Networks // Extracted from SSRN 3190874. 2018.

Bartholdi III John J, Bunimovich Leonid A, Eisenstein Donald D. Dynamics of two-and three-worker bucket brigade production lines // Operations research. 1999. 47, 3. 488–491.

Bartholdi III John J, Eisenstein Donald D. The bucket brigade web page. 1996a.

Bartholdi III John J, Eisenstein Donald D. A production line that balances itself // Operations Research. 1996b. 44, 1. 21–34.

Bartholdi III John J, Eisenstein Donald D. Using bucket brigades to migrate from craft manufacturing to assembly lines // Manufacturing & Service Operations Management. 2005. 7, 2. 121–129.

Bartholdi III John J, Eisenstein Donald D, Foley Robert D. Performance of bucket brigades when work is stochastic // Operations research. 2001. 49, 5. 710–719.

Bartholdi III John J, Eisenstein Donald D, Lim Yun Fong. Bucket brigades on in-tree assembly networks // European Journal of Operational Research. 2006. 168, 3. 870–879.

Bartholdi III John J, Eisenstein Donald D, Lim Yun Fong. Deterministic chaos in a model of discrete manufacturing // Naval Research Logistics (NRL). 2009. 56, 4. 293–299.

Bartholdi III John J, Hackman Steven T. Warehouse and Distribution Science. 2019.

Batt Robert J, Kc Diwas S, Staats Bradley R, Patterson Brian W. The effects of discrete work shifts on a nonterminating service system // Production and operations management. 2019. 28, 6. 1528–1544.

Bell Steven L, Williams Ruth J, others . Dynamic scheduling of a system with two parallel servers in heavy traffic with resource pooling: Asymptotic optimality of a threshold policy // Annals of Applied Probability. 2001. 11, 3. 608–649.

Bukchin Yossi, Hanany Eran, Khmelnitsky Eugene. Bucket brigade with stochastic worker pace // IISE Transactions. 2018. 50, 12. 1027–1042.

Chan Carri W, Huang Michael, Sarhangian Vahid. Dynamic server assignment in multiclass queues with shifts, with applications to nurse staffing in emergency departments // Operations Research. 2021.

Correia Isabel, Nickel Stefan, Gama Francisco Saldanha-da. A stochastic multi-period capacitated multiple allocation hub location problem: Formulation and inequalities // Omega. 2018. 74. 122–134.

Dai J. G., Shi P. A Two-Time-Scale Approach to Time-Varying Queues in Hospital Inpatient Flow Management // Operations Research. 2017. 65, 2. 514–536.

Dai JG, Gluzman Mark. Queueing network controls via deep reinforcement learning // arXiv preprint arXiv:2008.01644. 2020.

Dai JG, Shi Pengyi. Inpatient overflow: An approximate dynamic programming approach // Manufacturing & Service Operations Management. 2019. 21, 4. 894–911.

Duenyas Izak, Gupta Diwakar, Olsen Tava Lennon. Control of a single-server tandem queueing system with setups // Operations Research. 1998. 46, 2. 218–230.

Farrar Timothy Martin. Optimal use of an extra server in a two station tandem queueing network // IEEE Transactions on Automatic Control. 1993. 38, 8. 1296–1299.

Fershtman Daniel, Pavan Alessandro. Dynamic Matching: Experimentation and Cross-subsidization. 2015.

Ghosh Supriyo, Varakantham Pradeep, Adulyasak Yossiri, Jaillet Patrick. Dynamic repositioning to reduce lost demand in bike sharing systems // Journal of Artificial Intelligence Research. 2017. 58. 387–430.

Guo Mian, Guan Quansheng, Ke Wende. Optimal scheduling of VMs in queueing cloud computing systems with a heterogeneous workload // IEEE Access. 2018. 6. 15178–15191.

Gupta Diwakar, Wang Lei. Revenue management for a primary-care clinic in the presence of patient choice // Operations Research. 2008. 56, 3. 576–592.

Harrison J Michael, López Marcel J. Heavy traffic resource pooling in parallel-server systems // Queueing systems. 1999. 33, 4. 339–368.

He Shuangchi, Sim Melvyn, Zhang Meilin. Data-driven patient scheduling in emergency departments: A hybrid robust-stochastic approach // Management Science. 2019. 65, 9. 4123–4140.

Hopp Wallace J, Oyen Mark P. Agile workforce evaluation: a framework for cross-training and coordination // Iie Transactions. 2004. 36, 10. 919–940.

Hopp Wallace J, Spearman Mark L. Factory physics. 2011.

Hopp Wallace J, Tekin Eylem, Van Oyen Mark P. Benefits of skill chaining in serial production lines with cross-trained workers // Management Science. 2004. 50, 1. 83–98.

Iravani Seyed MR, Posner Morton J. M., Buzacott John A. A two-stage tandem queue attended by a moving server with holding and switching costs // Queueing systems. 1997. 26, 3. 203–228.

Işık Tuğçe, Andradóttir Sigrún, Ayhan Hayriye. Optimal control of queueing systems with non-collaborating servers // Queueing Systems. 2016. 84, 1. 79–110.

Jaillet P., Loke G.G., Sim M. Strategic Manpower Planning under Uncertainty // Extracted from SSRN 3168168. 2019.

Jaillet Patrick, Lu Xin. Online resource allocation problems // Rock & Soil Mechanics. 2011. 86. 3701–3704.

Jaillet Patrick, Lu Xin. Online stochastic matching: New algorithms with better bounds // Mathematics of Operations Research. 2014. 39, 3. 624–646.

Johari Ramesh, Kamble Vijay, Kanoria Yash. Matching while learning // Operations Research. 2021.

Karp Richard M, Vazirani Umesh V, Vazirani Vijay V. An optimal algorithm for on-line bipartite matching // Proceedings of the twenty-second annual ACM symposium on Theory of computing. 1990. 352–358.

Kaufman David L, Ahn Hyun-soo, Lewis Mark E. On the introduction of an agile, temporary workforce into a tandem queueing system // Queueing Systems. 2005. 51, 1. 135–171.

Kırkızlar Eser, Andradóttir Sigrún, Ayhan Hayriye. Robustness of efficient server assignment policies to service time distributions in finite-buffered lines // Naval Research Logistics (NRL). 2010. 57, 6. 563–582.

Kırkızlar Eser, Andradóttir Sigrún, Ayhan Hayriye. Flexible servers in understaffed tandem lines // Production and Operations Management. 2012. 21, 4. 761–777.

Kırkızlar Eser, Andradóttir Sigrún, Ayhan Hayriye. Profit maximization in flexible serial queueing networks // Queueing Systems. 2014. 77, 4. 427–464.

Lim Yun Fong. Cellular bucket brigades // Operations research. 2011. 59, 6. 1539–1545.

Lim Yun Fong. Order-picking by cellular bucket brigades: a case study // Warehousing in the global supply chain. 2012. 71–85.

Lim Yun Fong. Performance of cellular bucket brigades with hand-off times // Production and Operations Management. 2017. 26, 10. 1915–1923.

Lim Yun Fong, Wu Yue. Cellular bucket brigades on U-lines with discrete work stations // Production and Operations Management. 2014. 23, 7. 1113–1128.

Lim Yun Fong, Yang Kum Khiong. Maximizing throughput of bucket brigades on discrete work stations // Production and Operations Management. 2009. 18, 1. 48–59.

Lyu Guodong, Cheung Wang Chi, Teo Chung-Piaw, Wang Hai. Multi-objective online ride-matching // Available at SSRN 3356823. 2019.

Mathematical analysis: functions, limits, series, continued fractions. // . 1965.

Mandelbaum Avishai, Stolyar Alexander L. Scheduling flexible servers with convex delay costs: Heavy-traffic optimality of the generalized $c\mu$-rule // Operations Research. 2004. 52, 6. 836–855.

Martonosi Susan E. Dynamic server allocation at parallel queues // IIE Transactions. 2011. 43, 12. 863–877.

Mazur James E, Hastie Reid. Learning as accumulation: A reexamination of the learning curve. // Psychological Bulletin. 1978. 85, 6. 1256.

Özkan Erhun, Ward Amy R. Dynamic matching for real-time ride sharing // Stochastic Systems. 2020. 10, 1. 29–70.

Puha Amber L, Ward Amy R. Scheduling an overloaded multiclass many-server queue with impatient customers // Operations Research & Management Science in the Age of Analytics. 2019. 189–217.

Qi Wei, Li Lefei, Liu Sheng, Shen Zuo-Jun Max. Shared mobility for last-mile delivery: Design, operational prescriptions, and environmental impact // Manufacturing & Service Operations Management. 2018. 20, 4. 737–751.

Reeves Gary R, Sweigart James R. Multiperiod resource allocation with variable technology // Management Science. 1982. 28, 12. 1441–1449.

Riedel Marco. Online matching for scheduling problems // Annual Symposium on Theoretical Aspects of Computer Science. 1999. 571–580.

Rosberg Zvi, Varaiya P, Walrand J. Optimal control of service in tandem queues // IEEE Transactions on Automatic Control. 1982. 27, 3. 600–610.

Ross Sheldon. A first course in probability. 2014.

Ross Sheldon M, Kelly John J, Sullivan Roger J, Perry William James, Mercer Donald, Davis Ruth M, Washburn Thomas Dell, Sager Earl V, Boyce Joseph B, Bristow Vincent L. Stochastic processes. 2. 1996.

Rostami Borzou, Kämmerling Nicolas, Naoum-Sawaya Joe, Buchheim Christoph, Clausen Uwe. Stochastic single-allocation hub location // European Journal of Operational Research. 2021. 289, 3. 1087–1106.

Serna Ainhoa, Gerrikagoitia Jon Kepa, Bernabe Unai, Ruiz Tomás. A method to assess sustainable mobility for sustainable tourism: The case of the public bike systems // Information and Communication Technologies in Tourism 2017. 2017. 727–739.

Shu Jia, Chou Mabel C, Liu Qizhang, Teo Chung-Piaw, Wang I-Lin. Models for effective deployment and redistribution of bicycles within public bicycle-sharing systems // Operations Research. 2013. 61, 6. 1346–1359.

Spivey Michael Z, Powell Warren B. The dynamic assignment problem // Transportation science. 2004. 38, 4. 399–419.

Tang Q., Zhang Y., Zhou M. Vehicle Repositioning under Uncertainty // From SSRN 3612626. 2020.

Véricourt Francis de, Jennings Otis B. Nurse staffing in medical units: A queueing perspective // Operations research. 2011. 59, 6. 1320–1331.

Villalobos JR, Estrada F, Munoz LF, Mar L. Bucket brigade: A new way to boost production // Twin Plant News. 1999a. 14, 12. 57–61.

Villalobos JR, Munoz LF, Mar L. Assembly line designs that reduce the impact of personnel turnover // Proceedings of IIE Solutions Conference, Phoenix, AZ. 1999b.

Webster Scott, Ruben Robert A, Yang Kum-Khiong. Impact of storage assignment decisions on a bucket brigade order picking line // Production and Operations Management. 2012. 21, 2. 276–290.

Williams Ruth J. On dynamic scheduling of a parallel server system with complete resource pooling // Fields Institute Communications. 2000. 28, 49-71. 5–1.

Yan Zhenzhen, Gao Sarah Yini, Teo Chung Piaw. On the design of sparse but efficient structures in operations // Management Science. 2018. 64, 7. 3421–3445.

Yom-Tov Galit B, Mandelbaum Avishai. Erlang-R: A time-varying queue with reentrant customers, in support of healthcare staffing // Manufacturing & Service Operations Management. 2014. 16, 2. 283–299.

Youssef Marie Josepha, Veeravalli Venugopal V, Farah Joumana, Nour Charbel Abdel, Douillard Catherine. Resource allocation in NOMA-based self-organizing networks using stochastic multi-armed bandits // arXiv preprint arXiv:2101.06340. 2021.

Zhou Minglong, Loke Gar Goei, Bandi Chaitanya, Liau Zi Qiang Glen, Wang Wilson. Intraday Scheduling with Patient Re-Entries and Variability in Behaviours // Manufacturing & Service Operations Management. 2021. Available online.