# Maximizing throughput in zero-buffer tandem lines with dedicated and flexible servers

Mohammad H. Yarmand and Douglas G. Down

*Department of Computing and Software, McMaster University, Hamilton, ON, L8S 4K1, Canada*

**Abstract**

For tandem queues with no buffer spaces and both dedicated and flexible servers, we study how flexible servers should be assigned to maximize the throughput. When there is one flexible server and two stations each with a dedicated server, we completely characterize the optimal policy. We use the insights gained from applying the Policy Iteration algorithm on systems with three, four, and five stations to devise heuristics for systems of arbitrary size. These heuristics are verified by numerical analysis. We also discuss the throughput improvement, when for a given server assignment, dedicated servers are changed to flexible servers. Supplementary materials are available for this article. Go to the publisher's online edition of *IIE Transactions* for detailed proofs.

*Keywords:* Server Allocation, Zero Buffer, Tandem Queue, Flexible Servers

## 1. Introduction

Consider a tandem queueing network with $N \geq 2$ stations, $M \geq 1$ dedicated servers, and $F \geq 1$ flexible servers. At any given time, each station can be assigned multiple servers and each server can work only on one job, and a job may have at most one server assigned to it. Assume that the service times of each job at station $i \in \{1, \ldots, N\}$ are independent and identically distributed with rate $\mu_i$, i.e. the service rate of each server at the $i^{th}$ station is only dependent on the station.

In the above system, we assume that dedicated servers are already assigned to the stations. We are interested in determining the dynamic assignment policy for flexible servers that maximizes the long-run average throughput. For simplicity, we assume the travel times for jobs to progress and also the travel times for flexible servers to move between stations are negligible. We also assume

*Email address:* yarmanmh,downd@mcmaster.ca (Mohammad H. Yarmand and Douglas G. Down)

there is an infinite supply of jobs in front of the first station and infinite space for jobs completed at the last station. There are no buffer spaces between stations.

We introduce the concept of "hand-off" for flexible servers. This appears to be a new concept, as there does not appear to be any work in the literature that considers a mixture of dedicated and flexible servers, where servers cannot collaborate. Hand-off happens when a flexible server passes the job it is serving to a dedicated server at the same station. Although it is possible to perform hand-off at any time, we let it occur only in the following two cases. When a station has a busy flexible server and a free dedicated server, the flexible server can pass its job to the dedicated server and become free. When a station has a busy flexible server and a blocked dedicated server, jobs can be swapped between the two servers. In either of the two cases, we say a hand-off has taken place. In this work, we only consider the manufacturing blocking mechanism (see [7] for details of manufacturing versus communication blocking).

Any allocation policy should define appropriate actions when blocking or starvation occurs. In cases where there are multiple blocked or starved servers, policies should prioritize resolving blocking or starvation of the involved servers. We will show that when workloads (i.e. the ratio of the mean service time to the number of dedicated servers for a station) are equal, a near-optimal policy is one that clears blocking from the end of the line to the beginning and avoids starving servers. With different workloads, a near-optimal policy prioritizes clearing blocking for stations with higher mean service times.

An instance of the application of the problem described above is the bed management problem from the healthcare domain. Bed management deals with optimizing patient flow into and out of beds, so that waiting times are reduced [8]. A patient arriving at a hospital might need to go through a sequence of units. For example, the patient might go through, in order, the emergency, express, medicine, surgery, recovery, and complex units. The role of the express unit is to accommodate the patient until a bed becomes available in the medicine unit. A number of beds, called express beds, are shared between the emergency, express, and medicine units to move patients between these units. However, sharing is performed in an ad-hoc manner and is limited to these three units only. Several questions arise, including: how much is the throughput improvement, if beds are shared among the units? How should the shared beds be allocated dynamically to the units? If we need to take a number of dedicated beds from the units and share them, which units should be chosen?

While we do not model this application directly, our results are a first step in an analytic approach to such problems. The literature of tandem lines with multiple servers and finite buffers is not large (see the discussion in [15]), but there could potentially be additional applications in areas such as manufacturing in zero-buffer settings where a subset of fixtures could be reconfigured for different stations in the line (see [10] for an example of a zero-buffer automobile manufacturing setting).

Beyond multiple server and finite buffer systems, there is a rich literature on the assignment of flexible servers in tandem queues. Most of the papers in this domain focus on minimizing holding costs. Iravani [11] considers tandem queues attended by a moving server with holding and switching costs. As a basic model, he shows that for two-stage tandem queues, the policy which minimizes the total discounted and long run average holding and switching costs is a greedy and exhaustive policy in the second stage. The first stage could then follow static, gated-limited, or double-threshold policies.

Ahn et al. [1] consider the optimal control of two parallel servers in a two-stage tandem queueing system with two flexible servers and holding costs. They examine both collaborative and non-collaborative cases. In the collaborative case, servers may collaborate to work on the same job at the same time, while in the non-collaborative case each job can be served by at most one server at a time. They provide simple conditions under which it is optimal to allocate both servers to station 1 or 2 in the collaborative case. In the non-collaborative case, they show that the same condition as in the collaborative case guarantees the existence of an optimal policy that is exhaustive in station 1. However the condition for exhaustive service at station 2 to be optimal does not carry over.

Pandelis [14] considers a two-station tandem line with both dedicated and flexible servers where all servers have time-varying rates. Servers can work collaboratively. With a given probability, jobs can leave the system after completion in the first station. The optimal policy to minimize holding costs is described.

For a two-class queueing system with one dedicated server, one flexible server, and no exogenous arrivals, Ahn et al. [2] characterize the server assignment policy that minimizes the expected total holding cost incurred until all jobs initially present in the system have departed.

Andradóttir et al. [6] consider tandem queues having both flexible and dedicated servers with finite or infinite buffers between stations. They study the dynamic assignment of servers, in the collaborative case, such that the long-run average throughput is maximized. They assume that the

service requirements of jobs at stations are exponentially distributed. For 2 stations and 3 servers (both flexible and dedicated) the allocation of the flexible servers is of threshold type. Multiple thresholds are used to express allocation policies and the thresholds are specified.

Wu et al. [16] consider the notion of dedicated and flexible servers to determine the allocation of flexible servers that minimizes holding cost in a clearing system with two queues in tandem and in which dedicated servers are subject to failure. Wu et al. [17] extend [16] to more general serial lines with external arrivals under discounted and average cost criteria.

Andradóttir et al. [4, 5], consider the dynamic assignment of servers to maximize the long-run average throughput in queueing networks with flexible servers. They consider tandem queues with two stations and flexible servers with a finite buffer between the stations. Servers work collaboratively. They use Policy Iteration to show that with less than three servers, a policy which avoids blocking at the first station and starvation at the second station, is optimal. Andradóttir and Ayhan [3] study the same system with three servers. Assuming server 1(2) is more efficient than server 2(3) for serving jobs at the second station, they prove that the optimal policy has three mandates: (i) server 1 works at station 1, unless the station is blocked; (ii) server 3 works at station 2, unless it is starved; (iii) server 2 is a roving server, it works at station 1 if the buffer is low and works at station 2 when the buffer level is high.

Hasenbein and Kim [9] also consider the system introduced in the last paragraph. They prove that the conjecture proposed by Andradóttir and Ayhan [3] for generic numbers of servers, is indeed true. They use general properties of the bias of the optimal policy to show that a threshold policy on buffer levels exists that relies on ordering of server efficiency at stations. They further determine the threshold values. They also prove that restricting the buffer size decreases the maximum achievable throughput.

Kırkızlar et al. [12] analyze a tandem line which is understaffed, i.e. there are more stations than servers. They consider tandem queues with three stations, two servers, different flexibility structures, and either deterministic service times and arbitrary buffers or exponential service times and small buffers. In the deterministic setting, they prove that the best possible production rate with full server flexibility and infinite buffers can be attained with partial flexibility and zero buffers.

In contrast to [1, 2, 11, 14] which consider holding costs, we study throughput in our work. The problem in [3] differs from our work as it studies the collaborative case and includes buffers

between stations. The work in [4, 6, 9] encompasses buffer spaces, collaborative servers, assumes heterogeneous service rates for servers, and allows a single server per station. Note that in a zero-buffer setting, server allocation in tandem lines with collaborative servers is a different problem than server allocation with non-collaborative servers. In [5], jobs in the system belong to different classes and service rates are heterogeneous based on server and job class. In this paper, we determine optimal policies for dynamic allocation of flexible servers in zero-buffer tandem lines with heterogeneous service rates for stations. We study the non-collaborative case and our goal is to maximize throughput. In addition, we report the throughput improvement gained from making dedicated servers flexible.

This paper is organized as follows. In Section 2, the optimality of hand-off and non-idling for tandem lines with two stations is shown. In Section 3 we use Markov Decision Process theory to derive the optimal policy for tandem lines with two stations, an arbitrary number of dedicated servers, and one flexible server. We further show how to employ the Policy Iteration algorithm to construct the optimal policy. In Section 4 we apply Policy Iteration to larger instances (tandem lines with 3, 4, and 5 stations) and describe a more generic near-optimal policy. Using the insights gained from Section 4, in Section 5 we provide heuristics for allocation policies for systems of arbitrary size and with heterogeneous mean service times. We also study configurations with service times that are not exponentially distributed. Section 6 studies the effects of increasing the number of flexible servers on the throughput. Finally, Section 7 concludes the paper and discusses future work. The proofs of most of our results are included in the online supplement.

## 2. Optimal Policy Properties

In this section we introduce and prove two properties of the optimal policy for tandem lines with two stations. By optimal we mean that the policy maximizes throughput, which we equivalently consider that the policy leads to the highest number of departures (at every point in time) from the first station. These properties are the optimality of hand-off and of non-idling of flexible servers. As a result, the state and action spaces when modelling the system with Markov Decision Processes are constrained so that any states or actions which do not perform hand-off or idle flexible servers can be excluded. This fact becomes helpful when search spaces are big, i.e. for systems with a large number of stations or servers. In Theorems 1 and 2 there is one dedicated server at each station and

one flexible server that can work at either station. In Section 2.4, Corollary 1 extends the theorems and shows that Theorems 1 and 2 hold for systems with arrivals and for clearing systems. Also, Corollary 2 shows that the results hold for systems with arbitrary numbers of dedicated and flexible servers. The proofs of Theorems 1 and 2 are sample path proofs, i.e. we fix a sample path and generate all of the service times at the beginning. The service times can be from any distribution.

*2.1. Hand-off property*

In this subsection, we state and prove Theorem 1 on the optimality of hand-off.

**Theorem 1**: *The optimal policy performs hand-off whenever possible.*

**Proof**: There are only two scenarios under which hand-off is possible:

1. A dedicated server becomes blocked in the same station where a flexible server is working (for $N = 2$, this can only happen in the first station). Hand-off is used to clear blocking.
2. A dedicated server becomes available in the same station where a flexible server is working (for $N = 2$, this can only happen in the second station). Hand-off is used to avoid a flexible server working at a station where there is an idle dedicated server.

Let $\omega$ and $\omega'$ be policies that always perform hand-off, with the only exception that $\omega'$ does not perform a hand-off at one of the times when a hand-off is possible.

We define the property $ahead(n)$ as follows. Comparing the system under policies $\omega$ and $\omega'$ when $n$ jobs have departed from the first station under $\omega$, $ahead(n)$ holds if any of the following hold for each job at a station: i) the same job is being served with identical time spent in the system; ii) the same job is being served with less residual service time under $\omega$; iii) $\omega$ is serving a job that $\omega'$ has not yet admitted. Note that when there are two jobs in a station, more than one of the above situations (i.e. i), ii), and iii)) can hold. Both stations must satisfy these conditions. The proof is structured as follows:

For both scenarios, *Lemma 1* states that if the $n^{th}$ departure happens sooner under $\omega$ than $\omega'$ and $ahead(n)$ holds, then the $n{+}1^{st}$ departure under $\omega'$ will not occur sooner under $\omega$.

For both scenarios, *Lemma 2* completes the proof by showing that given a system with $m > 1$ departures from the first station that satisfies $ahead(m)$, Lemma 1 can be applied iteratively on the system to satisfy $ahead(m + 1)$.

Given Lemmas 1 and 2, a simple induction argument follows. Define $D^{\omega}(n)$ to be the time at which the $n^{th}$ departure from the first station occurs under $\omega$. We use induction to show $\forall n, m.(D^{\omega}(n) \leq D^{\omega'}(n) \wedge m > n) \Rightarrow (D^{\omega}(m) \leq D^{\omega'}(m))$.

(Theorem 1) $\square$

**Lemma 1**: *For both scenarios,* $\forall n \ . \ (D^{\omega}(n) \leq D^{\omega'}(n) \wedge ahead(n)) \Rightarrow (D^{\omega}(n+1) \leq D^{\omega'}(n+1))$.

**Lemma 2**: $\forall m > 2 \ . \ ahead(m) \Rightarrow ahead(m+1)$.

*2.2. Proof of Lemmas 1 and 2*

In all the lemmas below, the following labelling convention is used. The labelling *Case* $l_1.....l_{k-1}.l_k$ means that all of the assumptions made in cases $l_1$ to $l_1.....l_{k-1}$ hold for this case, in addition to the assumption introduced in $l_k$. *Case* $l_1.....l_{k-1}.l_k$ and *Case* $l_1.....l_{k-1}.l'_k$ where $l_k \neq l'_k$ refer to two different branches of $l_1.....l_{k-1}$ that differ only in the assumptions introduced in $l_k$ and $l'_k$.
In all figures in the lemmas below, the top label is the time stamp of the configuration, ovals represent stations, the first row shows $\omega$ and the second shows $\omega'$. Within each oval are job numbers, in each station the top job is served by a dedicated server and the bottom job by a flexible server. The notation below the stations shows the residual service time of a job appearing in the same station under both policies when residual service times are different. In some cases, a hand-off is shown by two configurations in a row separated by a vertical line.
The following notation is used throughout the following lemmas. Let $X^{p,i}_{n,t}$ be the residual service time of job $n$ at time $t$ served by server $X$ that works in station $i \in \{1,2\}$ under policy $p \in \{\omega, \omega'\}$. Server $X$ can be replaced by $d$ or $f$, representing a dedicated or a flexible server, respectively. The residual service time for a job is the time remaining to complete the job's service. Dropping any of the indices of Server $X^{p,i}_{n,t}$ means that the dropped index can take any value. Also note that the service time of job $n$ is independent of its admission time, the server being dedicated/flexible, and the policy used, i.e. $X^i_n = d^{p,i}_{n,t} = f^{\omega',i}_{n,t'}$. To clarify, $d^{\omega,i}_{n,t}$ is the residual service time of job $n$ at time $t$ which is being served by a dedicated server at station $i$ under policy $\omega$. Finally, to refer to a server working at station $i$, we use $\sigma^i_X$, where $X \in \{d, f\}$ as above. Note that $D^w(n)$ is the time of the $n^{th}$ departure which in general is not the $n^{th}$ job to enter the system. To emphasize this fact, we do not use $n$ to enumerate jobs according to their time of entry ($k$ and $p$ are used instead).

In all of the following lemmas, let $t_0$ be the hand-off time at which the two policies make a different choice. We give part of the proof of Lemma 1 here (to give a flavour of how the proof

proceeds), the remainder is in the online supplement. The proof of Lemma 2 is also given in the online supplement.

**Lemma 1**: *For both scenarios,* $\forall n.(D^{\omega}(n) \leq D^{\omega'}(n) \wedge ahead(n)) \Rightarrow (D^{\omega}(n+1) \leq D^{\omega'}(n+1))$.

**Proof**: The two primary steps of the proof (basis and inductive steps) are shown in the following. We consider the first scenario here and leave the second scenario for the online supplement.

**Basis step**:

If time $t_0$ is not reached, both systems remain the same and there is nothing to prove in the basis step. Otherwise, start the system from the empty state and let the first departure from the first station ($D^{\omega}(1) = D^{\omega'}(1)$) occur. If $d^1_{2,t_0} < \min\{f^1_{3,t_0}, d^2_{1,t_0}\}$, the system follows the first scenario. Server $\sigma^1_d$ is serving job 2, $\sigma^1_f$ is serving job 3, and $\sigma^2_d$ is serving job 1. Up until time $t_0$, the two policies are the same. When time $t_0$ is reached, the following cases are possible.

**Case 1:** Assume $d^2_{1,t_0} > f^1_{3,t_0}$, meaning that the flexible server completes its service before the dedicated server. Let $t_1 = f^1_{3,t_0}$. The policy $\omega'$ waits $t_1$ time units until $\sigma^1_f$ completes job 3 and sends the server to the second station with $f^{\omega',2}_{3,t_0+t_1}$ as residual service time. The policy $\omega$ performs a hand-off at $t_0$ and sends the flexible server to the second station with a residual service time of $f^{\omega,2}_{2,t_0}$. Therefore at $t_0 + t_1$, $D^{\omega}(2) < D^{\omega'}(2)$.

**Case 2:** Assume $d^2_{1,t_0} \leq f^1_{3,t_0}$, meaning that the dedicated server completes its service before the flexible server. Let $t_1 = d^2_{1,t_0}$. The policy $\omega'$ waits $t_1$ time units until $\sigma^2_d$ becomes available. So at time $t_0 + t_1$, the blocked job goes to the second station with $d^{\omega',2}_{2,t_0+t_1}$ as residual service time. The policy $\omega$ performs a hand-off at $t_0$ and sends the flexible server to the second station with a residual service time of $f^{\omega,2}_{2,t_0}$ (here the residual service time is equal to the service time). Therefore at $t_0 + t_1$, $D^{\omega}(2) < D^{\omega'}(2)$.

**Inductive step**:

If $t_0$ has occurred in the basis step, Lemma 2 shows that for each case considered in the inductive step, the system respects the *ahead* property. Hence $\omega'$ will not have departures occurring sooner than $\omega$. If time $t_0$ is not reached, the two systems remain the same and there will be nothing to prove at this stage.

If time $t_0$ occurs after the $n-1^{st}$ departure but before the $n^{th}$ departure, consider this configuration: a dedicated blocked server (serving job $k$) and a flexible busy server (serving job $k+1$) at

the first station and a dedicated busy server (serving job $p$) at the second station.

**Case 1:** Assume $d^2_{p,t_0} > f^1_{k+1,t_0}$, meaning that the flexible server completes its service before the dedicated server. Let $t_1 = f^1_{k+1,t_0}$, assume $\sigma^1_d$ is holding the $k^{th}$ job, and $n-1$ departures from the first station have occurred. The policy $\omega'$ waits $t_1$ time units until $\sigma^1_f$ completes service and sends the server to the second station with $f^{\omega',2}_{k+1,t_0+t_1}$ as residual service time. (To simplify the argument, we assume that at $t_0+t_1$ a hand-off occurs and the $k^{th}$ job is sent to the second station instead of the $k+1^{st}$ job and hence $\sigma^2_f$ should serve a job with service time $f^2_{k,t_0+t_1}$. This can be done as the jobs are indistinguishable). The policy $\omega$ performs a hand-off at $t_0$ and sends the flexible server to the second station with a service time of $f^{\omega,2}_{k,t_0}$. Therefore at $t_0+t_1$, $D^{\omega}(n) < D^{\omega'}(n)$.
Considering the system at $t_0+t_1$, the following cases can happen. Note that $\sigma^1_d$ is blocked at $t_0+t_1$ under both policies, holding job $k+1$.

**Case 1.1:** Assume $f^{\omega,2}_{k,t_0} - t_1 < d^2_{p,t_0+t_1} < f^{\omega',2}_{k,t_0+t_1}$. Let $t_2 = f^{\omega,2}_{k,t_0} - t_1$. Then $D^{\omega}(n+1) = t_0+t_1+t_2$ and $D^{\omega'}(n+1) = t_0+t_1+d^{\omega',2}_{p,t_0+t_1}$. Therefore $D^{\omega}(n+1) < D^{\omega'}(n+1)$. Figure 1 illustrates this case.

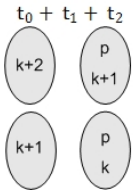


Figure 1: Case 1.1

**Case 1.2:** Assume $d^2_{p,t_0+t_1} < f^{\omega,2}_{k,t_0} - t_1$. Let $t_2 = d^2_{p,t_0+t_1}$. Therefore $D^{\omega}(n+1) = D^{\omega'}(n+1) = t_0+t_1+t_2$. At $t_0+t_1+t_2$, $\sigma^1_d$ is serving job $k+2$ and $\sigma^2_d$ is serving job $k+1$ under both policies. Under $\omega$ the residual service time of job $k$ is $f^{\omega,2}_{k,t_0} - t_1 - t_2$ and under $\omega'$ it is $f^{\omega',2}_{k,t_0+t_1} - t_2$. Figure 2 illustrates this case. We further divide this case.

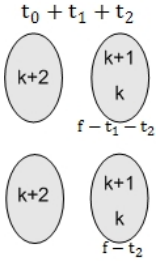


Figure 2: Case 1.2

**Case 1.2.1:** Assume $d^1_{k+2,t_0+t_1+t_2} < \min\{d^2_{k+1,t_0+t_1+t_2}, f^{\omega,2}_{k,t_0} - t_1 - t_2\}$. We divide this case further.

**Case 1.2.1.1:** Further assume $f^{\omega,2}_{k,t_0} - t_1 - t_2 < d^2_{k+1,t_0+t_1+t_2}$. Assumption 1.2.1 implies that job $k+2$ is blocked at time $(t_0 + t_1 + t_2 + d^1_{k+2,t_0+t_1+t_2})$. Therefore $D^{\omega}(n+2) < D^{\omega'}(n+2)$.

**Case 1.2.1.2:** Now assume $d^2_{k+1,t_0+t_1+t_2} < f^{\omega,2}_{k,t_0} - t_1 - t_2$. Let $t_3 = d^2_{k+1,t_0+t_1+t_2}$. Therefore $D^{\omega}(n+2) = D^{\omega'}(n+2) = t_0 + t_1 + t_2 + t_3$. At $t_0 + t_1 + t_2 + t_3$, under both policies, $\sigma^1_d$ is serving job $k+3$ and $\sigma^2_d$ is serving job $k+2$. Under $\omega$ the residual service time of job $k$ is $f^{\omega,2}_{k,t_0} - t_1 - t_2 - t_3$ and under $\omega'$ it is $f^{\omega',2}_{k,t_0+t_1} - t_2 - t_3$. Figure 3 illustrates this case. This is similar to Case 1.2. Although this reference seems circular, the fact that job $k+2$ eventually leaves the system avoids this.

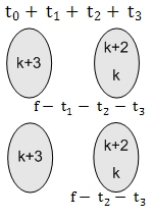


Figure 3: Case 1.2.1.2

**Case 1.2.1.3:** Now assume $d^2_{k+1,t_0+t_1+t_2} = f^{\omega,2}_{k,t_0} - t_1 - t_2$. This is similar to Case 1.2.2.1, discussed in the online supplement.

The remaining cases in the inductive step are provided in the online supplement.

(Lemma 1) □

*2.3. Non-idling property*

In Theorem 1 we proved that an optimal policy uses hand-off whenever possible. Now in Theorem 2, we aim to show that an optimal policy avoids idling the flexible server. In the proof of Theorem 2, we will use the results of Theorem 1, i.e. the two policies used in the proof perform hand-offs whenever possible. Having said that, we only claim non-idling to be a property of an optimal policy if hand-off is employed. Otherwise there might be cases where idling leads to better results.

For example, assume at time $t_0$, jobs $k$ and $k+1$ are being served by dedicated servers at the first and second stations, respectively, and that policies do not perform hand-off. At $t_0$, a non-idling pol-

icy assigns the flexible server to the first station to serve job $k+2$. An idling policy lets the flexible server idle at $t_0$. Now assume $\sigma_d^1$ completes serving its job sooner than the other servers. At $t_0+d_{k,t_0}^1$, under the non-idling policy $\sigma_d^1$ (job $k$) becomes blocked. Under the idling policy the flexible server is assigned to the second station to admit job $k$ and $\sigma_d^1$ admits job $k+2$. Hence, assuming there have been $n-1$ departures from the first station for both policies, $D^{idling}(n) < D^{non\text{-}idling}(n)$ which means idling the flexible server leads to better results. As a result of this observation, our induction method in the proof of Theorem 2 below can not be used (when hand-off is not assumed).

**Theorem 2**: *The optimal policy is non-idling under hand-off.*

**Proof**: Let $\pi'$ and $\pi$ be non-idling policies, i.e. whenever the flexible server becomes idle, the policy assigns it to:

1. the second station, if the first station is blocked;
2. the first station, otherwise.

The only exception is that $\pi'$ idles the flexible server once for only $t$ time units starting from $t_0$.

Only two other non-idling policies exist. The first is a policy that assigns the idle flexible server to the first station, if the first station is blocked. But as hand-off is used, the flexible server performs a hand-off and takes the blocked job to the second station, resulting in the same behavior as $\pi$. The second policy assigns the idle flexible server to the second station, if the first station is not blocked. Again, as hand-off is used, the flexible server passes its job to the free dedicated server in the second station and admits a new job at the first station, leading to the same behavior as $\pi$. Therefore we only need to consider $\pi$ and $\pi'$ throughout the rest of the proof.

There are three scenarios under which idling is possible:

1. a blocked server at the first station and a busy server at the second station
2. two busy servers, one at each station
3. one busy server at the first station

The proof requires explicit, careful, exhaustive evaluation of all possibilities. Due to space limitations, we do not include the details of the proof for this theorem. However the proof is similar to the proof of Theorem 1. A lemma is needed for each of the three scenarios above to do the induction over departures. A fourth lemma is required to show that the *ahead* property holds

when these three lemmas are used (i.e. after the next departure). The proof of Theorem 2 includes more cases than Theorem 1. The reason is that in addition to comparing the residual service times of jobs to each other, we need to compare them at an extra point in time, $t_0 + t$ (defined above).

(Theorem 2) □

A natural extension of Theorem 2 is as follows: introducing idling to any non-idling policy, decreases the throughput, assuming hand-off is used. If hand-off is not used, a non-idling policy can lead to better results, as discussed at the beginning of Section 2.3. Theorem 2 shows that after the idling period ($[t_0, t_0 + t]$), the system is *ahead* under the policy which is always non-idling. It also shows that if a non-idling policy is applied to the system, the *ahead* property still holds.

To investigate the proposed extension, we could follow an approach similar to the proof of Theorem 2. Given a policy with idling, we produce another policy by making it non-idling for a period of time. Theorem 2 shows that after this period, the system is *ahead* under the modified policy. Conceptually, we believe the fact that hand-off can be employed, makes it impossible for an idling policy to provide a higher throughput than a non-idling policy.

### *2.4. Extensions*

In this subsection, we state that the results of Theorems 1 and 2 can be extended to systems with arrivals, clearing systems, and systems with additional dedicated or flexible servers. The proofs of the following corollaries are presented in the online supplement.

**Corollary 1**: *Theorems 1 and 2 hold for i) systems where jobs are waiting at the first station, ii) systems with arrivals, and iii) clearing systems.*

**Corollary 2**: *Theorems 1 and 2 and Corollary 1 hold for arbitrary numbers of dedicated and flexible servers.*

In this section we proved that an optimal policy performs hand-off and does not idle. In the next section, we model tandem lines with two stations and determine the optimal allocation policy. The properties proven in Section 2 will be used to reduce the size of the action and state spaces.

## 3. Markov Decision Process Model

We use a Markov Decision Process (MDP) to model the above problem. For our controlled continuous-time Markov chain (CTMC), let $S, A$, and $A_s$ represent the state space, the action space, and the action space conditioned on the Markov chain being in state $s \in S$, respectively.

A state $s \in S$ denotes a tuple

$$s = (x_1, y_1, \ \ldots, \ x_i, y_i, \ \ldots, \ x_N) \tag{1}$$

where $x_i$ is the number of busy servers in station $i$ and $y_i$ is the number of blocked servers in station $i$. Note that we do not need to include $y_N$ in the state, as no servers can be blocked at the last station. We uniformize the CTMC (see Lippman [13]) to convert the continuous time problem to discrete time. The normalization constant that we will employ is denoted by $q$, and is defined below.

Let $v_i$ be the number of dedicated servers at station $i$. Using the non-idling and hand-off properties of optimal policies, the number of flexible servers at station $i$ is $x_i + y_i - v_i$. Hence the state (1) also determines the locations of the flexible servers.

Constructing the transition matrices is a two stage process. One needs to start from the initial state ($s$) and follow possible actions ($a$) to determine the new states ($s'$). State $s'$ is an intermediate state which does not appear in the transition matrix. The transition between $s$ and $s'$ is immediate. From there, it is possible to follow further transitions and reach new states ($s''$) with the probabilities defined in the transition matrix ($P_a$). We have:

$$s \xrightarrow{a} s' \xrightarrow{P_a} s''$$

that is reflected in the transition matrix as $P_a(s, s'') = \frac{\mu}{q}$ where $q = \max_{s \in S} \sum_{s'' \in \{S-s\}} P_a(s, s'')$ and $\mu$ is the transition rate from $s'$ to $s''$.

The size of the action space is $|A| = \binom{F+N-1}{N-1}$. An action $a \in A$ is denoted by a tuple

$$a = (a_1, \ldots, a_j, \ldots, a_N)$$

where $a_j$ is the number of flexible servers assigned to station $j$, $0 \leq a_j \leq F$, and $\sum_{j=1}^{N} a_j = F$.

*3.1. Generic MDP*

We begin by specializing to the case of $N = 2, F = 1, v_1 = I, v_2 = J$. We intend to apply the Policy Iteration (PI) algorithm to find the optimal policy.

**States:**

Using the state description in (1), the state space is as follows:

$S =$
$\{(I+1,0,j), 0 \leq j < J,$
$(I+1,0,J),$
$(I,0,J+1),$
$(i,b,J+1), i,b>0, i+b=I,$
$(i,b,J), i \geq 0, b>0, i+b=I,$
$(0,I,J+1)\}$

Our state space is constrained so that hand-off is used whenever possible (as we showed in Theorem 1 that an optimal policy performs hand-off). If we made it optional to employ hand-off, additional states (such as $(I,1,J)$ and $(i,b,J), i,b>0, i+b=I+1$) would need to be introduced.

**Actions:**

The action space is as follows:

$$A = \{a_1, a_2\}$$

where $a_i$ means moving the flexible server to the $i^{th}$ station. The permissible actions are given by

$$A_s = \begin{cases} \{a_1, a_2\} & \text{for } s = (i,b,J) \\ \{a_1\} & \text{for } s = (I+1,0,j), (I+1,0,J) \\ \{a_2\} & \text{for } s = (I,0,J+1), (i,b,J+1), (0,I,J+1) \end{cases}$$

where $0 < j < J$ and $i + b = I$.

In the Policy Iteration algorithm, we choose the initial policy $(d_0)$ to be:

$$d_0(s) = \begin{cases} a_1 & \text{for } s = (I+1,0,j), (I+1,0,J) \\ a_2 & \text{for } s = (i,b,J), (I,0,J+1), (i,b,J+1), (0,I,J+1), \end{cases}$$

where $i$, $j$, and $b$ are constrained so that $s$ is in the state space.

The reward gained from $d_0$ is:

$$r_{d_0}(s) = \frac{1}{q} \begin{cases} j\mu_2 & \text{for } s = (I+1,0,j) \\ J\mu_2 & \text{for } s = (I+1,0,J) \\ (J+1)\mu_2 & \text{for } s = (i,b,J), (I,0,J+1), (i,b,J+1), (0,I,J+1). \end{cases}$$

The reward function as stated here is a valid choice for maximizing the throughput, as shown in Section 3 of Andradóttir et al. [4].

**Transition Probabilities:**

The transition probabilities filtered by each of the actions and by the initial policy are as follows:

$$P_{a_1}(s,s'') = \frac{1}{q}\begin{cases} (I+1)\mu_1 & \text{for } s=(I+1,0,j), s''=(I+1,0,j+1), 0 \le j < J \\ j\mu_2 & \text{for } s=(I+1,0,j), s''=(I+1,0,j-1), 0 < j < J \\ q-(I+1)\mu_1 - j\mu_2 & \text{for } s=s''=(I+1,0,j), 0 \le j < J \\ (I+1)\mu_1 & \text{for } s=(I+1,0,J), s''=(I,0,J+1) \\ J\mu_2 & \text{for } s=(I+1,0,J), s''=(I+1,0,J-1) \\ q-(I+1)\mu_1 - J\mu_2 & \text{for } s=s''=(I+1,0,J) \\ (i+1)\mu_1 & \text{for } s=(i,b,J), s''=(i,b,J+1), i \ge 0, b > 0, i+b=I \\ (J+1)\mu_2 & \text{for } s=(i,b,J), s''=(i+1,b-1,J), i \ge 0, b > 0, i+b=I \\ q-(i+1)\mu_1 - (J+1)\mu_2 & \text{for } s=s''=(i,b,J), i \ge 0, b > 0, i+b=I \end{cases}$$

$$P_{a_2}(s,s'') = \frac{1}{q}\begin{cases} I\mu_1 & \text{for } s=(I,0,J+1), s''=(I-1,1,J+1) \\ (J+1)\mu_2 & \text{for } s=(I,0,J+1), s''=(I+1,0,J) \\ q-I\mu_1 - (J+1)\mu_2 & \text{for } s=s''=(I,0,J+1) \\ i\mu_1 & \text{for } s=(i,b,J+1), s''=(i-1,b+1,J+1), i,b>0, i+b=I \\ (J+1)\mu_2 & \text{for } s=(i,b,J+1), s''=(i,b,J), i,b>0, i+b=I \\ q-i\mu_1 - (J+1)\mu_2 & \text{for } s=s''=(i,b,J+1), i,b>0, i+b=I \\ (i+1)\mu_1 & \text{for } s=(i,b,J), s''=(i,b,J+1), i \ge 0, b>0, i+b=I \\ (J+1)\mu_2 & \text{for } s=(i,b,J), s''=(i+1,b-1,J), i \ge 0, b>0, i+b=I \\ q-(i+1)\mu_1 - (J+1)\mu_2 & \text{for } s=s''=(i,b,J), i \ge 0, b>0, i+b=I \\ (J+1)\mu_2 & \text{for } s=(0,I,J+1), s''=(0,I,J) \\ q-(J+1)\mu_2 & \text{for } s=s''=(0,I,J+1) \end{cases}$$

$$P_{d_0}(s,s'') = \frac{1}{q}\begin{cases}
(I+1)\mu_1 & \texttt{for } s=(I+1,0,j), s''=(I+1,0,j+1), 0 \le j < J \\
j\mu_2 & \texttt{for } s=(I+1,0,j), s''=(I+1,0,j-1), 0 < j < J \\
q-(I+1)\mu_1-j\mu_2 & \texttt{for } s=s''=(I+1,0,j), 0 \le j < J \\
(I+1)\mu_1 & \texttt{for } s=(I+1,0,J), s''=(I,0,J+1) \\
J\mu_2 & \texttt{for } s=(I+1,0,J), s''=(I+1,0,J-1) \\
q-(I+1)\mu_1-J\mu_2 & \texttt{for } s=s''=(I+1,0,J) \\
I\mu_1 & \texttt{for } s=(I,0,J+1), s''=(I-1,1,J+1) \\
(J+1)\mu_2 & \texttt{for } s=(I,0,J+1), s''=(I+1,0,J) \\
q-I\mu_1-(J+1)\mu_2 & \texttt{for } s=s''=(I,0,J+1) \\
i\mu_1 & \texttt{for } s=(i,b,J+1), s''=(i-1,b+1,J+1), i,b>0, i+b=I \\
(J+1)\mu_2 & \texttt{for } s=(i,b,J+1), s''=(i,b,J), i,b>0, i+b=I \\
q-i\mu_1-(J+1)\mu_2 & \texttt{for } s=s''=(i,b,J+1), i,b>0, i+b=I \\
(i+1)\mu_1 & \texttt{for } s=(i,b,J), s''=(i,b,J+1), i\ge 0, b>0, i+b=I \\
(J+1)\mu_2 & \texttt{for } s=(i,b,J), s''=(i+1,b-1,J), i\ge 0, b>0, i+b=I \\
q-(i+1)\mu_1-(J+1)\mu_2 & \texttt{for } s=s''=(i,b,J), i\ge 0, b>0, i+b=I \\
(J+1)\mu_2 & \texttt{for } s=(0,I,J+1), s''=(0,I,J) \\
q-(J+1)\mu_2 & \texttt{for } s=s''=(0,I,J+1)
\end{cases}$$

where $q$ is the normalization factor defined as $q = I\mu_1 + J\mu_2 + max(\mu_1, \mu_2)$.

In the Policy Iteration algorithm we solve

$$r_{d_0} - g_0 e + (P_{d_0} - I_n)h_0 = 0,$$

subject to $h(1,0,0) = 0$ with respect to $g_0$ and $h_0$. Note that $e$ is a column vector of ones and $I_n$ is the $n$-dimensional identity matrix, where $n = |S|$. Then, to prove the optimality of $d_0$, we need to show $d_1(s) = d_0(s)$, where:

$$d_1(s) = \text{argmax}_{a\in A_s}\{r(s,a) + \sum_{j\in S} p(j|s,a)h_0(j)\}, \forall s \in S,$$

is the result of one iteration of the Policy Iteration algorithm. This is equivalent to showing that for all $s$

$$r(s,a) + \sum_{j\in S} p(j|s,a)h_0(j) - \left( r(s,d_0(s)) + \sum_{j\in S} p(j|s,d_0(s))h_0(j) \right) \le 0. \tag{2}$$

In inequality (2) given that $s \xrightarrow{a} s' \xrightarrow{P_a} s''$, $r(s,a) = x'_N \mu_N$ where $x'_N$ appears in the representation of state $s'$ according to (1) and $\mu_N = P_a(s,s'')$. Note that $s'$ is also a function of $s$ and $a$.

We implemented the Policy Iteration algorithm in MATLAB and intended to apply it to the stated generic MDP. However it is not an easy task to model the generic MDP in MATLAB as each of the state space members represents a set of states. This fact becomes problematic when matrices are to be constructed. We have not been able to perform Policy Iteration for the generic MDP. Therefore we observed several configurations with $N = 2, F = 1$ and specific values for the numbers of dedicated servers. For example we set $(I, J)$ to $\{(1,1),(1,2),(2,3),(4,2),(2,4),(3,3),(3,4),(7,3)\}$.

The result of the Policy Iteration algorithm (inequality (2)) is an expression in terms of service rates. We tried to verify the inequality for generic cases using induction on $I$ and $J$. We have not been able to find such a relation to this point. However we could prove that the expression holds for arbitrary service rates, for every specific pair of values $I$ and $J$ that we considered. So, we have the following conjecture, which has been verified for the values of $I$ and $J$ described above.

**Conjecture 1.** The optimal policy resulting from these observations is the policy that *clears blocking whenever possible and assigns the flexible server to the first station otherwise.*

Comparing this policy with allocation policies for collaborative servers, Andradóttir et al. [6] (Lemma 4.1) show that for a tandem queue with two stations, a finite buffer, a dedicated server at each station, and a flexible server, the optimal policy is threshold type. The optimal policy assigns the flexible server based on the number of jobs in the system (both in service and in the buffer). The optimal policy sends the flexible server to the second station, when there is a busy dedicated server at each station (even if there are no blocked servers). This difference with our optimal policy occurs due to the server collaboration assumption in [6].

### *3.2. Sample MDP*

In order to show how we applied the Policy Iteration algorithm, in this Section we give details for a specific example with $I = 3, J = 2, F = 1$. To be able to construct the discrete-time MDP, we choose the following normalization factor: $q = 2\mu_1 + 3\mu_2 + max(\mu_1, \mu_2)$. Here,

$$A = \{a_1, a_2\}$$

$$S = \{(3,0,3),(3,0,2),(3,0,1),(3,0,0),(2,1,3),(2,0,3),(1,2,3),(1,1,4),(1,1,3),(0,2,4),(0,2,3)\}$$

$$A_s = \begin{cases} \{a_1, a_2\} & \text{for } s = (2,1,3), (1,2,3), (1,1,3), (0,2,3) \\ \{a_1\} & \text{for } s = (3,0,0), (3,0,1), (3,0,2), (3,0,3), (2,0,3) \\ \{a_2\} & \text{for } s = (1,1,4), (0,2,4) \end{cases}$$

Our candidate optimal policy $d_0$ is:

$$d_0(s) = \begin{cases} a_1 & \text{for } s = (3,0,0), (3,0,1), (3,0,2), (3,0,3), (2,0,3) \\ a_2 & \text{for } s = (2,1,3), (1,2,3), (1,1,4), (1,1,3), (0,2,4), (0,2,3) \end{cases}$$

$$r_{d_0}(s) = \begin{cases} 4\mu_2 & \text{for } s = (2,1,3), (1,2,3), (1,1,4), (1,1,3), (0,2,4), (0,2,3) \\ 3\mu_2 & \text{for } s = (3,0,3), (2,0,3) \\ 2\mu_2 & \text{for } s = (3,0,2) \\ \mu_2 & \text{for } s = (3,0,1) \\ 0 & \text{for } s = (3,0,0) \end{cases}$$

For the following matrices, assume the following state ordering: (3, 0, 3), (3, 0, 2), (3, 0, 1), (3, 0, 0), (2, 0, 3), (1, 2, 3), (1, 1, 4), (1, 1, 3), (0, 2, 4), (0, 2, 3).

$$P_{a_1}(s, s'') = \frac{1}{q} \begin{pmatrix} q_{1,a_1} & 3\mu_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 3\mu_1 & q_{2,a_1} & 2\mu_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3\mu_1 & q_{3,a_1} & \mu_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3\mu_1 & q_{4,a_1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 3\mu_2 & 0 & 0 & 0 & q_{5,a_1} & 0 & 0 & 0 & 0 & 0 & 2\mu_1 \\ 0 & 3\mu_2 & 0 & 0 & 3\mu_1 & q_{6,a_1} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3\mu_2 & 0 & q_{7,a_1} & 0 & 0 & \mu_1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & q_{8,a_1} & 0 & 0 & 0 \\ 3\mu_2 & 0 & 0 & 0 & 0 & 0 & 2\mu_1 & 0 & q_{9,a_1} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & q_{10,a_1} & 0 \\ 0 & 0 & 0 & 0 & 3\mu_2 & 0 & 0 & 0 & 0 & \mu_1 & q_{11,a_1} \end{pmatrix}$$

$$P_{a_2}(s, s'') = \frac{1}{q} \begin{pmatrix} q_{1,a_2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & q_{2,a_2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & q_{3,a_2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & q_{4,a_2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & q_{5,a_2} & 4\mu_2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & q_{6,a_2} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & q_{7,a_2} & 0 & 4\mu_2 & \mu_1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & q_{8,a_2} & 4\mu_2 & \mu_1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 4\mu_2 & 0 & 2\mu_1 & q_{9,a_2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & q_{10,a_2} & 4\mu_2 \\ 0 & 0 & 0 & 0 & 3\mu_2 & 0 & 0 & 0 & 4\mu_2 & \mu_1 & q_{11,a_2} \end{pmatrix}$$

where $q_{i,a} = q - \sum_{j=1 \wedge j \neq i}^{|S|} P_a(i,j)$ and $a \in A$.

Now we want to show inequality (2) to verify that $d_0(s)$ is optimal. Assume $\mu_1 > \mu_2$. Then $q = 3(\mu_1 + \mu_2)$. Let

$$A = 27\mu_1^6 + 108\mu_1^5\mu_2 + 216\mu_1^4\mu_2^2 + 288\mu_1^3\mu_2^3 + 288\mu_1^2\mu_2^4 + 192\mu_1\mu_2^5 + 64\mu_2^6.$$

Then

$$g_0 = \frac{12(9\mu_1^6\mu_2 + 36\mu_1^5\mu_2^2 + 72\mu_1^4\mu_2^3 + 72\mu_1^3\mu_2^4 + 48\mu_1^2\mu_2^5 + 16\mu_1\mu_2^6)}{A}.$$

For example for $s = (2, 0, 3)$,

$$h_0(s) = \frac{q(9\mu_1^5\mu_2 + 54\mu_1^4\mu_2^2 + 162\mu_1^3\mu_2^3 + 224\mu_1^2\mu_2^4 + 176\mu_1\mu_2^5 + 64\mu_2^6)}{A}$$

and inequality (2) simplifies to checking

$$16\mu_2^6\mu_1 + 80\mu_2^5\mu_1^2 + 190\mu_2^4\mu_1^3 + 288\mu_2^3\mu_1^4 + 261\mu_2^2\mu_1^5 + 126\mu_2\mu_1^6 + 27\mu_1^7 > 0$$

which is trivially true. If $\mu_1 < \mu_2$, with similar calculations inequality (2) simplifies to verifying

$$32\mu_2^6\mu_1 + 144\mu_2^5\mu_1^2 + 316\mu_2^4\mu_1^3 + 450\mu_2^3\mu_1^4 + 360\mu_2^2\mu_1^5 + 153\mu_2\mu_1^6 + 27\mu_1^7 > 0,$$

which is also trivially true. Similarly it can be shown that inequality (2) holds for all other states. In this example $d_0$ makes the Policy Iteration algorithm converge in one step. Hence $d_0$ is the optimal policy. It should be noted that the policy is independent of $\mu_1$ and $\mu_2$.

## 4. Larger systems

With two stations, only one station can be blocked. As we saw in Section 3, the primary task of the optimal policy is to clear blocking. When the number of stations is increased, blocking can occur in more than one station, and policies should prioritize the order in which blocked servers are cleared. Also, starvation might occur in the system. Starvation occurs when a dedicated server becomes free and has no jobs to process. To study these effects, in this section we apply the MDP modelling and Policy Iteration algorithm to longer tandem lines, and use these results to provide heuristics. Similar to Section 3, the policies that we consider are non-idling and perform hand-off whenever possible (which are properties of optimal policies as shown by Theorems 1 and 2).

To begin, we study tandem lines where stations have equal mean service times, have a dedicated server per station, and have one flexible server. Solving the MDP for 3, 4, and 5 stations suggests the following near-optimal policy: *i) clear blocking from the end to the beginning unless it causes starvation; ii) if clearing a blocked server would cause starvation, admit a new job.*

With 3 stations there is no state from the optimal policy, obtained from policy iteration, under which clearing blocking results in starvation. With 4 stations, under the optimal policy, these states cause starvation: (1,0, 1,0, 0,1, 1), (1,0, 0,0, 0,1, 1), (0,1, 1,0, 0,1, 1). In these states the optimal policy admits a new job, which is compliant with the suggested policy stated in the previous paragraph. However, while the following states also cause starvation, clearing a blocked server is preferred: (1,0, 0,1, 1,0, 1) and (1,0, 0,1, 0,1, 1) (as opposed to the suggested policy).

For 5 stations the suggested policy offers the optimal allocations in most cases. The number of states is 236 from which the suggested policy results in the optimal allocation in 228 of them. With 5 stations, the following states cause starvation: (1,0, 1,0, 0,0, 0,1, 1), (1,0, 0,0, 1,0, 0,1, 1), (1,0, 0,0, 0,1, 1,1, 1), (1,0, 0,0, 0,1, 1,0, 1), (1,0, 0,0, 0,0, 0,1, 1), (0,1, 1,0, 1,0, 0,1, 1), (0,1, 1,0, 0,1, 1,0, 1), (0,1, 1,0, 0,0, 1,0, 2), (0,1, 0,1, 1,0, 0,1, 1). In all these states avoiding starvation is favoured over clearing blocking. However while the following states cause starvation, clearing blocking is preferred: (1,0, 1,0, 1,0, 0,1, 1), (1,0, 1,0, 0,1, 1,0, 1), (1,0, 0,1, 1,0, 1,0, 1), (1,0, 0,1, 1,0, 1,0, 0), (1,0, 0,1, 1,0, 0,0, 1).

Also in the following three states blocking is not cleared and starvation does not occur: (1,0, 1,0, 1,0, 1,1, 1), (1,0, 0,0, 0,1, 1,1, 1), (0,1, 0,1, 0,1, 0,1, 1). The exceptions above suggest that while an optimal policy does not have a simple structure, a near-optimal policy exists as stated above.

Note that as $N$ increases, the number of states increases exponentially, which makes the overhead of constructing the MDP prohibitive. For example when $N = 5$ (with the configuration introduced above), there are 5 actions, 236 states, and $P_{a_1}$, $P_{a_2}$, $P_{a_3}$, $P_{a_4}$, and $P_{a_5}$ have 498, 496, 462, 447, and 413 non-zero entries, respectively. Calculating each of these entries is difficult to automate, and hence is error-prone. In Section 5 we will use simulation to show that if the policy stated in the beginning of this section is employed, near-optimal results are gained for different configurations.

## 5. Numerical results

A number of policies and configurations are considered here. The first set of configurations (Tables 1, 2, and 3) has $v_i = 1, \mu_i = 1, i = 1, \ldots, N$, and $F = 1$. $N$ ranges from 4 to 30. The second set of configurations (Tables 4, 5, and 6) deals with heterogeneous service rates and a single server per station. We let $W = (w_1, \ldots, w_i, \ldots, w_N)$ be the vector of mean service times, where $w_i$ is the mean service time at the $i^{th}$ station.

The policies we consider can be distinguished by the way they treat blocking and starvation and how they employ hand-off. A policy should determine the order in which it clears blocking. It should specify how it deals with the trade-off between blocking and starvation. The policy also describes if it uses hand-off or not. While Theorem 1 suggests that hand-off should be used, here we also consider policies that do not employ hand-off. We intend to show how the suggested heuristic works and also study the impact of allowing hand-offs.

In all policies if there is no blocking in the system and free flexible servers exist, flexible servers are sent to the first station to admit a new job. This is consistent with the necessity of non-idling for optimality.

The policies we studied are as follows:

- Policy 1: clears blocking from end to beginning only if it does not cause starvation in the $\lfloor \frac{2}{3}N \rfloor$ previous stations (derived experimentally); uses hand-off
- Policy 2: ignores blocking and admits new jobs when possible; uses hand-off
- Policy 3: clears blocking from end to beginning only if it does not cause starvation; uses hand-off
- Policy 4: clears blocking in order of descending service rates only if it does not cause starvation; uses hand-off
- Policy 5: clears blocking in order of descending service rates; uses hand-off
- Policy 6: clears blocking from beginning to end; uses hand-off
- Policy 7: clears blocking from end to beginning; uses hand-off
- Policy 8: clears blocking from beginning to end; does not use hand-off

Policies 4 and 5 prioritize clearing blocking caused by the station with the highest mean service time (among the stations that cause blocking). Policies 4 and 5 are applied only on the second set of configurations. In practice, Policies 2, 3, and 4 show similar behavior as Policies 3 and 4 clear a blocked server only if all stations preceding a blocked station are blocked. Hand-off in Policy 2 pushes a flexible server through all consecutive blocked stations. Simulation results reveal that Policies 1 and 4 perform better than the other policies, depending on the configuration considered. In Policy 1, the $\lfloor \frac{2}{3}N \rfloor$ parameter gives higher priority to avoiding starvation, compared to Policies 2, 3, and 4. This parameter is derived experimentally and is used to ignore a blocked server, if clearing that blocking causes simultaneous starvations in a large number of stations.

We applied the above policies to several configurations (defined below) and measured throughput in a simulation environment. Each simulation is a long run, in terms of the number of departures from the system. The run is long enough (100 million departures) so that the effect of starting the system from an empty state becomes negligible. In the following tables policies are ordered based on their throughputs.

### *5.1. Homogeneous mean service times (the first set)*

Table 1 compares the throughputs of configurations with one flexible server, $N \in \{4, 5, 8, 15, 30\}$ stations where the service time of each station is exponentially distributed with rate one, and one dedicated server at each station. Policy 1 is close to what the Policy Iteration algorithm suggested and results in the best throughput among these policies. It can be observed that Policies 1 and 2 lead to identical throughputs. This is because when a station and all its preceding stations are blocked, Policy 1 clears blocking from the last station in the sequence of blocked stations and Policy 2 uses hand-off to pass the flexible server to the last station in the sequence of blocked stations. Although the throughputs are identical, Policy 2 needs to perform extra hand-offs compared to Policy 1.

| **N** | **Policy1** | **Policy2** | **Policy3** | **Policy6** | **Policy7** | **Policy8** | $F = 0$ |
|---|---|---|---|---|---|---|---|
| 4 | 0.93248 | 0.93248 | 0.93065 | 0.92021 | 0.91869 | 0.78592 | 0.59596 |
| 5 | 0.83049 | 0.83049 | 0.82448 | 0.81400 | 0.81032 | 0.66214 | 0.53038 |
| 8 | 0.66720 | 0.66720 | 0.64631 | 0.64298 | 0.62598 | 0.47267 | 0.41827 |
| 15 | 0.51520 | 0.51520 | 0.47234 | 0.50238 | 0.44993 | 0.33768 | 0.32384 |
| 30 | 0.40490 | 0.40490 | 0.34292 | 0.40149 | 0.32791 | 0.26023 | 0.25696 |

Table 1: Throughput for configurations with $i = 1, \ldots, N, w_i = 1$, and $F = 1$

Comparing Policies 1 and 3, Policy 1 allows starvation in the initial stations, while Policy 3 avoids it in all stations. Policy 1 has better performance than Policy 3. The reason is that the downside of having starvation in the initial stations is less than the benefit of clearing blocked servers at the end stations. There are two explanations: i) starvation closer to the first station is likely to be resolved more quickly compared to starvation in the end stations; ii) starvation at the initial stations, when the downstream stations are busy, does not affect the downstream stations. Hence in Policy 1, the trade-off between clearing blocking and avoiding starvation (in the initial stations) is resolved in favour of clearing blocking.

To illustrate the effect of including flexible servers, in the last column of Table 1 we show the case where there are no flexible servers and the extra server is assigned to one of the middle stations. It can be observed that making a server flexible can lead to throughput improvements up to 36.08%.

We also study configurations that include service time distributions with coefficients of variation other than 1. In simulations, an Erlang distribution is used for coefficients of variation less than one and a Hyper-exponential distribution for coefficients of variation greater than one. Table 2 is similar to Table 1 except that the coefficients of variation of all stations are 0.5. Table 3 is similar to Table 1 except that the coefficients of variation are 1.34. The results in both cases are almost the same as the case with coefficients of variation equal to 1 for all stations.

Let $cv_i$ be the coefficient of variation of the $i^{th}$ station and $cv$ be the vector of coefficients of variation. We simulated a number of configurations with different coefficients of variation and concluded the same results as the case with coefficients of variation equal to 1 for all stations. More specifically we tested the following configurations: $N = 4, cv_3 = 0.5, 1.34$; $N = 5, cv_2 = 0.5$; $N = 5, cv_4 = 1.34$; $N = 8, cv_3 = 0.5, cv_7 = 1.34$; $N = 15, cv_3 = cv_{10} = 0.5, cv_7 = cv_{13} = 1.34$; $N = 30, cv_3 = cv_{10} = cv_{18} = cv_{25} = 0.5, cv_7 = cv_{13} = cv_{22} = cv_{28} = 1.34$. The unspecified coefficients of variation are equal to one.

| **N** | **Policy1** | **Policy2** | **Policy3** | **Policy6** | **Policy7** | **Policy8** |
|---|---|---|---|---|---|---|
| 4 | 0.99498 | 0.99498 | 0.99257 | 0.98428 | 0.98265 | 0.81297 |
| 5 | 0.90918 | 0.90918 | 0.90360 | 0.89475 | 0.84575 | 0.73190 |
| 8 | 0.77088 | 0.77088 | 0.75612 | 0.75361 | 0.73646 | 0.60107 |
| 15 | 0.64377 | 0.64377 | 0.61052 | 0.63681 | 0.58832 | 0.50407 |
| 30 | 0.54746 | 0.54746 | 0.50081 | 0.54584 | 0.48228 | 0.43911 |

Table 2: Throughput for configurations with $i = 1, \ldots, N, w_i = 1, cv_i = 0.5$, and $F = 1$

| N | Policy1 | Policy2 | Policy3 | Policy6 | Policy7 | Policy8 |
|---|---|---|---|---|---|---|
| 4 | 0.94136 | 0.94136 | 0.93919 | 0.92778 | 0.93919 | 0.79680 |
| 5 | 0.83372 | 0.83372 | 0.82736 | 0.81564 | 0.81244 | 0.66997 |
| 8 | 0.65759 | 0.65759 | 0.63753 | 0.63090 | 0.61551 | 0.46233 |
| 15 | 0.49317 | 0.49317 | 0.44717 | 0.47511 | 0.42999 | 0.29728 |
| 30 | 0.37423 | 0.37423 | 0.30740 | 0.36776 | 0.29448 | 0.19957 |

Table 3: Throughput for configurations with $i = 1, \ldots, N, w_i = 1, cv_i = 1.34$, and $F = 1$

*5.2. Heterogeneous mean service times (the second set)*

Table 4 compares the throughputs for configurations with various numbers of stations, different service rates, and one flexible server. Policies 4 and 5 are not applicable to the first set of configurations, but they are compared against other policies in the second set of configurations. Table 4 illustrates that all orders remain the same as Table 1, except that Policy 4 outperforms Policy 1 in certain cases. The intuition is that while for small systems, an optimal policy clears blocking from the end to the beginning, for large systems an optimal policy prefers to clear blocking for bottleneck stations first (i.e. stations with higher mean service times). For small systems, several adjacent blocked stations could deny admissions to the system. Therefore it is valuable to clear blocking from the end as Policy 1 does.

| $W$ | Policy4 | Policy1 | Policy2 | Policy5 | Policy7 |
|---|---|---|---|---|---|
| (1,3,6,9,5,4,2,3,7,10,5,2,1) | 0.10409 | 0.10409 | 0.10409 | 0.09940 | 0.09676 |
| (1,3,2,4,5,7,10,8,6,4,5,2,1) | 0.10286 | 0.10286 | 0.10286 | 0.09769 | 0.09635 |
| (7,5,6,4,2,3,1,3,2,4,5,9,8) | 0.10876 | 0.10872 | 0.10872 | 0.10448 | 0.10472 |
| (5,2,14,3,7,12,6,1,10) | 0.08403 | 0.08402 | 0.08402 | 0.08088 | 0.07996 |
| (9,8,7,6,5,4,3,2,1) | 0.11143 | 0.11143 | 0.11143 | 0.10508 | 0.10319 |

Table 4: Throughput for configurations with $F = 1$ and $i = 1, \ldots, N, cv_i = 1$

Table 5 is similar to Table 4 except that the coefficients of variation of all service times are 0.5. The results are almost the same as when the coefficients of variation are equal to 1. Table 6 is similar to Table 4 except that the coefficients of variation are greater than 1 for all stations (with different values). For $W = (1, 3, 6, 9, 5, 4, 2, 3, 7, 10, 5, 2, 1)$, $W = (1, 3, 2, 4, 5, 7, 10, 8, 6, 4, 5, 2, 1)$, and $W = (7, 5, 6, 4, 2, 3, 1, 3, 2, 4, 5, 9, 8)$ mean service time vectors, coefficients of variation vectors are $cv = (2.38, 2.31, 1.66, 1.27, 1.83, 2.06, 2.63, 2.31, 1.51, 1.17, 1.83, 2.63, 2.38)$, $cv = (2.38, 2.31, 2.63, 2.06, 1.83, 1.51, 1.17, 1.38, 1.66, 2.06, 1.83, 2.63, 2.38)$, and $cv = (1.51, 1.83, 1.66, 2.06, 2.63, 2.31, 2.38, 2.31, 2.63, 2.06, 1.83, 1.27, 1.38)$, respectively. Compared to the case with coefficients of variation equal to 1,

Policies 1 and 2 perform better than Policy 4. The intuition is that high variability increases the chance of blocking or starvation in adjacent stations.

| $W$ | **Policy4** | **Policy1** | **Policy2** | **Policy5** | **Policy7** |
|---|---|---|---|---|---|
| (1,3,6,9,5,4,2,3,7,10,5,2,1) | 0.11852 | 0.11848 | 0.11848 | 0.11628 | 0.11571 |
| (1,3,2,4,5,7,10,8,6,4,5,2,1) | 0.11734 | 0.11734 | 0.11734 | 0.11277 | 0.11080 |
| (7,5,6,4,2,3,1,3,2,4,5,9,8) | 0.12611 | 0.12492 | 0.12492 | 0.12488 | 0.12494 |

Table 5: Throughput for configurations with $F = 1$ and $i = 1, \ldots, N$, $cv_i = 0.5$

| $W$ | **Policy4** | **Policy1** | **Policy2** | **Policy5** | **Policy7** |
|---|---|---|---|---|---|
| (1,3,6,9,5,4,2,3,7,10,5,2,1) | 0.09882 | 0.09901 | 0.09901 | 0.09478 | 0.09609 |
| (1,3,2,4,5,7,10,8,6,4,5,2,1) | 0.09759 | 0.09760 | 0.09760 | 0.09307 | 0.09547 |
| (7,5,6,4,2,3,1,3,2,4,5,9,8) | 0.10031 | 0.10049 | 0.10049 | 0.09897 | 0.09798 |

Table 6: Throughput for configurations with $F = 1$ and $i = 1, \ldots, N$, $cv_i > 1$

Tables 2, 3, and 5 suggest that if coefficients of variation of all stations are changed uniformly, the ordering between policies remains the same as when coefficients of variation are 1. However Table 6 indicates that when coefficients of variation of stations are arbitrarily modified (in this case all greater than one), the ordering between policies might change. Finally we should comment that our results on coefficients of variation are not conclusive, in the sense that we have not considered vectors of coefficients of variation ($cv$) with more heterogeneous values.

## 6. Increasing the number of flexible servers

In this section we consider the effect of increasing $F$, the number of flexible servers. Assume we are given a system with a particular allocation of dedicated servers. We start to change dedicated servers to flexible servers until all servers are flexible. In other words, $F$ takes all values between 0 and $N$. An interesting question is how much throughput improves as $F$ increases.

### *6.1. Homogeneous W*

For equal workloads, we consider a tandem line with 15 stations where originally each station possesses one dedicated server. Service times at each station follow an exponential distribution with rate 1. The deployed server allocation policy is one that clears blocking from the end to the beginning of the line, unless there is blocking caused by a station with no dedicated servers. In that

case, the policy prioritizes allocating servers to the station with no dedicated server over clearing blocked servers from the end to the beginning. The policy also avoids starvation.

While increasing $F$, one should specify the order in which dedicated servers are changed to flexible servers. We contemplate four ordering policies: Policy i) choosing servers from the end to the beginning of the line; Policy ii) picking servers from the beginning to the end of the line; Policy iii) selecting servers around the middle station in a zig-zag way; Policy iv) choosing servers randomly. Simulations showed that Policies i and ii lead to similar results.

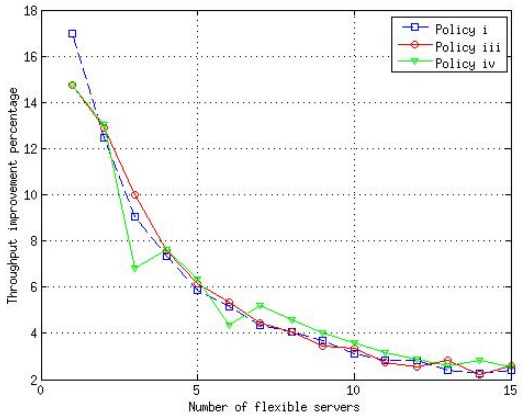


Figure 4: Throughput improvement (as a percentage) versus $F$ for $W = (1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1)$

Figure 4 contrasts Policies i and iii (for the 15 stations configuration described above) and shows that throughputs are fairly close for the two policies, but slightly higher for Policy i. Policy iv performs slightly worse than the two other policies, suggesting that choosing the first few servers is more sensitive to the employed policy, compared to choosing the last servers. It is worthwhile to note that, as can be seen in Figure 4, the highest improvements in throughput are gained from making the first few servers flexible. Also, as $F$ increases, the throughput improvement is diminished.

### *6.2. Heterogeneous W*

In addition to the equal workload case, we want to consider more generic configurations with heterogeneous service rates and multiple servers per station. We propose four ordering policies that specify the order in which dedicated servers are changed to flexible servers.

- Policy I. Changing a dedicated server of a station to a flexible server reduces the total service

rate of that station and can even make the station a bottleneck. Therefore to gain better results, the order in which flexible servers are chosen should depend on the mean total service rates of stations. As we intend to avoid introducing bottlenecks, consider the case where each station has one less dedicated server and then compare their total service rates. More specifically, start making servers flexible from the stations with the highest $\mu_i(v_i - 1)$ value. In case the values of $\mu_i(v_i - 1)$ are equal for more than one station, choose the one with the lowest $\mu_i$ first. See [18] for further details.

- Policy II. Start choosing servers from the station with the highest $v_i$ value. In case more than one station has the same value for $v_i$, choose the one with the lowest $\mu_i$.
- Policy III. Start picking servers from the station with the highest $\mu_i$ value regardless of the value of $v_i$.
- Policy IV. Select servers randomly.

Policy I follows a selection order that intends to avoid introducing new bottlenecks (if all $v_i$ and $\mu_i$ values are the same, introducing bottlenecks is inevitable). On the other hand, Policy II makes use of the server multiplicity effect [18]. This effect suggests that comparing two stations with equal total mean service times, the station with a higher number of servers leads to higher throughput. Policy II chooses the station that benefits most from this effect. Policy III is based on the fact that jobs could depart sooner from stations with higher service rates (when there is no blocking).

Figure 5 compares these four policies over a configuration with $W = (12, 7, 13, 3, 5, 4, 1, 10, 9)$ and an initial server allocation of (4, 5, 5, 2, 4, 2, 1, 4, 5). Figure 5 shows that the throughput improvement is dependent on the order that servers are chosen to become flexible. As the server allocation policy in the simulations, we employed Policy 4 (from Section 5) enhanced as follows: the policy prioritizes helping a bottleneck caused by a station with no dedicated servers.
We tried additional configurations to compare the performance of the proposed policies. Figure 6 illustrates the comparison for $W = (9, 7, 10, 3, 5, 4, 1, 13, 12)$ with an initial server allocation of (11, 7, 12, 3, 5, 4, 1, 15, 14). Figure 7 presents the comparison for $W = (1, 3, 4, 5, 7, 9, 10, 12, 13)$ with an initial server allocation of (3, 5, 6, 7, 9, 11, 12, 14, 15). Figures 5, 6, and 7 show that Policy I performs better than Policies II and III. The figures also illustrate that the highest throughput

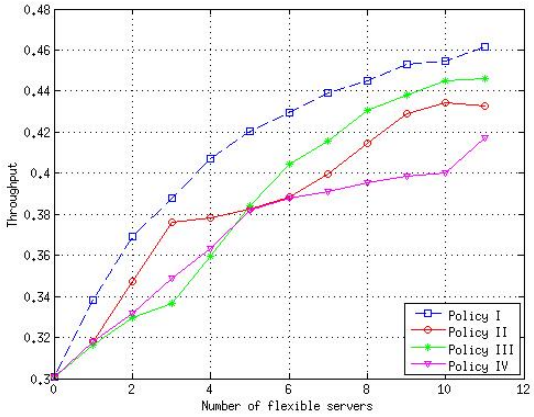


Figure 5: Throughput versus $F$ for $W = (12, 7, 13, 3, 5, 4, 1, 10, 9)$

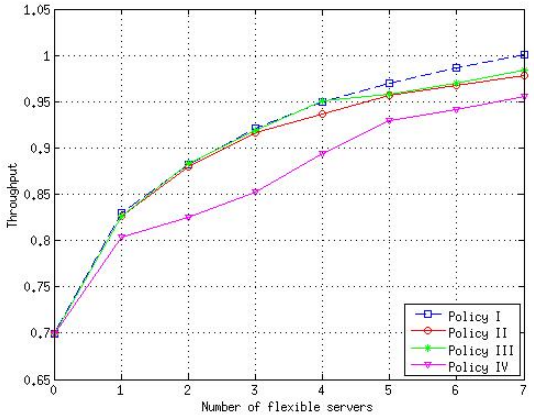


Figure 6: Throughput versus $F$ for $W = (9, 7, 10, 3, 5, 4, 1, 13, 12)$

improvements are gained when the first few servers are made flexible, and that as $F$ increases, the throughput improvement is diminished. Additionally, the performance of Policy IV in these figures shows that the ordering policy employed affects the throughput improvement values (and if not chosen well, results in lower improvements). The fact that in some cases, Policy IV performs better than Policies II and III, implies that Policies II and III, while intuitively reasonable, can be problematic to employ.

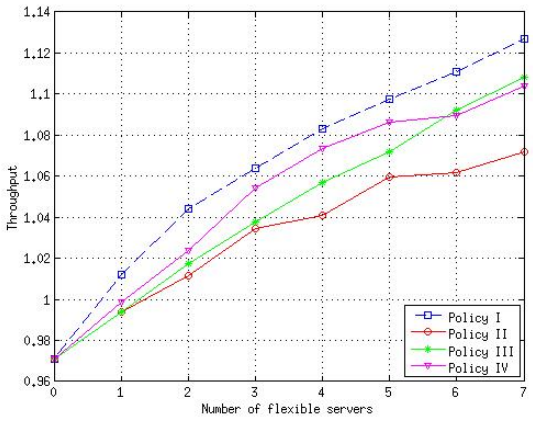


Figure 7: Throughput versus $F$ for $W = (1, 3, 4, 5, 7, 9, 10, 12, 13)$

## 7. Conclusion

In this paper we studied the allocation of flexible servers in zero buffer tandem lines in the presence of dedicated servers. We focused on configurations with only one flexible server. The optimal policy for two stations performs hand-off, clears blocking, and admits new jobs when there is no blocking. We used the Policy Iteration algorithm to show that for small systems with equal workloads, the optimal policy clears blocking from end to beginning and avoids starvation. Simulation results illustrated that for longer lines with varying workloads the optimal policy tends to prioritize clearing blocking from stations with higher mean service times. When making dedicated servers flexible, we observed that significantly higher throughput improvements were gained when the first few servers were made flexible. The improvement was diminished as the number of flexible servers increased.

In this work we have considered non-collaborative homogeneous servers. A natural extension is to study the case where servers are heterogeneous (i.e. have different service rates at different

stations) or can work collaboratively at a station. Another potential direction is to run the Policy Iteration algorithm for $F > 1$ to identify optimal allocation policies. With $F > 1$, the allocation problem is more challenging, as the allocation of a flexible server is dependent on the assignment of other flexible servers. The problem is more interesting when long lines are considered for $F > 1$, as more complicated collaboration among flexible servers is required. Considering costs for making a server flexible and then minimizing the overall cost and maximizing the throughput, is also of interest. It may also be of interest to introduce hand-off costs or server switching costs.

## References


[1] Ahn, H.-S., Duenyas, I., and Lewis, M. E. (2002) Optimal control of a two-stage tandem queuing system with flexible servers. *Probability in the Engineering and Informational Sciences*, **16**, 453–469.

[2] Ahn, H.-S., Duenyas, I., and Zhang, R. Q. (2004) Optimal control of a flexible server. *Advances in Applied Probability*, **36**, 139–170.

[3] Andradóttir, S. and Ayhan, H. (2005) Throughput maximization for tandem lines with two stations and flexible servers. *Operations Research*, **53**, 516–531.

[4] Andradóttir, S., Ayhan, H., and Down, D. G. (2001) Server assignment policies for maximizing the steady-state throughput of finite queueing systems. *Management Science*, **47**, 1421–1439.

[5] Andradóttir, S., Ayhan, H., and Down, D. G. (2003) Dynamic server allocation for queueing networks with flexible servers. *Operations Research*, **51**, 952–968.

[6] Andradóttir, S., Ayhan, H., and Down, D. G. (2007) Dynamic assignment of dedicated and flexible servers in tandem lines. *Probability in the Engineering and Informational Sciences*, **21**, 497–538.

[7] Avi-Itzhak, B. and Halfin, S. (1993) Servers in Tandem with Communication and Manufacturing Blocking. *Journal of Applied Probability*, **30**, 429–437.

[8] Department of Health and Aged Care (1999) Managing beds better - balancing supply and demand the NDHP-2 experience. Commonwealth of Australia, Canberra.

[9] Hasenbein, J. J. and Kim, B. (2011) Throughput maximization for two station tandem systems: a proof of the Andradóttir–Ayhan conjecture. *Queueing Syst. Theory Appl.*, **67**, 365–386.

[10] Hu, S., Ko, J., Weyand, L., ElMaraghy, H., Lien, T., Koren, Y., Bley, H., Chryssolouris, G., Nasr, N., and Shpitalni, M. (2011) Assembly system design and operations for product variety. *CIRP Annals - Manufacturing Technology*, **60**:715 – 733.

[11] Iravani, S. M. R. (1997) *Tandem queues attended by a moving server.* PhD thesis, University of Toronto.

[12] Kırkızlar, E., Andradóttir, S., and Ayhan, H. (2012) Flexible servers in understaffed tandem lines. *Production and Operations Management*, **21**, 761–777.

[13] Lippman, S. A. (1975) Applying a new device in the optimization of exponential queuing systems. *Operations Research*, **23**, 687–710.

[14] Pandelis, D. G. (2008) Optimal stochastic scheduling of two interconnected queues with varying service rates. *Operations Research Letters*, **36**, 492–495.

[15] van Vuuren, M., Adan, I. J., and Resing-Sassen, S. A. E. (2005) Performance analysis of multi-server tandem queues with finite buffers and blocking. *OR Spectrum*, **27**:315–338.

[16] Wu, C.-H., Down, D., and Lewis, M. (2008) Heuristics for allocation of reconfigurable resources in a serial line with reliability considerations. *IIE Transactions*, **40**, 595–611.

[17] Wu, C.-H., Lewis, M., and Veatch, M. (2006) Dynamic allocation of reconfigurable resources in a two-stage tandem queueing system with reliability considerations. *Automatic Control, IEEE Transactions on*, **51**, 309–314.

[18] Yarmand, M. H. and Down, D. G. (2013) Server allocation for zero buffer tandem queues. *European Journal of Operational Research*, **230**, 596–603.