Server allocation for zero buffer tandem queues

Mohammad H. Yarmand and Douglas G. Down

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

Abstract

In this paper we consider the problem of allocating servers to maximize throughput for tandem queues with no buffers. We propose an allocation method that assigns servers to stations based on the mean service times and the current number of servers assigned to each station. A number of simulations are run on different configurations to refine and verify the algorithm. The algorithm is proposed for stations with exponentially distributed service times, but where the service rate at each station may be different. We also provide some initial thoughts on the impact on the proposed allocation method of including service time distributions with different coefficients of variation.

Keywords: Server Allocation, Zero Buffer, Tandem Queue

1. Introduction

Consider a tandem line consisting of N stations ($N \geq 2$) where the service rate of a server assigned to station i is μ_i ($i = 1, 2, \ldots, N$). The service times at each station follow exponential distributions and are independent and identically distributed with rate μ_i (i.e. the rate can depend on the station). There are M servers available to be allocated to the stations. The servers are capable of working at any station and can process only one job at a time. The servers are homogeneous meaning that servers assigned to the i^{th} station each work at rate μ_i.

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

We assume that there are always jobs waiting to be served at the first station. Jobs served at the last station immediately leave the system. The throughput of the system is equal to the departure rate from the last station. We assume there are no buffers between stations. Our problem of interest is: *given M servers, allocate them to the N stations, such that the throughput is maximized.* We could define a similar problem in terms of blocking and starvation probabilities. In that case, the goal would be to minimize an aggregated measure of these probabilities over all stations.

We propose an algorithm that has as a primary goal to roughly equalize the workloads at each of the stations, meaning that the number of servers is proportional to the mean service time at a station. However, the heterogeneous mean service times and lack of buffers introduce additional complexity beyond making the workloads equal (further discussed in Section 2). We use simulation to obtain insights about the nature of the system and later to measure the performance of our algorithm for a number of configurations. In addition to exponentially distributed service times, we extend the algorithm by considering service times with coefficients of variation other than 1. We illustrate that the algorithm performs well if the coefficients of variation of all stations are increased or decreased equally. Based on a number of simulations, we infer that the algorithm also works well on configurations where the majority of stations have service times with coefficient of variation near one and the remaining stations have service times with coefficient of variation less than one.

The stated problem is motivated by the bed management issue in hospitals. In short, bed management is the problem of assigning a number of beds to different departments of a hospital, such that patient flow is optimized [6]. Patients need to go through these departments to complete their treatment cycle (e.g. the emergency, express, medicine, and alternative level of care departments). The fact that patients must be assigned to a bed at all times, represents the zero-buffer nature of this problem.

A zero-buffer environment arises either from characteristics of the processing technology itself, or from the absence of storage capacity between stations. The bed management problem is caused by the absence of storage capacity. Another example is the allocation of facilities/workers to the stations of an assembly line. As a concrete example, Hu et al. [14] consider a

car assembly line in which each car is carried by a specific conveyor with no extra conveyors between stations.

An example of a case where the technology itself requires a zero-buffer environment is the canning process in which delays should be avoided to keep the food fresh. In particular, no buffer space is allowed between the cooking operation and the canning operation [3]. Another example is the production of steel, where molten steel undergoes a series of operations such as molding into ingots, unmolding, reheating, soaking, and preliminary rolling [21]. To maintain the molten steel's temperature, each operation should follow the previous operation, immediately. Such applications are closely related to the problem of scheduling jobs in a no-wait setting.

Optimization modeling is typically used to formulate general allocation problems in this research domain (see Hillier and So [11], for example). Throughput is denoted by $R(q, s, w)$, where $q = (q_1, q_2, \ldots, q_N)$, $s = (s_1, s_2, \ldots, s_N)$, and $w = (w_1, w_2, \ldots, w_N)$ denote the allocation of buffers, servers, and workload to stations respectively. Q is the total number of available buffer spaces and W is the total mean service time over all stations. The optimization problem is expressed as:

maximize $R(q, s, w)$
subject to

$$\sum_{j=2}^{N} q_j = Q,$$

$$\sum_{j=1}^{N} s_j = M,$$

$$\sum_{j=1}^{N} w_j = W,$$

q_j is an integer greater than or equal to $0, j \in \{2, 3, \ldots, N\}$,
s_j is an integer greater than $0, j \in \{1, 2, \ldots, N\}$,
$w_j > 0, j \in \{1, 2, \ldots, N\}$,

where $q, s,$ and w are decision vectors (q has entries q_j, etc.). Note that workload allocation (w) is the problem of determining the mean service time at

each station, given that the mean service times sum to a fixed value W.

Hillier and So [10] aim to maximize throughput for tandem queues with equal workloads (w_i equal for all i) and small or no buffers ($q_i = 0$ or 1). They claim the optimal server allocation (s) assigns extra servers rather uniformly to the interior stations and refine this claim based on the number of servers and stations at hand. They introduce the *bowl phenomenon*: with single server stations, different mean service times, and equal buffers, the optimal workload allocation (w) assigns less work to the interior stations than to the end stations. It appears that the interior stations (especially the center stations) are critical in determining system performance and so should be given preferential treatment when making design decisions. Alexandros and Papadopoulos [1] extend Hillier and So's method [10] and perform server allocation in large production lines with multiple parallel stations. They employ simulated annealing to solve the optimization problem which models the allocation problem.

Magazine and Stecke [15] consider a three station tandem queueing system with no buffers ($q_i = 0$). They follow the results of Hillier and So [10] and as the number of servers increases, the unbalancing in favour of the middle station is increased. This behavior continues until the unbalancing becomes too severe. At this point, a server is taken away from the middle station and a server is added to the first and third stations. They also state that if unbalancing and distributing servers (w, s) are left to our control, both should be as balanced as possible.

Avi-Itzhak and Yadin [4] study single server stations with no or finite buffers in between stations. For two-station lines, they calculate the mean response time in terms of probabilities of the first station being empty/busy, queue sizes, and the number of jobs in stations.

Cheng and Zhu [5] state that when assigning M heterogeneous servers to M stations with no buffer between the first two ($q_2 = 0$) (resp. the last ($q_M = 0$)) stations and possible buffers for interior stations, it is better to allocate the slower server to the first (resp. the last) station.

Woensel et al. [19, 24] move a step further and consider any possible acyclic multi-server configuration with arbitrary service and inter-arrival time

distributions. They model the joint buffer and server allocation problem (q, s) as a non-linear optimization problem with integer decision variables. They use the Generalized Expansion Method to evaluate throughput. They further use Powell's algorithm (detailed in Himmelblau [13]) for allocation purposes. Smith et al. [20] also model the buffer allocation problem (q) as an optimization problem and use the Generalized Expansion Method to estimate the throughput.

Andriansyah et al. [3] study zero-buffer multi-server general queueing networks. They use the Generalized Expansion Method to evaluate the throughput for a class of acyclic networks. They employ genetic algorithms to solve a multi-objective optimization problem to provide the trade-off between the total number of servers used and the throughput. van Vuuren et al. [23] study multi-server tandem queues with finite buffers with generally distributed service times. They decompose lines to two-station subsystems by a spectral expansion method.

Andradóttir et al. [2] study server allocation (s) in infinite buffer settings $(q_i = \infty)$ with flexible servers using a linear programming approach. We would like to contrast the two extremes (in terms of buffer sizes) in tandem lines for allocation of fixed servers. Namely, in Section 2 we compare our configuration of interest (zero-buffer) with a configuration with infinite buffers between stations.

There has also been work done on the effect of variability of service times for tandem lines. El-Rayah [7] studies the optimal arrangement of single server tandem lines (s) with no buffer spaces $(q_i = 0)$ and where servers have different coefficients of variation. They discover that assigning servers with higher coefficients of variation to the exterior stations leads to higher throughput. Muth and Alkaff [16] study the effect of independent changes in the mean service time and the service time variance on a tandem line's throughput. They study single-server tandem lines with three stations and no buffers and offer a method to compute the throughput. Papadopoulos et al. [9, 17, 18] examine specific production lines (with feedback or unreliable stations) by generating sparse transition matrices and solving them using the Successive Over Relaxation (SOR) method. They consider single-server tandem lines with finite buffers and Erlang or exponential service times.

Futamura [8] studies the effect of service time variability in systems with and without buffers. Futamura suggests that server allocation should follow the inverted bowl phenomenon except that more servers are assigned to stations with higher coefficient of variation to alleviate the impact of higher variance. Hillier et al. [12] define the *inverted bowl phenomenon*: when the total amount of storage space is a decision variable and workloads are equal (w_i equal for all i), the optimal buffer allocation (q) commonly follows an inverted bowl pattern. In other words, the allocation provides the stations toward the center of the line with more buffer storage space than the other stations.

The problem we consider is different in the following respects. The models in [5, 12, 22, 24] include buffers in their configurations. Avi-ltzhak and Yadin [4] study small single server lines, however it is not clear how to generalize their results to longer multi-server lines. Hillier and So [10] consider tandem queues with small buffers and perform simulations for the case with no buffers. They assume that workload is balanced and the numbers of servers at stations differ by at most two (i.e. there is a limited number of extra servers). In other words, starting from a balanced system, they study how to allocate extra servers. We will apply their allocation method to more generic cases to discover its potential shortcomings. Futamura [8] studies the same tandem queues that Hillier and So [10] consider. The tandem line that Magazine and Stecke [15] targets is limited as all rates are equal and there are only three stations. Andriansyah et al. [3] focus on a system with arrivals, with better results achieved when the arrival rate is somewhat below the maximum possible throughput. Our problem of interest assumes a configuration with no buffers where we only have control over server allocation. There are no restrictions on the number of stations or their service rates and there is an infinite amount of work at the first station.

This paper is structured as follows. In Section 2 we model the server allocation problem and propose an algorithm to perform the allocation, given the service rates. In Section 3 we carry out simulations on a number of configurations and analyze the performance of the proposed algorithm. In Section 4 we explain how to modify the algorithm when the coefficients of variation of some or all of the stations are altered. Section 5 provides concluding remarks and a discussion of future work.

2. Server allocation method

The generic optimization problem (stated in the Introduction) refined to our specific problem becomes:

maximize $R(s)$
subject to
$$\sum_{j=1}^{N} s_j = M,$$
$q_j = 0, j \in \{2, 3, \ldots, N\}$,
$s_j \geq 1, s_j \in \mathbb{N}$ and $j \in \{1, 2, \ldots, N\}$,
$w_j = \frac{1}{\mu_j}, j \in \{1, 2, \ldots, N\}$.

In order to design an allocation algorithm, we identify different parameters affecting throughput. An optimal allocation might want to 1) clear blocking, 2) avoid starvation, and/or 3) reduce blocking probability. These parameters are correlated in the following manner: blocking probability increases when the *total service rate* of a station (which is equal to the throughput of a station if it were the only station) is higher than its subsequent station; a blocking station can cause starvation in downstream stations.

The idea is to balance the total service rate for all stations. In this way, the allocation does not increase the blocking and starvation probabilities of a station. Note that the maximum achievable throughput at a station is dependent on the number of servers assigned to all other stations - upstream stations due to starvation effects and downstream stations due to blocking effects. For that reason we intend to use an allocation algorithm that prevents introducing stations with high blocking probability by balancing total service rates. Algorithm 1 is our proposed allocation method.

First consider the following definitions. Whenever a server is allocated to a station, we say the allocation method has *visited* that station. For a given station, the total number of visits to other stations since the last visit to the given station is called the *visit distance* for that station. The visit distance that an allocation method tries to enforce for each station is called the *visit period* of that station.

In Algorithm 1, p_i is the visit period of station i, equal to the ratio of the sum of the mean service times of all stations (W) to the mean service time of station i. This equality was suggested after extensive simulation results. This is also compliant with our goal of balancing total service rates, as the total service rate of each station is proportional to the number of servers working at that station. The quantity l_i is the visit distance for station i. The algorithm assumes that the entries in w are not equal. The case where the entries in w are equal has previously been considered, for example in [10, 15, 22].

Algorithm 1 Allocation Algorithm (M, N, w)

1: $W = \sum_{i=1}^{N} \frac{1}{\mu_i}$
2: $p_i = W\mu_i$
3: $\forall i, s_i = 1$
4: $\forall i, l_i = 0$
5: **while** $\sum_{i=1}^{N} s_i \leq M$ **do**
6: $t =$ the index of the station where $\forall k, k \neq t \;.\; s_t\mu_t < s_k\mu_k$
7: $l_t = 0$
8: $\forall i, i \neq t \;.\; l_i = l_i + 1$
9: **while** $((\exists j \;.\; l_j \geq \lfloor p_j \rfloor) \wedge (j \in$ "high priority stations")) **do**
10: $s_j = s_j + 1$
11: $l_j = 0$
12: $\forall i, i \neq j \;.\; l_i = l_i + 1$
13: **end while**
14: **if** $\sum_{i=1}^{N} s_i \leq M$ **then**
15: $s_t = s_t + 1$
16: **end if**
17: **end while**

Our original conjecture was that the optimal allocation balances the total service rates ($s_i\mu_i$) of all stations. In Algorithm 1, lines 6 - 8 perform this task. The algorithm starts from a situation where one server is allocated to each station and upon allocation of the next server:

- assigns the next server to the station with the smallest value of $s_i\mu_i$ (total service rate);
- in cases where $\exists i, j.\ s_i\mu_i = s_j\mu_j \wedge i \neq j$, chooses the station with higher service rate. (If service rates are equal, follows [10] by assigning extra servers uniformly to the interior stations.)

Our original conjecture ignored the effect of zero buffers, i.e. we thought in terms of a line with infinite buffers between stations. For our system, we have that for all stations the rate of jobs arriving to a station is equal to the rate of jobs departing from that station. However, we know that the maximum departure rate of a station is less than or equal to its total service rate. Given M servers and N stations we consider the following linear programming problem:

maximize $R(s)$
subject to
$s_i\mu_i \geq R(s),\ \forall i \in \{1, \ldots, N\}$,
$\sum_{i=1}^{N} s_i = M.$

This integer programming problem has the solution $R^* = \max_s\{\min_i\{s_i\mu_i\}\}$. This expression is consistent with the results suggested by Andradóttir et al. [2], who consider a more general network topology and flexible servers.

This expression for the optimal throughput immediately leads to a greedy allocation algorithm that always helps the station with the smallest value of $s_i\mu_i$, the same as our original conjecture. However, simulation results revealed that this allocation method alone could lead to allocations that are far from optimal.

We observed that a method that solely balances $s_i\mu_i$ for all stations, could lead to cases where consecutive servers are assigned to a specific station. In particular, this happens when the service rate of a station is low compared to other stations. In such a case, the method allocates consecutive servers to that station to compensate for the low value of $s_i\mu_i$. Numerical results illustrating this fact are included in Section 3.

We performed simulation studies for a number of configurations to characterize properties of optimal allocations. Observing the behavior of optimal allocations, we found that *each station should be visited with a certain period*. In addition, some configurations include a set of stations which we call "high priority stations" (detailed below). If such a set exists, we might need to change the order of allocation to satisfy the following constraint: *visit distances for "high priority stations" should be less than or equal to their visit periods*. In other words, it might be necessary to prioritize a station in this set by postponing visits to stations not belonging to this set. A result of including high priority stations is that optimal allocations avoid the behavior described in the last paragraph, i.e. servers are not assigned consecutively to stations with lower service rates. Lines 9 - 13 of the algorithm implement this constraint.

While we do not specify "high priority stations" completely, we give guidelines on how to choose them. Consider the following expression:

$$\frac{\sum_{i \in \{\frac{1}{\mu_i} < \alpha\}} \frac{1}{\mu_i}}{W} \leq \beta. \tag{1}$$

The set of all stations with mean service times less than α constitute the "high priority stations", if summation of their mean service times over the summation of mean service times of all stations is less than β. We experimentally identified that the members of the set should be chosen so that β is close to 0.2. Also, α should be less than $\frac{W}{N}$.

For example, consider the configuration with mean service time vector $w = (5, 4, 2, 9, 3, 8, 7, 1, 6)$. $W = 45$ for the given configuration. The "high priority stations" are stations with mean service times 1, 2, 3, and 4. In this example we consider all stations with a mean service time less than 5 a member of the "high priority stations" set. Also, we have that the ratio of the sum of mean services times of the set's members to W is equal to $\frac{1+2+3+4}{45} = \frac{10}{45}$ $= 0.222$. As another example, for a system with mean service time vector $w = (12, 7, 13, 3, 5, 4, 1, 10, 9)$, "high priority stations" are stations with mean service times 1, 3, 4, and 5. $W = 64$ for this configuration. We have that the required ratio is equal to $\frac{1+3+4+5}{64} = \frac{13}{64} = 0.203$. Note that if we let

stations with mean service time less than 10 belong to this set, we would have $\frac{1+3+4+5+7+9}{64} = \frac{29}{64} = 0.453$ which makes the ratio too big and hence we do not consider the second and ninth stations as "high priority stations".

For a station to belong to this set, it is necessary that its mean service time is sufficiently small, in the sense that it is less than a proportion of mean service times of stations with higher mean service times. However, this is not sufficient. The number of stations with lower and higher mean service times should be considered, in the following sense. The sum of mean service times of stations with lower mean service times over the total workload is an important proportion. If this proportion is too big, a set of "high priority stations" does not exist. An example is where all stations have the same mean service times except one station, which has a higher mean service time.

The algorithm is not particularly sensitive to the choice of the set of "high priority stations". If it is not obvious if a station belongs to the set, it does not make much difference if it is counted as a "high priority station" or not. The reason is that counting the station as a member of the set results in a better allocation for some values of M and a worse allocation for some other values of M. For example, with $w = (5, 4, 2, 9, 3, 8, 7, 1, 6)$, it is not clear whether the first station should belong to the set or not. For instance, if this station is considered as a member of the set, the algorithm results in closer to optimal throughput for $M = 58$. If it is not considered a member, the algorithm results in closer to optimal throughput for $M = 69$.

Trying to comprehend the need to use "high priority stations", we notice the effect of having multiple servers at a station. In an infinite buffer setting we only care about the product $s_i\mu_i$ and not the individual terms, i.e. s_i and μ_i. For example, 5 servers each working at rate 2 offer the same throughput as 1 server working at rate 10. However, in a zero-buffer setting, a larger number of servers at a station leads to higher throughputs. For example, 5 servers working at rate 2 perform better than 1 server working at rate 10. The reason is that with no buffers, servers act as buffers when blocking occurs. Therefore more servers provide artificial buffer space that helps reduce blocking.

As we intend to balance $s_i\mu_i$ among all stations, a station with higher mean service time is assigned more servers compared to a station with lower

mean service time. However, as stated above, having more servers improves the throughput of a station. Therefore, in order for a low mean service time station to be able to admit multiple jobs coming from high mean service time stations, more servers should be assigned to the low mean service time station. Using the concept of "high priority stations" leads to allocations that take the multiplicity effect into account by making visit distances not bigger than visit periods for such stations.

To gain a better understanding of the multiplicity effect, we present a number of analytical results to support the belief that assigning multiple slow servers to a station leads to better performance than allocating a single fast server to the station (with equal total service rate). This belief is also consistent with the way we decide the server allocation order between stations with equal total service rate but different numbers of servers.

Note that replacing fast servers with slow servers could lead to situations where a station is working at a slower rate. More specifically, at a station with fast servers replaced by slow servers, when there are less jobs than the servers, the departure rate from the station is reduced compared to the case with fast servers. However, the analysis below shows that the trade-off between the increase in buffer size and the potential reduction in total service rate should always be resolved in favour of gaining extra buffer spaces.

Proposition 1: *In a tandem line with two stations and one server per station, the throughput increases if the server at the first station is replaced by a number of slow servers, preserving the total service rate at the first station.*

Proof: Assume the server at the first station (working at rate μ_1) is replaced by γ servers (each working at rate $\frac{\mu_1}{\gamma}$). Let the server at the second station work at rate μ_2. Let $p_{w^\gamma i}$ represent the probability of being in a state with γ busy servers at the first station and an idle server at the second station. Also, set $\mu = \frac{\mu_1}{\mu_2}$. We have (from the solution of the corresponding birth-death process)

$$p_{w^\gamma i} = \frac{1}{1 + \mu + \mu^2 + \sum_{j=1}^{\gamma-1} \frac{\gamma - j}{\gamma} \mu^{j+2}}.$$

The throughput of the system is equal to $\mu_2(1-p_{w^\gamma i})$. Hence the throughput increases as γ is increased.

□

Proposition 2: *In a tandem line with two stations and one server per station, the throughput increases if the server at the second station is replaced by a number of slow servers, preserving the total service rate at the second station.*

Proof: Assume the server at the second station (working at rate μ_2) is replaced by η servers (each working at rate $\frac{\mu_2}{\eta}$). Let the server at the first station work at rate μ_1. Let p_{bw^η} represent the probability of being in a state with a blocked server at the first station and η busy servers at the second station. Also, set $\mu = \frac{\mu_2}{\mu_1}$. We have (from the solution of the corresponding birth-death process)

$$p_{bw^\eta} = \frac{1}{1 + \mu + \mu^2 + \sum_{j=1}^{\eta-1} \frac{\eta - j}{\eta} \mu^{j+2}}.$$

The throughput of the system is equal to $\mu_1(1-p_{bw^\eta})$. Hence the throughput increases as η is increased.

□

Proposition 3: *In a tandem line with three stations and one server per station, the throughput increases if any of the servers is replaced by two slower servers each working at half of the rate of the original server.*

Proof: We only provide the proof for the case where the server at the third station is replaced by two slower servers. When there is a single server at each station, the states are:

$$\{wii, wwi, wiw, www, wbw, bwi, bww, bbw\},$$

where w, i, and b stand for a working, idling, and blocked server, respectively. The throughput is equal to $T = \mu_3(1 - (p_{wii} + p_{wwi} + p_{bwi}))$. When the third station's server is replaced by two slower servers, the states are:

$$\{wiii, wwii, wiwi, wwwi, wiww, wwww, bwii, bwwi, wbww, bbww, bwww\},$$

where the last two letters of each state correspond to the two servers at the third station. The throughput is equal to $T' = \mu_3(p_{wiww} + p_{wwww} + p_{wbww} + p_{bwww} + p_{bbww}) + \mu_3(p_{wiwi} + p_{wwwi} + p_{bwwi})/2$.

Evaluating the sign of $T' - T$ simplifies to:

$\mu_1^4\mu_2^3\mu_3^3(\mu_1+\mu_2)(\mu_1^2+\mu_1\mu_2+\mu_2^2)^2(\mu_1+\mu_2+2\mu_3)(4\mu_1^6+12\mu_1^5\mu_2+20\mu_1^5\mu_3+12\mu_1^4\mu_2^2+48\mu_1^4\mu_2\mu_3+41\mu_1^4\mu_3^2+4\mu_1^3\mu_2^3+44\mu_1^3\mu_2^2\mu_3+84\mu_1^3\mu_2\mu_3^2+44\mu_1^3\mu_3^3+26\mu_1^2\mu_2^3\mu_3+77\mu_1^2\mu_2^2\mu_3^2+78\mu_1^2\mu_2\mu_3^3+26\mu_1^2\mu_3^4+16\mu_1\mu_2^4\mu_3+52\mu_1\mu_2^3\mu_3^2+65\mu_1\mu_2^2\mu_3^3+37\mu_1\mu_2\mu_3^4+8\mu_1\mu_3^5+4\mu_2^5\mu_3+16\mu_2^4\mu_3^2+25\mu_2^3\mu_3^3+19\mu_2^2\mu_3^4+7\mu_2\mu_3^5+\mu_3^6)$,

which is clearly positive, allowing us to conclude $T' > T$. The cases where the first or second stations are replaced with slower servers are proved similarly. □

We considered several configurations with two stations and more than one server at each station (e.g. $s = (2, 3)$). The multiplicity effect holds for them. However, we found it difficult to show the effect for an arbitrary number of servers at the two stations. When the number of stations is increased (in a single-server setting), validating the multiplicity effect becomes a complex algebraic exercise. However, we believe this effect holds in general under Markovian assumptions. The multiplicity effect does not hold for lines with deterministic service times. Hence, the degree of the effect is determined by the service time variance.

In [10], Hillier and So adapt results for the single server setting to analyze multi-server settings. They state that using s_i servers working at rate μ_i at a station is almost equivalent to employing a single fast server working at rate $s_i\mu_i$ with $s_i - 1$ buffer spaces. The multiplicity effect is consistent with this argument. Having multiple servers introduces buffer spaces which in turn increases throughput by reducing blocking.

3. Simulation results

We have simulated a number of configurations with different numbers of stations and servers. As the numbers of servers and stations increase, the

number of possible allocations grows so dramatically that it becomes impractical to find the optimal allocation by simulation. Given M and N and assuming each station has at least one server, the number of possible combinations is N^{M-N}, which illuminates the exponential nature of the search space. In order to be able to simulate the problem, we need to reduce the size of the search space to a manageable size, i.e. filter the set of combinations that we consider. Each run of the simulation consists of tracking the system until a certain number of departures has occurred and then calculating system throughput.

Assume that the throughput values for all combinations of a configuration with given N, M, and w are known. We now want to simulate the system for $M + 1$ servers. Call the combinations leading to the highest throughput values *top combinations*. We track the top combinations with M servers and record the minimum value that each station takes (s_i^{min}) in this set. The s_i^{min} values for top combinations with $M + 1$ servers are greater than or equal to s_i^{min} for top combinations with M servers. If a station in the configuration with $M + 1$ servers has a lower s_i^{min} value compared to the configuration with M servers, this station will have a high blocking probability as its total service rate would be less than other stations, which in turn will reduce the throughput. This is consistent with Algorithm 1 and reinforced by simulations. Note that we do not claim that the optimal allocation for $M + 1$ servers has s_i values that are greater than or equal to those in the optimal allocation for M servers. Our claim is weaker as it targets a range of s_i values and combinations rather than a specific s_i value.

We used this property to limit our search space when simulating $M + 1$ servers using simulation results for M servers. For example, if a station is assigned 9, 10, or 11 servers within the top combinations with 80 servers, for simulating 81 servers we only consider a range around these numbers (say 7 to 13). In practice, the s_i values in top combinations tend to be equal or differ by at most two servers, which supports our choice of the numbers of servers (s_i) to consider.

In order to determine the order in which servers are allocated during a simulation study, we used the following criterion: whenever all s_i values for a station are greater than or equal to a certain number (x) in the top combinations, we let that station have x servers. We recorded the order in which

stations satisfied the above criterion.

For example, consider a case with $N = 9$ and $w = (12, 7, 13, 3, 5, 4, 1, 10, 9)$, with all values of M between 10 and 98. The simulation results are shown in Table 1. Analyzing this table helped us refine our original conjecture in the algorithm. In the table, the header is w, the first column is s_i, and the table entries represent the order in which servers are allocated to stations. To measure the performance of our algorithm, we applied Algorithm 1 to the above configuration. The allocation order generated by the algorithm is shown in Table 2.

s_i	**12**	**7**	**13**	**3**	**5**	**4**	**1**	**10**	**9**
1	11	14	10	24	17	19	52	12	13
2	16	23	15	42	29	33	94	18	20
3	22	32	21	60	39	48		25	27
4	28	40	26	81	54	63		31	35
5	34	50	30		66	77		37	43
6	38	59	36		76	92		44	47
7	45	69	41		90			49	58
8	51	80	46					57	64
9	55	86	53					65	71
10	62	96	56					70	79
11	68		61					74	85
12	73		67					83	93
13	78		72					89	
14	84		75					97	
15	88		82						
16	95		87						
17			91						
18			98						

Table 1: Empirical allocation order for $w = (12, 7, 13, 3, 5, 4, 1, 10, 9)$

To gain a better understanding of how the algorithm performs, we summarize the results in Table 3. In this table, the optimal allocation together with the relative error of the allocation generated by the algorithm are shown (for $w = (12, 7, 13, 3, 5, 4, 1, 10, 9)$). If the allocation suggested by the algorithm is not optimal, the optimal allocation is also presented below the algorithm's

allocation. The average relative error of the allocation provided by the algorithm for $M = 10$ to 98 servers is 0.80%. For the same configuration, if we delay increasing the value of l_i for "high priority stations" even for at most 3 visits, i.e. skipping lines 9 to 13 in the algorithm, experimental results are greatly changed and the average relative error increases to 2.00%. This is in fact the results of the greedy algorithm suggested for the infinite buffer setting (and our original conjecture).

s_i	**12**	**7**	**13**	**3**	**5**	**4**	**1**	**10**	**9**
1	11	14	10	26	17	21	48	12	13
2	16	23	15	45	29	35	85	18	19
3	22	32	20	63	40	49		24	27
4	28	41	25	82	52	62		31	34
5	33	51	30		64	77		37	42
6	38	60	36		76	92		44	47
7	42	69	39		89			53	56
8	50	78	46					58	65
9	55	88	54					66	70
10	59	97	57					71	80
11	67		61					79	86
12	72		68					84	94
13	75		72					91	
14	83		74					98	
15	90		81						
16	95		87						
17			93						
18			96						

Table 2: Algorithm 1 allocation order for $w = (12, 7, 13, 3, 5, 4, 1, 10, 9)$

We have simulated a number of other configurations to verify the proposed algorithm. We describe three of them here. Assume a nine station line with the following mean service times: $w = (1, 2, 3, 4, 5, 6, 7, 8, 9)$. For $M = 45$ through 70 the average relative error is 0.93%. Consider another nine station configuration with the following mean service times: $w = (5, 4, 2, 9, 8, 3, 8, 7, 1, 6)$. For $M = 45$ through 82 the average relative error is 0.46% which suggests that the algorithm is performing well.

M	***s***	***R(s)***	**Err.**	**M**	***s***	***R(s)***	**Err.**
10	(1, 1, 2, 1, 1, 1, 1, 1, 1)	0.0562		55	(10, 6, 10, 3, 5, 4, 2, 8, 7)	0.6037	0.57%
15	(2, 2, 3, 1, 1, 1, 1, 2, 2)	0.1085			(9, 6, 10, 4, 5, 4, 2, 8, 7)	0.6071	
20	(3, 2, 4, 1, 2, 1, 1, 3, 3)	0.1593	3.1%	60	(11, 7, 11, 3, 5, 4, 2, 9, 8)	0.6594	1.88%
	(3, 2, 4, 1, 2, 2, 1, 3, 2)	0.1644			(10, 7, 11, 4, 5, 4, 2, 9, 8)	0.6721	
25	(4, 3, 5, 1, 2, 2, 1, 4, 3)	0.2181	3.97%	65	(11, 7, 12, 4, 6, 5, 2, 9, 9)	0.7367	0.04%
	(4, 3, 4, 2, 2, 2, 1, 4, 3)	0.2271			(13, 9, 15, 5, 7, 6, 2, 12, 11)	0.9329	
30	(5, 3, 6, 2, 3, 2, 1, 4, 4)	0.2858		70	(12, 8, 13, 4, 6, 5, 2, 10, 10)	0.7995	0.43%
35	(6, 4, 6, 2, 3, 3, 1, 5, 5)	0.3482			(12, 8, 13, 4, 6, 5, 2, 11, 9)	0.8029	
40	(7, 4, 8, 2, 4, 3, 1, 6, 5)	0.4074	0.62%	75	(14, 8, 15, 4, 6, 5, 2, 11, 10)	0.8519	1.72%
	(6, 5, 7, 3, 4, 3, 1, 6, 5)	0.4099			(13, 8, 14, 4, 7, 6, 2, 11, 10)	0.8661	
45	(8, 5, 8, 3, 4, 3, 1, 7, 6)	0.4706	1.01%	80	(14, 9, 15, 4, 7, 6, 2, 12, 11)	0.9286	0.46%
	(7, 5, 8, 3, 4, 3, 2, 7, 6)	0.4754			(13, 9, 15, 5, 7, 6, 2, 12, 11)	0.9329	
50	(9, 5, 9, 3, 4, 4, 2, 7, 7)	0.5358	0.86%	85	(15, 9, 16, 5, 7, 6, 3, 13, 11)	0.9985	
	(8, 6, 9, 3, 4, 4, 2, 8, 6)	0.5404		90	(15, 10, 17, 5, 8, 7, 3, 13, 12)	1.0652	
				95	(17, 10, 18, 5, 8, 7, 3, 14, 13)	1.1298	0.09%
					(16, 11, 18, 5, 8, 7, 3, 14, 13)	1.1309	

Table 3: Algorithm 1 allocation for $w = (12, 7, 13, 3, 5, 4, 1, 10, 9)$

We also simulated a configuration with mean service times: $w = (3, 5, 10, 12, 13, 9, 7, 4, 1)$. For $M = 9$ through 31 the average relative error is 0.003% which is better than the result of the $w = (12, 7, 13, 3, 5, 4, 1, 10, 9)$ configuration. According to Hillier and So [10], assigning extra servers to the interior stations lead to higher throughputs. Also, the stations with higher mean service times have smaller visit periods. The algorithm works better for the former configuration, as the higher mean service time stations are also the interior stations.

To further evaluate the algorithm, we performed a simulation study for a longer tandem line which also included stations with equal mean service times. The configuration had mean service times: $w = (2, 5, 7, 14, 13, 10, 4, 3, 13, 5, 12, 11, 7, 3, 8)$. For $M = 15$ through 50 the average relative error is 0.006%. As explained in Section 2, ties are broken by assigning servers uniformly to the interior stations.

Comparing our allocations with the optimal allocations for $w = (12, 7, 13, 3, 5, 4, 1, 10, 9)$, Algorithm 1 always assigns more servers to stations 1, 3, and 9; it always assigns less servers to stations 4, 5, and 6; station 2 often has the same number of servers except for a few cases where the algorithm assigns

less servers. For $w = (5, 4, 2, 9, 8, 3, 8, 7, 1, 6)$, the algorithm always assigns more servers to stations 1, 4, and 9; it always assigns less servers to stations 3, 5, and 8.

Hillier and So [10] define $n = \lfloor \frac{M}{N} \rfloor$ and $E = M - nN$ for systems with equal workloads. They state that when $N - E = 1$ the optimal allocation assigns extra servers to every station except station 1; when $N - E = 2$ it allocates extra servers to every station except stations 1 and N; when $N - E > 2$ they could not characterize an optimal solution. However, in general terms, they suggest to spread the extra servers rather uniformly over the interior stations. Applying their method to $w = (5, 4, 2, 9, 8, 3, 8, 7, 1, 6)$ when $N - E = 1$, then $M = 53$ and the average relative error is 0.41%. When $N - E = 2$, then $M = 52$ and the average relative error is 0.24%. When $N - E > 2$, then we choose $M = 46$ through 50, resulting in an average relative error of 6.05% (our algorithm results in 1.31% average relative error). Note that their method is silent on allocations in the range of $M = 54$ through 89, as the workloads become equal again at $M = 90$.

Similarly, if we apply their guideline to $w = (12, 7, 13, 3, 5, 4, 1, 10, 9)$ when $N - E = 1$, then $M = 72$ and the average relative error is 0.06%. When $N - E = 2$, then $M = 71$ and the average relative error is 0.07%. When $N - E > 2$ we choose $M = 65$ through 69, the average relative error is 2.07%. Their method is silent on allocations in the range of $M = 73$ through 128. Notice that their guideline performs better on the latter configuration compared to the former configuration, as it is a case where "high priority stations" are also interior stations.

4. Impact of Service Time Variance

A natural extension to the above problem is considering service time distributions which are not exponential. More specifically, we modify the distributions so that the coefficients of variation (cv) are not equal to 1. In simulations, an Erlang distribution is used for coefficient of variation less than one and a Hyper-exponential distribution for coefficient of variation greater than one. In such systems, allocations depend on the position of stations in the tandem line, service rates, and coefficients of variation. We try to study the effect of these parameters separately and together. We let cv_j represent

the coefficient of variation of the service time distribution of station j.

The simulation results below suggest that for the following cases, the algorithm works well: coefficients of variation are approximately one except for a small number of stations which are less than one; coefficients of variation are less than one with equal values; coefficients of variation are greater than one with equal values. For these cases, the adjusted visit periods are inversely proportional to the coefficients of variation. For the remaining cases: coefficients of variation are approximately one except for a small number of stations where they are greater than one; coefficients of variation less than one with different values; coefficients of variation greater than one with different values, we provide some guidelines for how to modify the algorithm, but are unable to provide a complete picture.

In a tandem line including stations with $cv_j = 1$ and $cv_j > 1$, the guideline is to follow Algorithm 1 but expedite visits to stations with $cv_j > 1$. Employing Algorithm 1 alone with no modifications could lead to results that are far from optimal. The amount that visit periods should be advanced depends on the coefficients of variation, the p_j values, and the position of stations. However notice that visit periods are still mainly dependent on service rates. Therefore the change in visit periods is not very large compared to p_j. During our simulations, we have observed that a 100% increase in the coefficient of variation of a station changes its visit period by not more than 20%.

For the $w = (12, 7, 13, 3, 5, 4, 1, 10, 9)$ configuration with $cv_5 = 1.61, cv_8 = 1.31$ and $cv_j = 1$ for the remaining stations, the average relative error of the algorithm is 1.61%. As suggested in the previous paragraph, p_5 and p_8 should be decreased (we do not know the exact amount). However, we only need to perform simulations for a couple of M values to calculate new visit periods. These M values are less than multiples of p_5 and p_8, so that we can verify how much the visit periods need to be reduced. Choosing $M = 13, 14, 19$, we inferred that we should change p_5 from 12.8 to 10.75 and p_8 from 6.4 to 6.25. The adjusted visit periods of stations are calculated by dividing M by the number of servers assigned to the stations. Employing the algorithm with the modified visit periods results in an average relative error of 0.94%. Table 4 presents the optimal allocations for this example for $M = 43$ through 69.

In a tandem line where for all stations $cv_j > 1$ with different values, visits

to stations with higher mean service times are expedited which can result in an increase in visit periods of lower mean service time stations. Consider the $w = (12, 7, 13, 3, 5, 4, 1, 10, 9)$ configuration with $cv_j > 1$ for all stations. More specifically, let $cv = (1.22, 1.76, 1.14, 2.68, 2.12, 2.37, 2.45, 1.38, 1.50)$. Algorithm 1 with the original visit periods results in an average relative error of 5.53%. We performed simulations for a small number of M values ($M = 15, 17, 20$) and determined the following values for visit periods: $(6.11, 8, 5.4, 15.66, 10.8, 13, 35, 7, 7.28)$. With these adjusted visit periods the average relative error becomes 0.67%. Comparing optimal allocations with the allocations computed from Algorithm 1 adjusted with the above visit periods, the optimal allocations assign more servers to stations with higher mean service times.

In a tandem line where for all stations $cv_j > 1$ with equal values, the algorithm yields good results. For example for $w = (3, 2, 1, 2, 1, 1, 1, 2)$ and $cv_j = 2.15$ for all stations, for $M = 9$ through 50 the algorithm leads to an average relative error of 0.93%.

In a tandem line including stations with $cv_j = 1$ and $cv_j < 1$, the algorithm leads to reasonable results. However, the optimal allocation tends to postpone visits to stations with $cv_j < 1$ in comparison with p_j calculated in Algorithm 1. In other words, the optimal policy assigns less servers to stations with lower coefficients of variation. For the $w = (12, 7, 13, 3, 5, 4, 1, 10, 9)$ configuration with $M = 10$ through 69, $cv_5 = cv_8 = 0.5$, and $cv_j = 1$ for the rest, the average relative error of the allocations provided by the algorithm is 0.79%. Table 4 presents the optimal allocations for this example for $M =$ 43 through 69.

In a tandem line where for all stations $cv_j < 1$ with equal values, the algorithm yields good results. The optimal allocation assigns more servers to stations with higher mean service times and to stations at the two ends of the line. For the same configurations with $M = 10$ through 69 and $cv_j = 0.5$ for all stations, Algorithm 1 results in an average relative error of 0.52%.

In a tandem line where for all stations $cv_j < 1$ with different values, less servers are assigned to stations with higher mean service times. For $w = (3, 2, 1, 2, 1, 1, 1, 2)$ and $cv = (0.7, 0.7, 0.25, 0.25, 0.7, 0.25, 0.25, 0.25)$ with $M = 9$ through 50, the algorithm results in an average relative error of 1.95%.

The average relative error for the same configuration with $cv_j = 1$ is 0.58%. This suggests that the visit periods in the algorithm should be adjusted. Simulating for a small number of M values ($M = 19, 25, 26, 27$), we inferred that the adjusted visit periods should be $(4.75, 6.33, 12, 6.16, 12, 12.5, 12, 6.9)$. Employing the algorithm with the adjusted visit periods results in an average relative error of 0.58%.

In order to study the effect of coefficient of variation independent of mean service times, we consider a configuration with 4 stations, each having a mean service time of 1. We choose $cv_j = 0.5, 1, 1.34$ for the third station and let $cv_j = 1$ for the other stations. As expected, the inverted bowl phenomenon is followed, i.e. the second and third stations are prioritized. While visit periods remain the same when the coefficient of variation is changed, their ordering changes. For $cv_j = 1$, the allocation priority swings between the second and third station so that balance is achieved in the long run. For $cv_j \neq 1$, although the allocation priority swings between the second and third station, for $cv_j < 1$ the second station is prioritized and for $cv_j > 1$ the third station is prioritized. This observation is consistent with the above claims.

Futamura [8] states that the optimal server allocation follows the inverted bowl phenomenon but assigns more servers to stations with higher coefficient of variation. As the number of stations is increased, the optimal allocation tends to put servers at stations with higher coefficient of variation over stations in the middle. The systems under consideration have equal workload and all coefficients of variation are 1 except some stations which have higher coefficients of variation. Futamura performs simulations for configurations with balanced workloads and a limited number of extra servers. We observe that a more comprehensive analysis of such systems (i.e. when one considers extra servers beyond the limit that Futamura has considered) would suggest that increasing the coefficient of variation of some stations changes the allocation order but leaves visit periods unchanged. Table 4 illustrates this fact. In Table 4, while the fifth station with $cv_5 > 1$ takes servers 5 and 6 sooner compared to the case $cv_5 = 1$, the visit periods for both cases are the same. This is consistent with our earlier observation that changing the coefficient of variation has an effect on the server assignment, but not to the degree that there is a large difference in the number of servers assigned (over the exponential case).

M	$cv_j = 1$	$cv_5 = cv_8 = 0.5$	$cv_5 = 1.61, cv_8 = 1.31$
43	(7, 5, 8, 3, 4, 3, 2, 6, 5)	(7, 5, 8, 3, 4, 3, 1, 6, 6)	(7, 5, 8, 3, 4, 3, 1, 7, 5)
44	(7, 5, 8, 3, 4, 3, 2, 6, 6)	(7, 5, 8, 3, 4, 3, 2, 6, 6)	(7, 5, 8, 3, 4, 3, 1, 7, 6)
45	(7, 5, 8, 3, 4, 3, 2, 7, 6)	(8, 5, 8, 3, 4, 3, 2, 6, 6)	(7, 5, 8, 3, 5, 3, 1, 7, 6)
46	(8, 5, 8, 3, 4, 3, 2, 7, 6)	(8, 5, 8, 3, 4, 3, 2, 7, 6)	(8, 5, 8, 3, 5, 3, 1, 7, 6)
47	(8, 5, 9, 3, 4, 3, 2, 7, 6)	(8, 5, 9, 3, 4, 3, 2, 7, 6)	(8, 5, 8, 3, 5, 3, 2, 7, 6)
48	(8, 5, 9, 3, 4, 4, 2, 7, 6)	(8, 5, 9, 3, 4, 4, 2, 7, 6)	(8, 5, 9, 3, 5, 3, 2, 7, 6)
49	(8, 6, 9, 3, 4, 4, 2, 7, 6)	(8, 6, 9, 3, 4, 4, 2, 7, 6)	(8, 5, 9, 3, 5, 3, 2, 8, 6)
50	(8, 6, 9, 3, 4, 4, 2, 8, 6)	(8, 6, 9, 3, 4, 4, 2, 7, 7)	(8, 6, 9, 3, 5, 3, 2, 8, 6)
51	(8, 6, 9, 3, 5, 4, 2, 7, 7)	(8, 6, 10, 3, 4, 4, 2, 7, 7)	(8, 6, 9, 3, 5, 4, 2, 8, 6)
52	(8, 6, 10, 3, 4, 4, 2, 8, 7)	(9, 6, 10, 3, 4, 4, 2, 7, 7)	(9, 6, 9, 3, 5, 4, 2, 8, 6)
53	(9, 6, 9, 3, 5, 4, 2, 8, 7)	(9, 6, 10, 3, 4, 4, 2, 8, 7)	(9, 6, 9, 3, 5, 4, 2, 8, 7)
54	(9, 6, 10, 3, 5, 4, 2, 8, 7)	(9, 6, 10, 3, 5, 4, 2, 8, 7)	(9, 6, 10, 3, 5, 4, 2, 8, 7)
55	(9, 6, 10, 4, 5, 4, 2, 8, 7)	(9, 6, 11, 3, 5, 4, 2, 8, 7)	(9, 6, 10, 3, 5, 4, 2, 9, 7)
56	(9, 6, 10, 4, 5, 4, 2, 9, 7)	(9, 6, 11, 3, 5, 4, 2, 9, 7)	(9, 6, 10, 3, 6, 4, 2, 9, 7)
57	(9, 7, 10, 4, 5, 4, 2, 9, 7)	(10, 6, 11, 3, 5, 4, 2, 9, 7)	(9, 6, 11, 3, 6, 4, 2, 9, 7)
58	(10, 6, 11, 4, 5, 4, 2, 8, 8)	(10, 6, 11, 4, 5, 4, 2, 9, 7)	(10, 6, 11, 3, 6, 4, 2, 9, 7)
59	(10, 6, 11, 4, 5, 4, 2, 9, 8)	(10, 6, 11, 4, 5, 4, 2, 9, 8)	(10, 6, 11, 3, 6, 4, 2, 9, 8)
60	(10, 7, 11, 4, 5, 4, 2, 9, 8)	(10, 7, 11, 4, 5, 4, 2, 9, 8)	(10, 6, 11, 4, 6, 4, 2, 9, 8)
61	(10, 7, 11, 4, 5, 5, 2, 9, 8)	(10, 7, 11, 4, 5, 5, 2, 9, 8)	(10, 7, 11, 4, 6, 4, 2, 9, 8)
62	(10, 7, 11, 4, 6, 5, 2, 9, 8)	(10, 7, 12, 4, 5, 5, 2, 9, 8)	(10, 7, 11, 4, 6, 4, 2, 10, 8)
63	(11, 7, 11, 4, 6, 5, 2, 9, 8)	(10, 7, 12, 4, 5, 5, 2, 10, 8)	(10, 7, 12, 4, 6, 4, 2, 10, 8)
64	(11, 7, 12, 4, 5, 5, 2, 10, 8)	(11, 7, 12, 4, 5, 5, 2, 10, 8)	(10, 7, 12, 4, 6, 5, 2, 10, 8)
65	(11, 7, 12, 4, 6, 5, 2, 10, 8)	(11, 7, 12, 4, 5, 5, 2, 10, 9)	(11, 7, 12, 4, 6, 5, 2, 10, 8)
66	(11, 7, 12, 4, 6, 5, 2, 10, 9)	(11, 7, 13, 4, 5, 5, 2, 10, 9)	(11, 7, 12, 4, 6, 5, 2, 10, 9)
67	(11, 8, 12, 4, 6, 5, 2, 10, 9)	(11, 8, 12, 4, 6, 5, 2, 10, 9)	(11, 7, 13, 4, 6, 5, 2, 10, 9)
68	(11, 8, 13, 4, 6, 5, 2, 10, 9)	(11, 8, 13, 4, 6, 5, 2, 10, 9)	(11, 8, 13, 4, 6, 5, 2, 10, 9)
69	(11, 8, 13, 4, 6, 5, 2, 11, 9)	(11, 8, 13, 4, 6, 5, 2, 11, 9)	(11, 8, 13, 4, 6, 5, 2, 11, 9)

Table 4: Comparison of coefficient of variation modifications for $w = (12, 7, 13, 3, 5, 4, 1, 10, 9)$

5. Conclusion

In this paper we studied the allocation of a number of servers to stations in a zero-buffer tandem line. We considered a wide range of configurations, varying the number of stations, number of servers per station, and service time distributions. In contrast to infinite buffer settings where balancing total service rates of stations is the only concern, in zero-buffer tandem lines, under certain conditions, stations with lower mean service times should be prioritized. We introduced the multiplicity effect to explain that servers also play the role of buffers in the absence of real storage between stations. Studying the impact of variability in service times through a small number of configurations, we found that the algorithm performs well for the following combinations of service time distributions: coefficients of variation are approximately one except for a small number of stations which are greater than one; coefficients of variation are less than one with equal values; coefficients of variation are greater than one with equal values. In other cases adjustments are required to the algorithm.

In terms of future work, additional simulations might lead to more concrete statements on how to justify visit periods for arbitrary coefficients of variation across all stations. Simulation variables would be the number of stations, distributions of service times, number of servers, and arrangement of coefficients of variation. In this work we have considered non-collaborative homogeneous servers. A natural extension to this work 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.

[1] D. C. Alexandros and P. T. Chrissoleon. On the server allocation in large reliable production lines with exponential processing times. In *Fifth International Conference on "Analysis of Manufacturing Systems - Production Management"*, Zakynthos Island, Greece, 2005.

[2] S. Andradóttir, H. Ayhan, and D. G. Down. Dynamic server allocation for queueing networks with flexible servers. *Operations Research*, 51(6):952 – 968, 2003.

[3] R. Andriansyah, T. Van Woensel, F. R. B. Cruz, and L. Duczmal. Performance optimization of open zero-buffer multi-server queueing networks. *Computers and Operations Research*, 37(8):1472–1487, 2010.

[4] B. Avi-Itzhak and M. Yadin. A sequence of two servers with no intermediate queue. *Management Science*, 11(5):553–564, 1965.

[5] D. W. Cheng and Y. Zhu. Optimal order of servers in a tandem queue with general blocking. *Queueing Systems*, 14(3):427–437, 1993.

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

[7] T. E. El-Rayah. The effect of inequality of interstage buffer capacities and operation time variability on the efficiency of production line systems. *International Journal of Production Research*, 17(1):77–89, 1979.

[8] K. Futamura. The multiple server effect: Optimal allocation of servers to stations with different service-time distributions in tandem queueing networks. *Annals of Operations Research*, 93(1):71–90, 2000.

[9] C. Heavey, H. T. Papadopoulos, and J. Browne. The throughput rate of multistation unreliable production lines. *European Journal of Operational Research*, 68(1):69–89, 1993.

[10] F. S. Hillier and K. C. So. The assignment of extra servers to stations in tandem queueing systems with small or no buffers. *Performance Evaluation*, 10(3):219–231, 1989.

[11] F. S. Hillier and K. C. So. On the optimal design of tandem queueing systems with finite buffers. *Queueing Systems*, 21(3-4):245–266, 1995.

[12] F. S. Hillier, K. C. So, and R. W. Boling. Notes: Toward characterizing the optimal allocation of storage space in production line systems with variable processing times. *Management Science*, 39(1):126–133, 1993.

[13] D. M. Himmelblau. *Applied Nonlinear Programming*. McGraw-Hill Book Company, 1972.

[14] S. Hu, J. Ko, L. Weyand, H. ElMaraghy, T. Lien, Y. Koren, H. Bley, G. Chryssolouris, N. Nasr, and M. Shpitalni. Assembly system design and operations for product variety. *CIRP Annals - Manufacturing Technology*, 60(2):715 – 733, 2011.

[15] M. J. Magazine and K. E. Stecke. Throughput for production lines with serial work stations and parallel services facilities. *Performance Evaluation*, 25(3):211–232, 1996.

[16] E. J. Muth and A. Alkaff. The bowl phenomenon revisited. *International Journal of Production Research*, 25(2):161–173, 1987.

[17] H. Papadopoulos, C. Heavey, and M. O'Kelly. Throughput rate of multistation reliable production lines with inter station buffers (I) exponential case. *Computers in Industry*, 13(4):229 – 244, 1989.

[18] H. Papadopoulos, C. Heavey, and M. O'Kelly. Throughput rate of multistation reliable production lines with inter station buffers (II) erlang case. *Computers in Industry*, 13(4):317 – 335, 1990.

[19] J. M. Smith, F. R. B. Cruz, and T. Van Woensel. Optimal server allocation in general, finite, multi-server queueing networks. *Applied Stochastic Models in Business and Industry*, 26(6):705–736, 2010.

[20] J. M. Smith, F. R. B. Cruz, and T. Van Woensel. Topological network design of general, finite, multi-server queueing networks. *European Journal of Operational Research*, 201:427–441, 2010.

[21] W. F. Smith and J. Hashemi. *Foundations of Materials Science and Engineering*. McGraw-Hill, 2006.

[22] D. Spinellis, C. Papadopoulos, and J. MacGregor Smith. Large production line optimization using simulated annealing. *International Journal of Production Research*, 38(3):509–541, 2000.

[23] M. van Vuuren, I. J. Adan, and S. A. Resing-Sassen. Performance analysis of multi-server tandem queues with finite buffers and blocking. *OR Spectrum*, 27(2/3):315–338, 2005.

[24] T. Van Woensel, R. Andriansyah, F. R. B. Cruz, J. M. Smith, and L. Kerbache. Buffer and server allocation in general multi-server queueing networks. *International Transactions in Operational Research*, 17(2):257–286, 2010.