Allocating Outsourced Warranty Service Contracts

Motivated by our interactions with a leading manufacturer of computers, in this paper we consider static allocation as applied to the problem of minimizing the costs of outsourcing warranty services to repair vendors. Under static allocation, a manufacturer assigns each item to one of several contracted repair vendors; every time a particular item fails, it is sent to its preassigned vendor for repair. In our model, the manufacturer incurs a repair cost each time an item needs repair and also incurs a goodwill cost while items are undergoing repair. We model each service vendor as a finite population multi-server queueing system and formulate the resulting outsourcing problem as an integer-variable resource allocation problem. After establishing convexity results regarding the queue lengths at the repair vendors, we show that marginal allocation is optimal. Through a detailed computational study we compare the optimal algorithm with five static allocation heuristics in terms of time and optimality gap. Our study indicates that the optimal algorithm takes less than a minute to solve industry size problems on average. Further, the commonly used heuristics are far away from the optimal on average, thus emphasizing the benefits of the optimal allocation algorithm. We also compare the optimal static allocation to two simple dynamic allocation heuristics. The results of this study further validate the use of static allocation as a justifiable and easy-to-implement policy. Among other computational insights we show that when the number of items to be allocated is large, a single-server approximation leads to optimal allocations in most of the cases.

1 Introduction

The last decade has seen an explosion in the degree of outsourcing of various business operations that were traditionally performed in-house. Specifically, warranty services, which is a major component of the manufacturing and retail industry, has experienced a rising trend in terms of outsourcing. Several large companies in the past few years have begun to outsource their repair activities. For example, electronic equipment manufacturers routinely augment in-house servicing of warranties by contracting outside vendors to repair the items (machines like personal computers, or components like hard drives) that are sold under warranty. According to a recent report by Merrill Lynch, the warranty services represents a 100 billion dollar opportunity for electronics manufacturers and subcontractors (Serant [18]). Several vendors in the recent past have made large investments to enhance their capabilities in this dimension. For example, Solectron has acquired seven repair companies for mobile phones, desktop computers and notebooks since 1999 (Serant [17]).

Outsourcing repair and warranty services provides an opportunity for the original equipment manufacturer (OEM) to improve turnaround times and have better asset utilization by using the core competencies of the vendors specializing in repairs. However, it also poses downward risks in terms of customer satisfaction, particularly in cases where poor experience in repair services may translate into future lost sales. Thus, it is very important for the OEM to not only pay attention to cost reductions but also customer experience while managing the outsourced warranty and repair services.

Our motivation for this research comes from extensive discussions with the warranty and repair services division of a prominent local computer manufacturer, although these issues are encountered by other manufacturers in the electronic industry as well. A typical large manufacturer of personal computers (PC) sells them with a warranty that covers parts and labor for the maintenance of the PC for a stipulated period of time. The charges for repairs are spelled out in the contract. Generally, the customer pays no money if the computer is under warranty, and the manufacturer absorbs the entire cost. While there are many important issues in the outsourcing of warranty repair services to subcontracted vendors, we focus mainly on the problem of allocation. That is, when a manufacturer has contracts with several repair vendors, it must decide which vendor will be used to repair each failed item.

In particular, we are interested in the problem of minimizing the costs of outsourcing warranty services to alternative repair vendors using static allocation. Under static allocation, each item is assigned to one of the repair vendors; every time a particular item fails, it is sent to its preassigned vendor for repair. In our model, the cost structure is such that the manufacturer incurs a repair cost each time an item needs repair and also incurs a goodwill cost for the delay experienced by a customer while items are awaiting or undergoing repair. Using this cost structure, our model is one of partitioning the customer base into several groups, where each group has its own multi-server queue for service. Therefore, although we have chosen to focus on static allocation of warranty repair services, the model could easily be adapted to many situations in which a finite population of customers is serviced by one of several possible queues. The purpose of this study is to gain insights into the performance of the actual system; the algorithms developed were not designed with the sole intent of being a decision-support model.

One may expect that a dynamic model—that is, an item under warranty is not assigned to a particular vendor until the time of failure, and the current workload of each vendor is taken into account when deciding the assignment—would perform better than a static

model. However, for problems of practical size, computing the optimal dynamic assigments is intractable. In addition, the static model has significant practical advantages in terms of its simplicity and ease of implementation. For example, the manufacturer does not need to keep track of the current state at each vendor, and does not even need to maintain a centralized call center for handling warranty complaints. Rather, it can delegate this function to the individual repair vendors. We will show through a numerical study that the optimal static allocation of items to outsourced vendors is indeed a justifiable policy when compared to other static allocation heuristics, and also when compared to rather simple (non-optimal) dynamic allocation heuristics, such as join-the-shortest-queue or myopic dynamic allocation.

Furthermore, there may be situations which necessitate the use of static allocation over dynamic allocation, such as when the manufacturer does not have complete information about the current state of each vendor. For example, the manufacturer would certainly know how many items it has routed to each vendor, but if a vendor does not immediately report the repair completions to the manufacturer, the state information that the vendor possesses would be incorrect.

Although the application of static allocation models to warranty outsourcing is new, static allocation models are relatively common in areas such as load balancing (Combé and Boxma [5], Hordijk, Loeve, and Tiggelman [12], Cheng and Muntz [4]), server allocation (Rolfe [16], Dyer and Proll [7], Shanthikumar and Yao [19], [20]), and portfolio selection (Zipkin [22]). Using performance measures such as the mean waiting time of a customer or the total number of customers in the system, Combé and Boxma [5] consider two different types of static allocation — probabilistic allocation and pattern allocation — for an open network with single-server queues. Under probabilistic allocation, a customer is routed to queue i with probability p_i, independent of the number of customers in each of the queues. Pattern allocation uses an infinite string of integers (with possibly repeating subsequences) $\{a_1, \ldots, a_n, \ldots\}$, where a_n denotes the number of the queue to which the nth customer in the arrival process is routed.

Several authors have considered the problem of allocating servers, rather than customers, in a queueing network. Rolfe [16] considers the case of Poisson arrivals (hence, an open queueing network), where the objective is to minimize the expected queueing time of customers in the system. He shows that marginal allocation is optimal in the case of constant service times, and suggests that it is optimal for exponential service times; Dyer and Proll [7] then prove this conjecture. Shanthikumar and Yao [19], [20] consider optimal allocation of servers

in a closed network in order to maximize the throughput of the system.

Heinhold [10] develops a model to explain the allocation of customers to automobile inspection stations, where the customers are free to choose any inspection station. Heinhold suggests that this approach can be used for similar problems: for example, the allocation of customers to supermarkets. The key difference in this model is that the customer chooses to which station he is routed.

Our model is most similar to Hordijk, Loeve, and Tiggelman [13], who analyze a closed queueing network in which customers must be assigned to one of two parallel single-server queues, and the routing decision may not depend on the numbers of customers in the queues. However, our model allows an arbitrary number of queues, each of which may have multiple servers. In addition, the particular cost structure of the warranty outsourcing problem includes a fixed cost of routing a customer to a given queue, as well as a holding cost while the customer is awaiting or undergoing repair. We have encountered no other static allocation models incorporating this cost structure.

To compute the optimal routing policy, Hordijk, Loeve, and Tiggelman [13] use an algorithm based on successive approximation, which, because of the complex state space, can only be used for a system with a small number of customers (see also Hordijk and Loeve [11]). In Hordijk and Loeve [12], closed queueing networks of quasi-reversible queues are considered, and techniques of linear programming are used to find an optimal deterministic routing of the customers, based on any cost function that depends on the state of the queueing network. In this paper, we show that marginal allocation (similar to Rolfe [16] and Dyer and Proll [7] for server allocation) can be used to compute the optimal static allocation of customers to the queues.

The rest of the paper is organized as follows. In Section 2 we present our model and formulate it as a resource allocation problem. In Section 3 we describe our solution method when (i) the objective function is convex; (ii) the objective function is concave; and (iii) the objective function may be neither convex nor concave. In Section 4 we present results from a detailed computational study that compares the optimal algorithm with five static allocation heuristics as well as two simple dynamic allocation heuristics. Section 5 contains concluding remarks and general managerial implications.

2 Model Formulation

We model a manufacturer that needs to allocate K identical items (computers in our case) under warranty among V vendors; one of the vendors could be the in-house repair facility of the manufacturer. This closed population assumption is realistic in cases where the manufacturer makes batch sales; for example, the sale of 10,000 computers to a large university. More importantly, however, the closed population model can be used to approximate a system that has, in steady state, a fixed number of identical items covered under warranty.

The manufacturer knows the number of servers (s_i) and their average rate of repair (μ_i) at each of the vendors. We assume that all servers at a single vendor are identical and that the service times are exponential. We also assume that the time between failures for a single item is exponentially distributed with rate λ. The exponential assumption for time between failure and for repair is common in the reliability literature (Blischke and Murthy [2]).

Note, however, that the exponential assumption for failure times is not required. For the machine interference model, Bunday and Scraton [3] show that the expected number of machines not running in steady state depends on the failure time distribution only through its mean. Another way to see this result is to think of this particular instance of the machine interference model as a closed network of two stations, one with an infinite number of servers and general service times, and the other with s servers and exponential service times. The service time in the infinite-server station models the time until failure for each customer (machine). It is a well-known result in the theory of queueing networks that the limiting distribution of the state of the network depends on the service time distribution in the infinite-server queue only through its mean (Baskett et al. [1]). Therefore, since all we will need in the cost calculation for our model is the expected number of machines not running (that is, the expected number of machines at the s-server station), the results depend only on the mean time to failure and not on the distribution of the time to failure.

The contract with vendor i ($i = 1, \ldots, V$) specifies that the manufacturer will pay the vendor a fixed amount c_i for each repair performed by the vendor. Such "fixed fee per repair" contracts, independent of the type of repair and the cost to the vendor, are common when all the materials and parts are charged to the manufacturer and the repair vendor only charges for the labor costs, or for items where the materials and parts costs are minimal compared to the labor costs of repair. In addition, we assume that the manufacturer incurs a goodwill cost at a rate of h_i per unit time that an item spends in service at vendor i. This is usually

the same for all vendors (i.e., $h_i = h$ for $i = 1, \ldots, V$) since it reflects the loss of goodwill for the manufacturer directly from the customer, independent of the repair vendor used. This goodwill cost prevents the manufacturer from overloading an inexpensive vendor with poor service, which would result in unacceptably large turnaround times for warranty service. Note that by assigning the same goodwill delay cost to all customers, we are assuming that all customers are identical.

Under the above assumptions, the manufacturer must decide x_i, the number of items to allocate to vendor i, to minimize the expected total warranty cost. There are two types of allocations that are commonly adopted by manufacturers. In a static allocation, the manufacturer projects total volume of sales of a particular product and then allocates all the items under warranty to the different vendors at the beginning; this allocation then remains in effect for the entire contract period. All repairs for a customer are handled by the assigned repair vendor. In a dynamic allocation, as each item experiences a failure, it is dynamically assigned to a vendor based on the current congestions. We focus our attention on the static allocation in this paper, for reasons described in Section 1. However, in Section 4, we include a computational comparison of the optimal static allocation with simple dynamic allocation heuristics.

Before we continue with the formulation of the model, we shall comment briefly on our model assumptions and the applicability of static allocation for outsourcing warranty repair contracts. We have stated that the manufacturer knows the service rate (μ_i) and number of servers (s_i) at each repair vendor. In reality, capacity for a repair vendor is often shared among several manufacturers. Therefore, we would also have to model a delay at the repair vendor to account for external customers. However, modeling such a delay could become very complicated and would require even more unrealistic assumptions—if it is unlikely that a manufacturer knows the service parameters of a repair vendor, then it is even more unlikely that the manufacturer knows the external rate of arrivals (failures) from other manufacturers that utilize that repair vendor.

On the other hand, for very large manufacturers, it is not uncommon for repair vendors to dedicate a portion of their repair facility exclusively to the large manufacturer. We will assume this is the case; in other words, the manufacturer effectively enforces preemptive priority in its contracts with outsourced repair vendors. Furthermore, we also assume that the manufacturer does know the service parameters μ_i and s_i for each vendor. One can imagine that if a vendor does not perform according to its disclosed service capacity, the

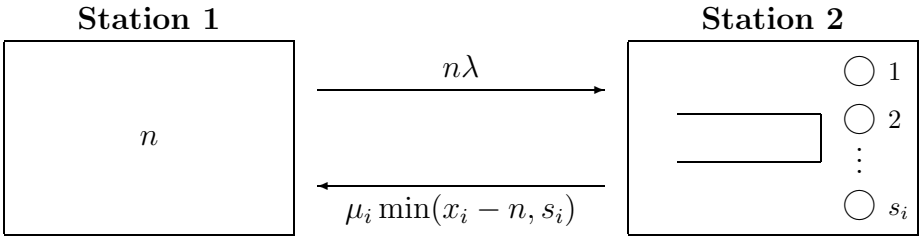

Figure 1: Finite-population queueing system

manufacturer will not continue to do business with that vendor. Therefore, in steady state, the relationship between the manufacturer and vendors is one that encourages full disclosure of information.

Under these assumptions, and given that x_i items have been allocated to vendor i, the repair process can be modelled by an $M/M/s_i/\infty/x_i$ queue with arrival rate λ, service rate μ_i, and finite population x_i. Figure 1 shows the population dynamics in this system. Station 1 includes all items that are properly functioning (that is, not undergoing repair); in this figure, there are n such items. These items each fail with rate λ, so items move from station 1 to station 2 with rate $n\lambda$. Station 2 is the repair queue, and consists of s_i servers each having repair rate μ_i. Therefore, items move from station 2 to station 1 with rate $\mu_i \min(x_i - n, s_i)$, where $x_i - n$ is the number of items at station 2 (because there are n items at station 1 and a finite population of x_i items).

Let $L_i(x_i)$ be the expected number of items at vendor i when x_i items are allocated to vendor i (that is, $L_i(x_i)$ is the expected number of items at station 2 in Figure 1). Directly computing $L_i(x_i)$ from the probability distribution is time-consuming and inefficient. However, the values of $L_i(x_i)$ can be computed recursively using mean value analysis, as described in Section 2.1. Among the x_i items that are allocated to vendor i, the expected number of properly functioning items is $x_i - L_i(x_i)$, and each such item has failure rate λ. Therefore, the expected number of arrivals to the ith vendor per unit time is $\Lambda_i(x_i) = \lambda(x_i - L_i(x_i))$.

The manufacturer's expected cost per unit time for repairs at the ith vendor includes a per unit cost c_i every time there is an arrival to the ith vendor, plus a goodwill cost h_i for each item that is waiting at vendor i, and is given as follows:

$$f_i(x_i) = c_i\Lambda_i(x_i) + h_iL_i(x_i) = \lambda c_i x_i + (h_i - \lambda c_i)L_i(x_i) \tag{1}$$

2.1 Recursive Calculation of $L(x)$

In this section, we describe how to efficiently compute $L_i(x_i)$, which is needed in the cost function $f_i(x_i)$ given in (1). Mean value analysis is commonly used to compute statistics such as mean queue sizes, mean waiting times, and throughput in a closed queueing network; see, for example, Reiser and Lavenberg [15]. The principal advantage of mean value analysis is that the product terms and normalization constants do not need to be computed, thereby avoiding problems with numerical stability. For the sake of completeness, in this section we describe the approach of mean value analysis as applied to the static allocation problem.

To simplify notation, we drop the subscript i. First consider the $M/M/s/s$ loss model with arrival rate λ and service rate μ. The blocking probability, $B(r, s)$, is given by

$$B(r,s) = \frac{r^s/s!}{\sum_{i=0}^{s} r^i/i!}, \qquad s > 0, \tag{2}$$

where $r = \lambda/\mu$. The blocking probability can be computed using the following recursion, starting with $B(r, 0) = 1$:

$$B(r,s) = \frac{B(r,s-1)r/s}{1 + B(r,s-1)r/s}, \qquad s > 0. \tag{3}$$

Now consider the $M/M/s$ model with arrival rate λ and service rate μ. The probability of waiting, $P_W(r, s)$, is given by

$$P_W(r,s) = \frac{r^s/s!}{(1-r/s)\sum_{i=0}^{s-1} r^i/i! + r^s/s!}, \qquad s > 0. \tag{4}$$

It follows from (2) and (4) that

$$P_W(r,s) = \frac{B(r,s-1)r/s}{1 - r/s + B(r,s-1)r/s}, \qquad s > 0. \tag{5}$$

Next consider a closed two-station network similar to that shown in Figure 1; station 1 is an ample-server station with service rate λ and station 2 is a multi-server station with s servers and service rate μ. Customers move from station 1 to station 2 and from station 2 to station 1 indefinitely. The number of circulating customers is x. The probability that all s servers in station 2 are busy is given by

$$P_B(\rho,x) = \begin{cases} 0, & x < s, \\ \dfrac{\sum_{i=0}^{x-s} \rho^i/i!}{\sum_{i=0}^{x-s} \rho^i/i! + A(\rho,x)\rho^{x-s}/(x-s)!}, & x \geq s, \end{cases} \tag{6}$$

where $\rho = s\mu/\lambda$ and

$$A(\rho, x) = \sum_{i=1}^{s} \frac{\rho^i s(s-1)\cdots(s-i+1)}{(x-s+1)\cdots(x-s+i)s^i}, \qquad x \geq s.$$

Note that the number of terms in $A(\rho, x)$ is s, independent of x. From (6) we obtain, for $x \geq s$,

$$P_B(\rho, x) = \frac{1}{1 + A(\rho, x)B(s\mu/\lambda, x-s)},$$

where $B(s\mu/\lambda, x-s)$ denotes the blocking probability in the loss model with $x-s$ servers.

To compute the mean number of customers in station 2, we use the mean value approach. Let $L_q(\rho, x)$ be the mean number of customers waiting in the queue at station 2 when there are x circulating customers in the network, let $W_q(\rho, x)$ be the mean waiting time in the queue at station 2, and let $\Lambda(\rho, x)$ be the throughput of station 2. Then the following relations hold (which are based on the arrival theorem and Little's law):

$$\begin{aligned} W_q(\rho, x) &= \frac{P_B(\rho, x-1) + L_q(\rho, x-1)}{s\mu} \\ \Lambda(\rho, x) &= \frac{x}{1/\lambda + W_q(\rho, x) + 1/\mu} \\ L_q(\rho, x) &= \Lambda(\rho, x) W_q(\rho, x) \end{aligned}$$

These relations are recursive in the population size x, starting from $L_q(\rho, 0) = P_B(\rho, 0) = 0$. Then $L(\rho, x)$ can be computed as $L(\rho, x) = L_q(\rho, x) + \Lambda(\rho, x)/\mu$.

When $s = 1$, $L(\mu/\lambda, x)$ can be simplified as follows:

$$L(\mu/\lambda, x) = x - \frac{\mu}{\lambda}\left(1 - B(\mu/\lambda, x)\right),$$

where $B(\mu/\lambda, x)$ can be computed recursively using (3).

2.2 Optimization Problem

To find the optimal allocation of the K items among the V vendors, the manufacturer solves a resource allocation problem with integer variables (see Gross [9], Fox [8], Ibaraki and Katoh [14]).

$$
\begin{aligned}
\min \quad & \sum_{i=1}^{V} f_i(x_i) \\
\text{s.t.} \quad & \sum_{i=1}^{V} x_i = K \\
& x_i \geq 0 \text{ and integer}, \quad i = 1, \ldots, V
\end{aligned}
$$

To use existing methods for solving this problem, we require convexity properties of the cost functions $f_i(x_i) = \lambda c_i x_i + (h_i - \lambda c_i) L_i(x_i)$. The convexity is established using the following Corollary; we drop the subscript i for ease of notation.

Corollary 1. *$L(x)$ is convex with respect to x.*

We first give the intuition behind the proof before stating the formal proof below. We can think of the system at a single vendor as a closed two-station network with x customers, similar to that shown in Figure 1. Station 1 is an ample-server station with service rate λ. Station 2 consists of s servers, each with service rate μ. There are x jobs circulating between the two stations.

Let $L_A(x)$ denote the mean number of jobs in station 1. To show that $L(x) = x - L_A(x)$ is convex in x, we can show that $L_A(x)$ is concave in x, i.e.,

$$L_A(x) - L_A(x-1) \geq L_A(x+1) - L_A(x). \tag{7}$$

Suppose that $x - 1$ jobs are red and one is blue. In station 2, the red jobs are serviced with preemptive priority over the blue job. Let $\rho(x)$ denote the fraction of time that the blue job spends in station 1. Because the blue job does not exist for the red jobs, it follows that the mean number of red jobs in station 1 is $L_A(x-1)$, and hence

$$\rho(x) = L_A(x) - L_A(x-1).$$

Thus to establish (7) it suffices to show that the utilization rate $\rho(x)$ is nonincreasing in x. To do so, we compare the systems with x and $x+1$ circulating jobs, each system having exactly one low priority blue job. When an additional red job is added to the system, it increases the blue job's mean waiting time in station 2, due to the preemptive priority of the red jobs. The blue job spends a larger fraction of time in station 2 and a smaller fraction of time in station 1; therefore, $\rho(x)$ is decreasing in x.

The formal proof of Corollary 1 follows directly from the proof of throughput concavity by Dowdy et al. [6]; see also Shanthikumar and Yao [21].

Proof. Consider the closed two-station network with x customers described above. Let T_2 be the long-run average throughput at station 2. The long-run average throughput at station 1 is given by $T_1 = \lambda(x - L(x))$. In a closed network, we must have $T_1 = T_2$; i.e., $T_2 = \lambda(x - L(x))$, or $L(x) = x - T_2/\lambda$. Dowdy et al. [6] prove that T_2 is a concave function of x; hence, it follows that $L(x)$ is a convex function of x. □

Therefore, from Corollary 1, $f_i(x_i)$ is convex if $h_i \geq \lambda c_i$, and concave if $h_i \leq \lambda c_i$.

3 Solution Method

In practice, it is reasonable to assume that $h_i \geq \lambda c_i$, because if $h_i < \lambda c_i$, there is no incentive to repair the item; the cost of holding it at the vendor is smaller than the expected cost to repair it as it fails over time. Therefore, when $h_i < \lambda c_i$, the manufacturer's motivation is not to repair items quickly, but rather to repair items slowly (or not at all) so that they remain in a failed state as long as possible. In doing so, the manufacturer would delay the time until the item fails again, thereby reducing the frequency with which the fixed cost for failure must be paid.

However, even though the case $h_i < \lambda c_i$ is extremely rare in practice and can only arise under unusual circumstances, we are interested in a complete solution to the problem under any possible relationship between h_i and λc_i. Therefore, in this section, we present the optimal algorithm for the case when (i) all $f_i(x_i)$ are convex; (ii) all $f_i(x_i)$ are concave; and (iii) some $f_i(x_i)$ are convex and other $f_j(x_j)$ are concave.

3.1 Convex Case

When $h_i \geq \lambda c_i$ for all $i = 1 \ldots, V$, then all $f_i(x_i)$ are convex. This problem is the separable convex resource allocation problem. The optimal allocation in this case can be found using marginal allocation, first proposed by Gross [9]:

- **Step 0:** Set $x_i = 0$, $i = 1, \ldots, V$
- **Step 1:** Choose a $j \in \underset{i=1,\ldots,V}{\operatorname{argmin}}\{f_i(x_i + 1) - f_i(x_i)\}$
- **Step 2:** Set $x_j = x_j + 1$
- **Step 3:** If $\sum_{i=1}^{V} x_i < K$ go to Step 1; else, stop.

An important implication of the optimality of marginal allocation is that a new customer who enters the system can be allocated without recalculating the optimal allocation for the K original customers. That is, when a new customer is added to the system, we continue to assume a closed queueing network, but with $K+1$ customers instead of K customers. If we have already found the optimal allocation of K customers, we need to perform only one more iteration of the marginal allocation algorithm to determine the allocation of the new customer. Note, however, that this does not imply that marginal allocation works for all dynamically changing population sizes. For example, if a customer leaves the system, the optimal allocation might be to reassign customers to different vendors, but reassignment is not allowed in our static allocation model.

3.2 Concave Case

When $h_i \leq c_i\lambda$ for all $i = 1, \ldots, V$, then all $f_i(x_i)$ are concave. The objective function is concave, and the optimum occurs at an extreme point. Therefore, the solution method is trivial: choose a vendor j for which $f_j(K)$ is minimum, and allocate all K items to that vendor.

3.3 Mixed Case

When at least one vendor i has $h_i \geq c_i\lambda$, and at least one vendor j has $h_j < c_j\lambda$, the objective function is neither convex nor concave. Let $A^+ = \{i : h_i \geq c_i\lambda\}$ and let $A^- = \{i_i : h < c_i\lambda\}$; that is, A^+ is the set of vendors for which the corresponding objective term is convex, and A^- is the set of vendors for which the corresponding objective term is concave. For a fixed value $k \in \{0, \ldots, K\}$, consider the following problem.

$$\begin{aligned} z(k) = \min \quad & \sum_{i \in A^+} f_i(x_i) + \sum_{i \in A^-} f_i(x_i) \\ \text{s.t.} \quad & \sum_{i \in A^+} x_i = k \\ & \sum_{i \in A^-} x_i = K - k \\ & x_i \geq 0 \text{ and integer}, \quad i = 1, \ldots, V \end{aligned} \tag{8}$$

Problem (8) gives us the optimal solution for a fixed k; we must then optimize over $k = 0, \ldots, K$ to obtain the optimal solution to the original problem,

$$z^* = \min_{k=0,\ldots,K} z(k). \tag{9}$$

Note, however, that solving (8) is equivalent to separately solving problems (10) and (11), below:

$$\begin{aligned} z^+(k) = \min \quad & \sum_{i \in A^+} f_i(x_i) \\ \text{s.t.} \quad & \sum_{i \in A^+} x_i = k \\ & x_i \geq 0 \text{ and integer}, \quad i \in A^+ \end{aligned} \tag{10}$$

$$\begin{aligned} z^-(k) = \min \quad & \sum_{i \in A^-} f_i(x_i) \\ \text{s.t.} \quad & \sum_{i \in A^-} x_i = K - k \\ & x_i \geq 0 \text{ and integer}, \quad i \in A^- \end{aligned} \tag{11}$$

Therefore, for a fixed value k, problem (10) can be solved using marginal allocation as in Section 3.1. Furthermore, the computation of the optimal allocation for $k = K$ requires the computation of the optimal allocation for all $k < K$, so finding the optimal solutions for all $k = 0, \ldots, K$ is no more difficult than finding an optimal solution for $k = K$ (although it does require more storage).

Similarly, for a fixed value $K - k$, problem (11) can be easily solved: choose a vendor $j \in A^-$ for which $f_j(K - k)$ is minimum, and allocate all $K - k$ items to that vendor.

4 Computational Study

In this section, we present results from our computational study. Our main aim in this numerical study was (i) to test the ability of the optimal algorithm to handle industry size problems; (ii) to compare the performance of the optimal algorithm with commonly used heuristics; (iii) to develop insights on the effect of service rates of vendors on the optimal allocation; and (iv) to test the effect of utilizing a single-server approximation of vendors on the optimal allocation.

Based on our interaction with our industrial contact we constructed a data set that adequately represented the complexity in the real environment. In particular, our parameter choices are as follows. We consider examples with four vendors ($V = 4$), a fixed failure rate of $\lambda = 1.2$ failures per year, and a fixed holding cost of $h = \$1000$ per year. For each value of $K \in \{100, 1000, 10000, 100000\}$ and for each vendor $i = 1, \ldots, 4$, we choose a low value

of μ_i, denoted $\underline{\mu}_i$, and a high value of μ_i, denoted $\overline{\mu}_i$. Similarly, we choose low and high values of c_i and s_i, denoted $\underline{c}_i$, $\overline{c}_i$, $\underline{s}_i$, and $\overline{s}_i$, respectively. We then create a single trial for each combination of μ_i, s_i, and c_i such that $\mu_i \in \{\underline{\mu}_i, \overline{\mu}_i\}$, $c_i \in \{\underline{c}_i, \overline{c}_i\}$, and $s_i \in \{\underline{s}_i, \overline{s}_i\}$ $(i = 1, \ldots, 4)$. This gives us $2^{12} = 4096$ trials for each value of K, for a total of 16,384 trials. We then repeated this process five times with different low and high values for each parameter, for a total of 81,920 trials. The data is given in Table 8 through Table 11 in the Appendix.

4.1 Running Time of Optimal Algorithm

Table 1 gives the average running time, in seconds, for computing the optimal allocation for each value of K. Note that the running time is approximately linear in K. Our results indicate that the optimal algorithm is very fast and can handle industry size problems within a minute in most cases.

Table 1: Mean time to compute the optimal allocation

K	Time (in seconds)
100	0.0233
1,000	0.2071
10,000	2.1349
100,000	24.5890

4.2 Heuristics

In this section we compare the optimal static allocation to several common allocation heuristics. Section 4.2.1 first focuses on optimal static allocation as compared to other (non-optimal) static allocation heuristics. The running times for these heuristics is negligible; therefore, we investigate the optimality gap when using these very fast heuristics.

Section 4.2.2 then compares optimal static allocation to two common dynamic allocation heuristics. The cost of using either of the dynamic allocation heuristics is computed using simulation. Therefore, rather than using the entire set of data described in Section 4, we make comparisons based on 50 examples with $K = 100$ and 50 examples with $K = 1{,}000$.

4.2.1 Static Heuristics

We chose five common static allocation heuristics for comparison with the optimal static allocation algorithm.

- **H1**: Equal allocation to all vendors. Note that if K/V is fractional, the allocation is $\lfloor K/V \rfloor$ to each of the first $V - K + V\lfloor K/V \rfloor$ vendors, and $\lfloor K/V \rfloor + 1$ to each of the remaining $K - V\lfloor K/V \rfloor$ vendors. This allocation policy might be used when the manufacturer has little or no information about each vendor.

- **H2**: Allocation is proportional to $1/c_i$, which favors low-cost vendors. This allocation might be used when the manufacturer has cost information about the vendors, but does not have good estimates of vendors' service rates. Note that if the allocation is fractional, the allocations to vendors 1, 2, and 3 are rounded to the nearest integer, and the allocation to vendor 4 is adjusted accordingly so the total allocation equals K.

- **H3**: Allocation is proportional to $s_i\mu_i/c_i$, which favors vendors with low cost c_i and high maximum expected service rate $s_i\mu_i$. This allocation policy might be preferred when the vendor has accurate estimates both of vendors' costs and service rates. Note that fractional allocations are adjusted as in H2.

- **H4**: All items are allocated to the vendor with the smallest cost c_i. This heuristic is an "all-or-nothing" version of H2.

- **H5**: All items are allocated to the vendor with the largest value of $s_i\mu_i/c_i$. This heuristic is an "all-or-nothing" version of H3.

The time required to compute an allocation based on any of these heuristics is negligible; however, there is a significant loss of optimality, as summarized in Tables 2 through 4.

Table 2 gives the number of times that each heuristic (H1,. . .,H5) performed best out of the five heuristics. The column sums are greater than 20,480, the number of trials for each value of K, since some trials had two or more heuristics giving the same allocation. Table 2 shows that heuristics H4 and H5 generally performed best among the five heuristics. For example, for the case $K = 100{,}000$, heuristic H4 (send all items to the lowest-cost vendor) performed at least as well as the other heuristics in approximately 72.5% of the trials.

Table 2: Number of times each heuristic performed best (among all 5 heuristics)

Heuristic	$K = 100$	$K = 1{,}000$	$K = 10{,}000$	$K = 100{,}000$
H1	235	966	419	519
H2	925	1213	1045	1275
H3	1590	558	1556	1338
H4	12471	14998	14876	14858
H5	12331	10348	9376	9891

Table 3 gives the number of times (out of 20,480) that the cost produced by each heuristic's allocation was exactly equal to the optimal cost. Again, heuristic H4 performs best. For $K = 100$, heuristic H4 gives the optimal cost in nearly 49% of the trials, and for $K = 100{,}000$, it gives the optimal cost in 72.5% of the trials. The values in Table 3 are zero for heuristics H1, H2, and H3, because these heuristics produce proportional allocations; it is unlikely that the optimal allocation will be exactly proportional to the problem data.

Table 3: Number of times each heuristic's cost was equal to the optimal cost

Heuristic	$K = 100$	$K = 1{,}000$	$K = 10{,}000$	$K = 100{,}000$
H1	0	0	0	0
H2	0	0	0	0
H3	0	0	0	0
H4	9976	14432	14528	14848
H5	9512	8174	6763	7401

From the results in Table 2 and Table 3, it may seem that heuristic H4 is a suitable heuristic for solving this allocation problem. However, Table 4 shows the mean relative difference between the costs of the heuristic allocation and the optimal allocation. For $K = 100{,}000$, the cost of heuristic H4's allocation is, on average, 166% away from optimal. In other words, heuristic H4 performs well quite often, but can also perform extremely poorly. This is a result of H4's myopic "all-or-nothing" structure. It chooses a single vendor based on cost parameters alone, and if the cheapest vendor also happens to be very slow, the results

can be disastrous. Table 4 shows that heuristic H5 appears to be the best heuristic overall, but even this allocation method is anywhere from 23% to 45% away from optimal.

Table 4: Mean relative difference between costs of heuristic and optimal allocations

Heuristic	$K = 100$	$K = 1{,}000$	$K = 10{,}000$	$K = 100{,}000$
H1	0.8354	0.8313	1.1189	1.0345
H2	0.6790	0.6409	0.8819	0.8056
H3	0.3337	0.4768	0.4775	0.5206
H4	1.1574	1.5569	1.4259	1.6621
H5	0.2296	0.3998	0.4214	0.4495

4.2.2 Dynamic Heuristics

We chose two simple dynamic allocation heuristics for comparison with the optimal static allocation algorithm.

- **Join the Shortest Queue (JSQ)**: At the time of failure, an incoming item is sent to a vendor with the shortest queue length. If more than one vendor has minimal queue length, the item is sent to the vendor among them with a smallest value of the fixed cost, c_i.

- **Individually Optimal (IO)**: At the time of failure, an incoming item is sent to a vendor for which the total cost associated with that particular item alone is minimal; in other words, this heuristic myopically routes the incoming items. Specifically, the item is sent to a vendor with the smallest value of $IO_i(x_i)$, where

$$IO_i(x_i) = \begin{cases} c_i + \dfrac{h_i}{\mu_i}, & x_i < s_i, \\ \\ c_i + \dfrac{(x_i+1)h_i}{s_i\mu_i}, & x_i \geq s_i. \end{cases}$$

The relative difference between the costs of optimal static allocation and the two dynamic allocation heuristics are summarized in Table 5. For example, for the 50 examples with $K = 100$, the cost of the optimal static algorithm ranged from 27.55% lower to 17.11% higher than the cost of using join-the-shortest-queue, with an average of 0.21% higher. For

the trials with $K = 100$, the optimal static allocation performed, on average, about as well as the join-the-shortest-queue heuristic, and about 4% worse than the individually optimal dynamic heuristic. However, for the 50 trials with $K = 1{,}000$, the optimal static allocation performed about 1.7% worse than the join-the-shortest-queue heuristic, but more than 5% better than the individually optimal dynamic heuristic.

These results show that one should not blindly use dynamic allocation, with the assumption that any dynamic allocation policy will perform uniformly better than static allocation. In fact, for many instances, one does not lose much by using static allocation rather than simple dynamic allocation, provided one uses the optimal static allocation. Given the simplicity and ease of implementation of static allocation policies as compared to dynamic allocation, it seems that optimal static allocation is, indeed, an attractive policy for minimizing the costs of outsourcing warranty repair services.

Table 5: Relative difference between costs of optimal static allocation and dynamic allocation heuristics, using 50 examples with $K = 100$ and 50 examples with $K = 1{,}000$

	$K = 100$		$K = 1{,}000$	
	JSQ	IO	JSQ	IO
Min	-0.2755	-0.0479	-0.2297	-0.2782
Max	0.1711	0.1822	0.1325	0.0433
Mean	0.0021	0.0396	0.0169	-0.0555
Std. Dev.	0.1026	0.0481	0.0552	0.0938

4.3 Single-Server Approximation

In finding the optimal static allocation, most of the computation time involves the calculation of $L(x)$ required in the cost function. For the multi-server case, the calculation for $L(x)$ in Section 2.1 is quite complicated; however, this is greatly simplified when there is only a single server. Therefore, in this section we explore the tradeoff between loss of optimality and increased speed of calculation when approximating a vendor with s_i servers and rate μ_i by a vendor with a single server and service rate $s_i\mu_i$.

First, note that for all values of K in the computational study reported in Section 4, the values of s_i for each of the four vendors were restricted to the interval from 1 to 20. This is to model the realistic setting where a vendor may have several repairpeople—it is quite

common for a vendor to have up to a few dozen repair employees, but it is difficult to imagine any situation where a repair vendor has thousands, or even hundreds, of employees assigned to a single manufacturer. In this section, we will extend this a bit further, to $s = 50$, in order to show what will happen with the single-server approximation of the cost of an individual vendor as the number of servers increases within a realistic range.

For example, suppose the failure rate is 1.2 failures per item per year and the holding cost is \$1,000 per year, and consider a repair vendor that has a fixed cost of \$100 per repair and an individual service rate of 12,500 repairs per year. Suppose the number of servers ranges from 1 to 50; when there is only a single server, the maximum number of items that the vendor can handle in order to satisfy (infinite-population) stability conditions is 10,416. Figure 2 shows the relative difference between the true cost (using the exact calculation for $L(x)$) and the approximate cost (using a single-server approximation in the calculation for $L(x)$) as a function of s when there are 10,000 items assigned to the vendor. The gap clearly increases with s; however, in even the worst case (corresponding to $s = 50$), the difference between the true cost and the approximation is less than 0.07%.

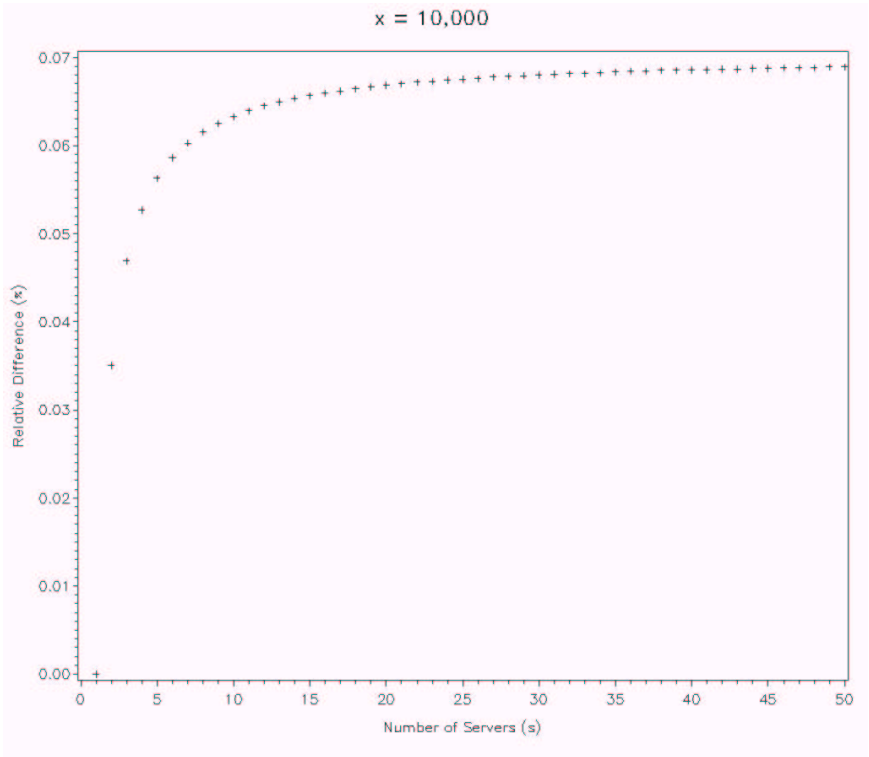

Figure 2: Relative difference between true cost and single-server approximate cost

Furthermore, plots of relative difference between the two costs even for the $x = 100$ and $x = 1{,}000$ are nearly identical to Figure 2; therefore, the plots are not included here. However, both of these cases also have a maximum relative difference of approximately 0.07%. In other words, using a single-server approximation to compute $L(x)$ provides a very close approximation of the cost function, provided the number of servers is within a realistic range.

As seen in Figure 2, the cost function for a single vendor does not change much when using a single-server approximation. Therefore, one would expect that the allocation to the vendors also will show little change if we use a single-server approximation for each of the repair vendors. For the 81,920 trials described in Section 4, we determine the approximate policy using a single server for each vendor and service rate $s_i\mu_i$, $i = 1, \ldots, V$, and we then compute the cost to the manufacturer of following this approximate policy. In 72.9% of the experiments, the allocation under the single-server approximation was exactly the same as the optimal allocation. Furthermore, this number increases with K: for $K = 100$, it is 57.1%, and for $K = 100{,}000$, it is 79.7%. In only 13.2% of the experiments was the difference between the optimal and approximate allocations for any vendor greater than 10, and this number decreases with K: for $K = 100$, it is 28.2%, and for $K = 100{,}000$, it is only 5.1%.

Table 6 shows the mean relative difference between the cost of the optimal allocation and the cost of using the allocation prescribed by the single-server approximation. For $K = 100$, the cost of the single-server approximate allocation is about 2% away from optimal. However, for the larger values of K, this difference is negligible (e.g., 9.06E-9 for $K = 100{,}000$).

Table 6: Mean relative difference: cost of optimal allocation vs. cost of single-server approximate allocation

K	Rel. Diff
100	0.02137
1,000	0.00036
10,000	1.58E-6
100,000	9.06E-9

Table 7 gives the average running time for determining the single-server approximate allocation. Note that the running time is again nearly linear in K, and is approximately 62% to 68% faster (depending on K) than finding the optimal allocation. However, this includes

only the time required to determine the allocation, and not the time required to compute the true expected cost of using this allocation. Computing the true expected cost of the allocation would bring the total running time near, or even above, that required to compute the optimal allocation (which automatically computes the expected cost). Therefore, this approximation may be advantageous when an approximation to the expected cost of the policy is sufficient. In addition, the results in this section show that the single-server approximation is also adequate in cases where the manufacturer may not have detailed information about s_i and μ_i for each vendor, but rather knows only the overall service rate at each vendor.

Table 7: Mean running time to determine single-server approximate allocation

K	Time
100	0.0090
1,000	0.0795
10,000	0.7943
100,000	7.9280

4.4 Effect of Service Rates

Finally, we compare the optimal allocations when we change all vendors from relatively slow service rates to faster service rates. That is, we compare $\underline{\mu} = (\underline{\mu}_1, \underline{\mu}_2, \underline{\mu}_3, \underline{\mu}_4)$ with $\overline{\mu} = (\overline{\mu}_1, \overline{\mu}_2, \overline{\mu}_3, \overline{\mu}_4)$, keeping all other data unchanged. This gives us $5(2^8) = 1280$ experiments for each value of K, for a total of 5120 trials. In approximately 43% of the 5120 trials, the allocation was exactly the same using $\underline{\mu}$ or $\overline{\mu}$. In particular, the optimal solution using $\underline{\mu}$ was 100% allocation to a single vendor. Therefore, one vendor was already fast enough to handle all items at relatively low cost, so increasing the service rates of the vendors did not change the allocation.

In the remaining 57% of the trials, the allocation did change when using $\overline{\mu}$ as compared to $\overline{\mu}$. The pattern we see from these results is that increasing the service rate for all vendors serves to shrink the vendor base. That is, if allocation is positive for exactly n of the V vendors when the service rates are given by $\underline{\mu}$, then the allocation will be positive for $m \leq n$ vendors when the service rates are given by $\overline{\mu}$. For example, for $K = 100$ the allocation might switch from $\mathbf{x}^* = (11, 35, 31, 23)$ when the service rates are given by $\underline{\mu}$ to $\mathbf{x}^* = (0, 0, 42, 58)$

when the service rates are given by $\overline{\mu}$. This is intuitive, since increasing the service rates means the manufacturer can allocate more items to any given vendor without significantly increasing the waiting times at the vendor; hence, fewer vendors will be required to handle all K items. Therefore, the manufacturer has an added advantage when the vendors have fast service rates: not only are the holding costs lower, but the manufacturer will require fewer vendors, which will likely reduce administration costs of maintaining the contracts.

5 Conclusions

In this paper, we consider the static allocation of items under warranty to alternative repair vendors. Modeling the costs to include both a fixed cost for repair as well as a goodwill holding cost to discourage extraordinarily long turnaround times, we develop an efficient optimal algorithm, where each vendor is modeled as a multi-server finite population queue. This algorithm is not specific to the application of outsourcing warranty repairs, but can be used for any static allocation problem that follows this cost and decision structure.

Through a detailed computational study we demonstrate that the optimal algorithm can handle real problem size and also performs much better than common static allocation heuristics. Furthermore, we show that the performance of the optimal static allocation is comparable to that of two simple dynamic allocation heuristics, join-the-shortest-queue and myopic routing. When factoring in the simplicity of finding the optimal static allocation, as well as the relative ease of implementing a static allocation policy, we see that optimal static allocation is indeed a viable choice for the manufacturer seeking to minimize outsourcing costs associated with warranty repairs. Furthermore, there are often situations in which static allocation is is the only option, when the manufacturer does not have reliable information about the current state at each repair vendor.

Among other computational insights we show that when the number of items to be allocated is large, a single-server approximation leads to optimal allocations in most of the cases. This approximation may be necessary when the manufacturer does not have detailed information about a vendor (such as the number of servers and individual service rate), but rather knows only the overall service rate, or effective capacity, of the vendor. Note, however, that this approximation is valid if the number of servers is comparatively small; when the number of servers is very large compared to the number of items allocated to the vendor, an infinite-server approximation may be more appropriate than a single-server approximation.

There are many possible future directions to this research, which we plan to explore in separate papers. First, one can explicitly formulate the dynamic allocation problem as a continuous time Markov decision process. Due to the size and nature of the state space, the problem of finding an optimal dynamic policy is intractable. However, we plan to use techniques of policy improvement and restless bandit models to develop more sophisticated dynamic allocation policies than the simple heuristics used in Section 4.2.2. The results of this investigation will be published elsewhere.

One other key assumption of this paper is that the manufacturer has a fixed number of items to allocate to the repair vendors, which results in a closed population model. In reality, the number of items covered under warranty might not be fixed. For example, items enter the population as they are sold with a warranty, while other items exit the population as the warranty coverage expires. We plan to explore static allocation as it applies to a changing population size. That is, each item is assigned to a vendor at the point of sale, and every time an item experiences a failure, it is sent to its assigned vendor for repair. When allocating a new item, the manufacturer would know how many items are already allocated to each vendor; however, the number of items actually being repaired at each vendor is not observed.

Appendix

The data for the computational study in Section 4 are shown in the following four tables.

Table 8: Data for $K = 100$

	Group 1		Group 2		Group 3		Group 4		Group 5	
	Low	High	Low	High	Low	High	Low	High	Low	High
c_1	61.96	157.95	53.74	142.18	55.21	156.39	68.74	136.34	67.92	148.64
c_2	67.78	139.79	67.26	131.73	57.36	138.43	62.44	142.75	59.81	139.28
c_3	55.60	162.36	60.16	159.83	56.67	153.66	59.99	156.30	58.92	147.37
c_4	46.74	173.32	52.48	184.44	47.14	171.87	52.52	184.59	50.79	176.17
μ_1	7.86	114.84	17.31	106.76	17.24	103.86	28.10	122.55	19.62	122.05
μ_2	36.31	121.02	23.67	125.29	30.14	112.45	29.09	112.73	33.22	120.57
μ_3	24.37	125.54	31.14	125.85	23.89	128.15	22.14	126.12	25.88	123.27
μ_4	37.53	160.63	30.84	153.60	28.18	162.53	32.65	162.77	28.06	155.17
s_1	3	11	1	11	3	10	1	11	1	11
s_2	2	15	1	17	1	12	1	14	1	16
s_3	2	13	4	14	5	16	3	13	5	13
s_4	1	5	2	6	1	8	2	9	1	7

Table 9: Data for $K = 1{,}000$

	Group 1		Group 2		Group 3		Group 4		Group 5	
	Low	High	Low	High	Low	High	Low	High	Low	High
c_1	57.22	151.98	57.08	148.10	56.02	144.55	65.72	156.69	61.70	147.00
c_2	61.60	132.25	56.37	135.24	59.84	134.34	59.99	145.78	60.98	146.47
c_3	57.22	150.48	51.51	158.45	56.18	162.12	51.55	156.70	56.96	160.59
c_4	51.80	174.18	49.09	172.90	49.43	164.35	47.84	170.03	47.94	180.51
μ_1	192.56	1505.98	164.23	1494.62	172.24	1484.92	161.62	1474.44	183.77	1504.23
μ_2	354.42	1802.65	351.78	1831.40	363.88	1886.53	352.22	1839.73	327.16	1867.52
μ_3	260.29	1872.17	266.40	1976.67	281.11	1920.60	256.34	1881.34	246.10	1933.35
μ_4	293.02	2789.58	300.54	2818.41	316.07	2800.25	296.71	2828.71	292.63	2786.68
s_1	1	8	3	11	3	12	2	10	3	9
s_2	1	11	2	16	2	12	2	15	2	14
s_3	2	20	4	17	4	7	3	10	2	14
s_4	1	8	1	9	1	7	1	8	2	9

Table 10: Data for $K = 10{,}000$

	Group 1		Group 2		Group 3		Group 4		Group 5	
	Low	High	Low	High	Low	High	Low	High	Low	High
c_1	61.47	159.59	60.00	157.77	63.28	145.57	58.65	156.70	58.07	151.52
c_2	62.91	142.89	62.03	142.18	63.68	135.74	66.60	129.53	61.72	138.93
c_3	69.78	151.38	52.97	160.27	57.38	154.11	53.48	147.12	53.33	157.18
c_4	53.30	188.90	59.50	161.24	55.38	170.13	55.30	172.95	45.85	164.70
μ_1	2122.30	17961.00	2064.06	18055.41	2107.26	18005.26	2083.28	18081.09	2067.61	18066.65
μ_2	1880.42	21047.19	1835.40	20907.20	1849.31	20976.57	1822.75	20989.56	1848.79	20940.04
μ_3	2206.53	16014.71	2216.44	15964.07	2207.74	15924.84	2221.18	16046.04	2204.84	15987.73
μ_4	2161.15	19811.02	2132.91	19899.36	2204.27	19866.39	2140.53	19937.76	2196.55	19923.36
s_1	1	10	1	12	1	10	1	12	2	8
s_2	1	15	2	16	1	14	2	17	1	12
s_3	5	16	4	7	3	11	5	9	5	12
s_4	2	8	1	5	1	7	2	5	2	6

Table 11: Data for $K = 100{,}000$

	Group 1		Group 2		Group 3		Group 4		Group 5	
	Low	High	Low	High	Low	High	Low	High	Low	High
c_1	68.48	144.85	57.12	135.99	61.31	160.78	60.54	145.40	53.64	136.95
c_2	68.00	139.92	58.34	153.69	65.25	140.77	68.89	140.03	61.60	138.17
c_3	46.53	167.47	64.81	154.85	54.10	162.40	57.21	166.45	57.39	149.61
c_4	47.23	178.57	48.11	180.42	48.61	172.51	55.84	177.11	53.83	174.74
μ_1	21036.39	222033.28	21322.31	218480.41	21274.77	222887.26	21455.08	228451.93	21105.24	232381.33
μ_2	23030.79	191618.05	25869.71	188166.68	24899.77	192447.63	22520.36	197853.31	23503.24	201670.44
μ_3	20095.03	207623.40	19963.75	189419.52	15707.37	212650.20	18505.10	198175.58	19055.07	221346.82
μ_4	20943.71	289047.94	21419.29	296212.16	19386.49	290130.16	21519.96	302640.41	21386.20	298925.98
s_1	3	11	2	11	2	9	1	8	3	8
s_2	2	15	2	13	1	12	2	12	1	14
s_3	4	14	2	20	4	14	5	20	4	19
s_4	1	6	2	9	1	7	1	5	1	6

References

[1] F. Baskett, M. Chandy, R. Muntz, and F. Palacious. Open, closed, and mixed networks of queues with different classes of customers. *Journal of the ACM*, 22:248–260, 1975.

[2] W. R. Blischke and D. N. P. Murthy. *Warranty Cost Analysis*. Marcel Dekker, New York, 1994.

[3] B. D. Bunday and R. E. Scraton. The $G/M/r$ machine interference model. *European Journal of Operational Research*, 4:399–402, 1980.

[4] W. C. Cheng and R. R. Muntz. Optimal routing for closed queueing networks. *Performance Evaluation*, 13:3–15, 1991.

[5] M. B. Combé and O. J. Boxma. Optimization of static traffic allocation policies. *Theoretical Computer Science*, 125:17–43, 1994.

[6] L. W. Dowdy, D. L. Eager, K. D. Gordon, and L. V. Saxton. Throughput concavity and response time convexity. *Information Processing Letters*, 19:209–212, 1984.

[7] M. E. Dyer and L. G. Proll. On the validity of marginal analysis for allocating servers in $m/m/c$ queues. *Management Science*, 23:1019–1022, 1977.

[8] B. L. Fox. Discrete optimization via marginal analysis. *Management Science*, 13(3):210–216, 1966.

[9] O. Gross. A class of discrete type minimization problems. Technical Report RM-1644, RAND Corp., 1956.

[10] M. Heinhold. An operational research approach to allocation of clients to a certain class of service institutions. *J. Opl. Res. Soc.*, 29:273–276, 1978.

[11] A. Hordijk and J. A. Loeve. Undiscounted Markov decision chains with partial information: an algorithm for computing a locally optimal periodic policy. *ZOR – Mathematical Methods of Operations Research*, 40:163–181, 1994.

[12] A. Hordijk and J. A. Loeve. Optimal static customer routing in a closed queueing network. *Statistica Neerlandica*, 54:148–159, 2000.

[13] A. Hordijk, J. A. Loeve, and J. Tiggelman. Analysis of a finite-source customer assignment model with no state information. *Mathematical Methods of Operations Research*, 47:317–336, 1998.

[14] T. Ibaraki and N. Katoh. *Resource Allocation Problems: Algorithmic Approaches*. MIT Press, Cambridge, Mass., 1988.

[15] M. Reiser and S. S. Lavenberg. Mean-value analysis of closed multichain queuing networks. *Journal of the Association for Computing Machinery*, 27(2):313–322, 1980.

[16] A. J. Rolfe. A note on marginal allocation in multiple-server service systems. *Management Science*, 17:656–658, 1971.

[17] C. Serant. EMS Companies Going in for Repairs. *EBN*, Dec. 17, 2001.

[18] C. Serant. Solectron to Provide Xbox Support. *EBN*, Oct. 22, 2001.

[19] J. G. Shanthikumar and D. D. Yao. Optimal server allocation in a system of multi-server stations. *Management Science*, 33:1173–1180, 1987.

[20] J. G. Shanthikumar and D. D. Yao. On server allocation in multiple center manufacturing systems. *Operations Research*, 36:333–342, 1988.

[21] J. G. Shanthikumar and D. D. Yao. Second-order properties of the throughput of a closed queueing network. *Mathematics of Operations Research*, 13:524–534, 1988.

[22] P. H. Zipkin. Simple ranking methods for allocation of one resource. *Management Science*, 26:34–43, 1980.