# Outsourcing Prioritized Warranty Repairs

Peter S. Buczkowski
University of North Carolina at Chapel Hill
Department of Statistics and Operations Research

Mark E. Hartmann
University of North Carolina at Chapel Hill
Department of Statistics and Operations Research

Vidyadhar G. Kulkarni †
University of North Carolina at Chapel Hill
Department of Statistics and Operations Research

***Keywords*** *Warranty, Outsourcing, Network, Resource Allocation*

***Abstract*** *We consider the problem of outsourcing warranty repairs to outside vendors when items have priorities in service. The manufacturer has a contract with a fixed number of repair vendors. The manufacturer pays a fixed fee for each repair done by a vendor which is independent of the repair type and priority class but depends on the vendor. There are a fixed number of items under warranty, and each item belongs to one of a fixed number of priority classes. The manufacturer also pays for holding costs incurred when the items are at the vendors, the holding cost being higher for the higher priority items. We focus on static allocation of the warranty repairs; that is, we assign all items to the vendors at the beginning of the warranty period. We give the known algorithm to optimally solve the one priority class problem and solve the multi-priority class problem by formulating it as a convex minimum cost network flow problem. Then, we give numerical examples to illustrate the cost benefits of a multi-priority structure.*

## 1 Overview

When designing product warranties, the manufacturers must decide on many issues, such as warranty policy, length of warranty period, repair policy, and quality control. They also have to plan to cover the costs associated with the warranty. An issue of critical importance to the manufacturers is managing the costs associated with the warranty effectively.

We consider the problem of outsourcing warranty repairs to outside vendors when items have priority levels. For example, some warranty contracts specify the repair turnaround time (e.g., one day, three days, or seven days). With careful management, repair outsourcing can be a major benefit to the manufacturer. A smooth operation can improve customer satisfaction and turnaround times while allowing the manufacturer to maintain its focus on production. While the manufacturer may have a central repair depot, it often is not effective to ship items to the depot due to time and cost constraints. Thus it might be beneficial to choose repair vendors distributed geographically so as to be close to the customers. The manufacturer must seek a balance between cost savings and

†This research was partially supported under NSF grant DMI-0223117.

customer service. If not, some customers will be lost because of poor service. Repair outsourcing is an especially important problem when considering priorities because high priority customers will typically inflict greater loss if the manufacturer does not meet their expectations.

We assume that the manufacturer assigns each item to one of the repair vendors at time 0. This is *static allocation*; the items are all assigned at a single time point. One example might be the sale of small electronic appliances. Typically, the warranty card inside the product package will have a phone number to contact for warranty repairs. When the manufacturer outsources the warranty repairs, this phone number might be a direct line to the repair vendor. In this case, the manufacturer assigns the items to a vendor at production time.

## 2 Literature Review

Warranty theory has been heavily studied over the past two decades. Blischke and Murthy (1994, 1996) wrote comprehensive references for the subject. They discuss many different types of warranty policies, including many warranty policies currently implemented in industry. Numerous cost and optimization models are developed from both the consumer's and the manufacturer's point of view, including life cycle and long-run average cost models. We use these models to compute the expected warranty cost of a product in our numerical examples. Their comprehensive literature review of the current research can be found in Murthy & Djamaludin (2002).

Warranty costs affect both the buyer and the seller. Mamer (1982) wrote the first paper to provide a comprehensive model of both the buyer's and seller's expected costs and long-run average costs for the free replacement warranty. Our research focuses on the manufacturers' view of warranty costs.

At its most basic structure, the static allocation model reduces to a resource allocation problem with integer variables. Without considering priorities, the problem has a separable objective function. This problem has been widely studied in the literature. Gross (1956) first proposed a simple greedy algorithm to find the optimal solution if the objective is convex. Several authors have since expanded the problem. Ibarki and Katoh (1988) provide a comprehensive review of resource allocation problems and algorithms to solve them. Their bibliography provides a review of the literature up to 1988. Bretthauer and Shetty (1995, 2002) also give a survey of a generalization: the nonlinear knapsack problem. They provide a proof of the greedy algorithm by the generalized Lagrange multiplier method. Zaporozhets (1997) gives an alternate proof of the greedy algorithm. Opp, et al. (2003) describes the greedy algorithm in detail for the convex separable resource allocation problem and its application to our problem without priorities. Some computational issues associated with the application are also discussed, mostly regarding the expected queue length.

Once priorities are considered, the objective is no longer separable. We extend the previous research by providing an algorithm to optimally solve the closed static allocation problem with priorities. Finally, we investigate the benefits of a multi-priority structure for the manufacturer. Another possible assignment method is *dynamic allocation* – the manufacturer assigns an item to a repair vendor at the time of failure. This proves to be a very difficult problem, even when priorities are not considered (Opp 2003). We do not address this allocation method here.

After a brief problem overview, we state the notation and assumptions of the problem in Section 3. In Section 4, we derive the cost function and state the optimization problem for the model. We describe the simple greedy algorithm to solve the single priority problem in Section 5.1. We reformulate the $m$-priority problem as a convex-cost network flow problem in Section 5.2, providing the general algorithm for the $m$-priority case. In Section 6, we discuss the computational issues that arise in the problem and provide an example in the following section. In Section 8, we

illustrate the cost benefits of the priority structure. We provide two examples: the first has very different holding costs between the high and low priority customers; while the second has relatively similar holding costs between the high and low class customers. We complete the discussion of the problem in Section 9 by presenting an optimization problem for the manufacturer when the customer pays additional monies for priority in service.

# 3 Problem Description

Consider a manufacturer that has a contract with $V$ repair vendors for a fixed fee per repair. Specifically, vendor $j$ charges the manufacturer $c_j$ dollars for each repair made under warranty, independent of the type of repair and the priority of the item. The contract does not specify a minimum or maximum number of repairs. Usually this contract situation arises when the manufacturer provides the replacement parts and the vendor just executes the repairs. Another scenario is that the vendor charges the expected cost per repair over the duration of the contract. We only consider a closed population model, i.e., we assume the number of items under warranty at any given time is constant. This model is appropriate for manufacturers that have a similar number of products in circulation at all times. It can also serve as an important benchmark for annual warranty expenses.

For most manufacturers, it is important to return some repaired items faster than others. We provide two examples where the same product with the same failure rate can have different priority in service for different customers. One example is a customer that makes large purchases with the manufacturer. They will expect a faster repair turnaround time than an individual customer. Another example is when the manufacturer includes a maximum repair turnaround time in the purchase contracts with a customer. Also, the manufacturer might offer a choice of warranties that specify the repair turnaround time. For example, the standard product warranty guarantees a one week repair turnaround time but can be upgraded by the customer to a two day repair turnaround time. The length of these turnaround times require that some of the repairs take priority over others. We treat the case where each item belongs to one of $m$ priority classes. Each type $i$ item has priority in service over all types $j > i$. The number $K_i$ of items of priority type $i$ is constant, and a total of $K = \sum_{i=1}^{m} K_i$ items must be allocated among the $V$ vendors.

We assume that the lifetimes of items are i.i.d. random variables with mean $1/\lambda$ (independent of the priority type), the $j$th vendor employs $s_j$ servers to repair items, and the repair times are i.i.d. exponential random variables with parameter $\mu_j$. We point out an invariance result for a $G/M/r$ interference model that the steady state probabilities depend only on the mean failure time $1/\lambda$ and not the failure distribution $G(\cdot)$ (Bunday & Scranton 1980). Therefore, since we are only concerned with long-run average cost, we need not assume that the lifetimes of items are exponentially distributed. We will assume that all information is perfect, meaning that all vendors know the item failure rate $\lambda$, and the manufacturer knows the service rate $\mu_j$ and the number of repair people, $s_j$, at each vendor. While a priority type $i$ item is in service (or waiting for service) at vendor $j$, it costs the manufacturer $h_{ij}$ dollars per unit time. This can be interpreted as a goodwill cost and is designed to prevent long delays in service. An example is the cost of a loaner while a type $i$ item is in service. We assume that the holding cost is increasing for increasing priority type for each vendor. That is,

$$h_{1j} > h_{2j} > \ldots > h_{mj} \quad (j = 1, \ldots, V).$$

This agrees with the intuition that items are given priority in service because of higher holding costs. Typically the holding cost for each priority type is independent of the vendor.

The overall goal of the manufacturer is to minimize their expected long-run average warranty cost. Given the above information, the manufacturer must decide on the optimal allocation matrix $X = [x_{ij}]$, where $x_{ij}$ represents the number of priority class $i$ items assigned to vendor $j$ for $i = 1, \ldots, m$ and $j = 1, \ldots, V$. We assume that this allocation, made at the beginning of the product life cycle, remains in effect during the entire contract period.

# 4 Problem Formulation

Based on the assumed item failure and service distributions of the previous section, we can model vendor $j$ as an $M/M/s_j/\cdot/x_{\cdot j}$ finite population queue with $m$ priority classes, where $x_{\cdot j} = (x_{1j}, \ldots, x_{mj})$. The priority structure gives priority class $i$ a preemptive resume priority over classes $j > i$. This means that when a type $i$ item enters the service queue and a type $j > i$ is in service, the service of the type $j$ item is preempted and is resumed after all higher priority items are serviced. Preemptive resume priority allows for easy calculation of the expected queue lengths at each vendor.

The long-run average warranty cost to the manufacturer consists of both repair costs and holding costs. First, we compute the long-run average repair cost. Let $y_{kj} = \sum_{i=1}^{k} x_{ij}$ be the total number of items of priorities $1, \ldots, k$ assigned to server $j$, and let $L_j(y)$ be the expected queue length of items (of all priorities) at vendor $j$ when $y = y_{mj}$ items are allocated to it. This can be computed by using the birth and death process analysis for an $M/M/s_j/\cdot/y$ queue. The expected number of properly functioning items at any particular time is $y - L_j(y)$. Since each functioning item has failure rate $\lambda$, the expected number of arrivals per unit time to the $j^{th}$ vendor is $\lambda\left(y - L_j\left(y\right)\right)$. Each such arrival costs the manufacturer $c_j$ dollars. Hence, the long-run average repair cost for vendor $j$ is given by

$$\lambda c_j \left(y_{mj} - L_j(y_{mj})\right). \tag{1}$$

Next we compute the long-run average holding cost. Since the holding cost depends on the priority class, we will need an expression for the expected queue length of each priority class. We argue as follows: Priority class 1 items only see other type 1 items in the queue since they preempt service for all other items. Therefore, the expected number of type 1 items in the queue at vendor $j$ is $L_j(y_{1j})$. Type 2 items see both type 1 and type 2 items in the queue. The expected number of type 1 and type 2 items in the queue at vendor $j$ is $L_j(y_{2j})$; therefore, the expected number of type 2 items in the queue at vendor $j$ is $L_j(y_{2j}) - L_j(y_{1j})$. Similar reasoning yields the expected queue length for type $i$ items at vendor $j$ as $L_j\left(y_{ij}\right) - L_j\left(y_{i-1,j}\right)$. Each type $i$ item in the queue at vendor $j$ costs the manufacturer $h_{ij}$ dollars per unit time. Therefore, the long-run average holding cost is given by

$$h_{1j}L_j(y_{1j}) + \sum_{i=2}^{m} h_{ij}\left(L_j\left(y_{ij}\right) - L_j\left(y_{i-1,j}\right)\right) = h_{mj}L_j\left(y_{mj}\right) + \sum_{i=1}^{m-1}\left(h_{ij} - h_{i+1,j}\right)L_j\left(y_{ij}\right). \tag{2}$$

We combine the expressions for repair cost (1) and holding cost (2) to express the total long-run cost rate of all items assigned to vendor $j$, $f_j(x_{1j}, x_{2j}, \ldots, x_{mj})$, as follows:

$$\lambda c_j \left(y_{mj} - L_j\left(y_{mj}\right)\right) + h_{mj} L_j\left(y_{mj}\right) + \sum_{i=1}^{m-1} \left(h_{ij} - h_{i+1,j}\right) L_j\left(y_{ij}\right) \; =$$

$$\lambda c_j y_{mj} + \left(h_{mj} - \lambda c_j\right) L_j\left(y_{mj}\right) + \sum_{i=1}^{m-1} \left(h_{ij} - h_{i+1,j}\right) L_j\left(y_{ij}\right). \tag{3}$$

The manufacturer wishes to minimize the total long-run average cost by outsourcing all warrantied items to the vendors. Therefore, the manufacturer wishes to solve the following optimization problem, $P_m$:

$$\begin{aligned} \text{minimize} \quad & \sum_{j=1}^{V} f_j(x_{1j}, x_{2j}, \ldots, x_{mj}) \\ \text{subject to} \quad & \sum_{j=1}^{V} x_{ij} = K_i \quad (i = 1, \ldots, m) \\ & x_{ij} \geq 0 \text{ and integer} \quad (i = 1, \ldots, m;\ j = 1, \ldots, V) \end{aligned}$$

where $f_j(x_{1j}, x_{2j}, \ldots, x_{mj})$ is given by expression (3).

When $m = 1$, the optimization problem defined above reduces to a standard resource allocation problem with a separable objective, studied by Gross (1956), Ibaraki and Katoh (1988), Bretthauer and Shetty (1995), and Opp et al. (2003). We provide these results in Section 5.1. However, if $m > 1$ the problem is substantially more complicated. Ibaraki and Katoh (1988) mention a dynamic programming procedure to solve the resource allocation problem, but it is essentially the same as enumerating all possible solutions and it is therefore not recommended for large problems. In Section 5.2, we exploit the structure of the objective to develop an algorithm to solve the problem with multiple priority classes.

Computing $L_j(y)$ is the key to evaluating $f_j(x_{1j}, x_{2j}, \ldots, x_{mj})$. Dowdy et al. (1984) provides a result which states that the expected queue length $L(y)$ of a $M/M/s/\cdot/y$ finite population queue is convex in $y$, the size of the finite population. If there is only one server, this reduces to the convexity of the Erlang Loss Function, which was given by Messerli (1972) and Jagers and Van Doorn (1986). They provide a stable technique for computing $L(y)$ by an efficient recursion formula. If there are multiple servers at a vendor, we use mean-value analysis in the closed-queueing system to compute $L(y)$. Both of these computational techniques were described by Opp (2003). We summarize these results in Section 6.

# 5 Solution Method

## 5.1 Single Priority Class

We begin by describing the solution method for the single priority class case. Since there is only one priority class, we omit it from the notation and write $x_{1j} = x_j$ and $h_{1j} = h_j$. The problem

described in the previous section reduces to $P_1$:

$$\begin{aligned} \text{minimize} \quad & \sum_{j=1}^{V} f_j(x_j) \\ \text{subject to} \quad & \sum_{j=1}^{V} x_j = K \\ & x_j \geq 0 \text{ and integer} \quad (j = 1, \ldots, V) \end{aligned}$$

where the objective is now separable (i.e. it is a sum of functions of a single variable each):

$$f_j(x) = \lambda c_j x + (h_j - \lambda c_j) L_j(x).$$

Since $L_j(x)$ is a convex function in $x$, $f_j(x)$ is convex if $h_j - \lambda c_j > 0$ and concave if $h_j - \lambda c_j < 0$. If the functions $f_j(x)$ are all concave, the optimum occurs at an extreme point. The optimal solution can be found by picking the vendor $j$ where $f_j(K)$ is minimum, and allocating all items to vendor $j$. Opp et al. (2003) discuss the case where the functions $f_j(x)$ are mixed convex and concave. If the functions $f_j(x)$ are all convex, this problem is the separable convex resource allocation problem. We assume this is true, since it should be more desirable to repair an item rather than hold it in the repair queue.

Gross (1956) first proved that the following greedy algorithm finds the optimal allocation to problem $P_1$ for a positive integer $K$:

initialize $\boldsymbol{x} := \mathbf{0}$;
while $\sum_{j=1}^{V} x_j < K$ do

- find $k \in \arg\min_{j=1,\ldots,V} \delta f_j(x_j + 1)$;
- increment $x_k$ by 1;

end

where $\delta f(x) = f(x) - f(x-1)$ for $x \geq 1$. Ibaraki & Katoh (1988) provide many algorithms to produce the optimal solution to the separable convex resource allocation problem, some of which run in polynomial time. The complexity of the greedy algorithm is $O(V + K \log V)$, since the minimization step takes $O(\log V)$ time.

## 5.2 Network Flow Formulation

### 5.2.1 Solution to $m$-priority problem

To solve the optimization problem $P_m$ for $m > 1$ priority classes, we reformulate the problem as a minimum convex-cost network flow problem (assuming that $h_{mj} \geq \lambda c_j$ for $j = 1, 2, \ldots, V$). Using

the variables $y_{ij}$ introduced in Section 4, we have

$$\begin{aligned}
\text{minimize} \quad & \sum_{i=1}^{m}\sum_{j=1}^{V} f_{ij}(y_{ij}) \\
\text{subject to} \quad & \sum_{j=1}^{V} x_{ij} = K_i \quad (i = 1, \ldots, m) \\
& y_{1j} = x_{1j} \quad (j = 1, \ldots, V) \\
& y_{ij} = x_{ij} + y_{i-1,j} \quad (i = 2, \ldots, m;\ j = 1, \ldots, V) \\
& x_{ij} \geq 0 \text{ and integer} \quad (i = 1, \ldots, m;\ j = 1, \ldots, V) \\
& y_{ij} \geq 0 \text{ and integer} \quad (i = 1, \ldots, m;\ j = 1, \ldots, V)
\end{aligned}$$

where $f_{ij}(y) = (h_{ij} - h_{i+1,j})L_j(y)$ for $i < m$ and $f_{mj}(y) = \lambda c_j y + (h_{mj} - \lambda c_j)L_j(y)$.

The network model for this problem is provided in Figure 1. This network has $m$ source

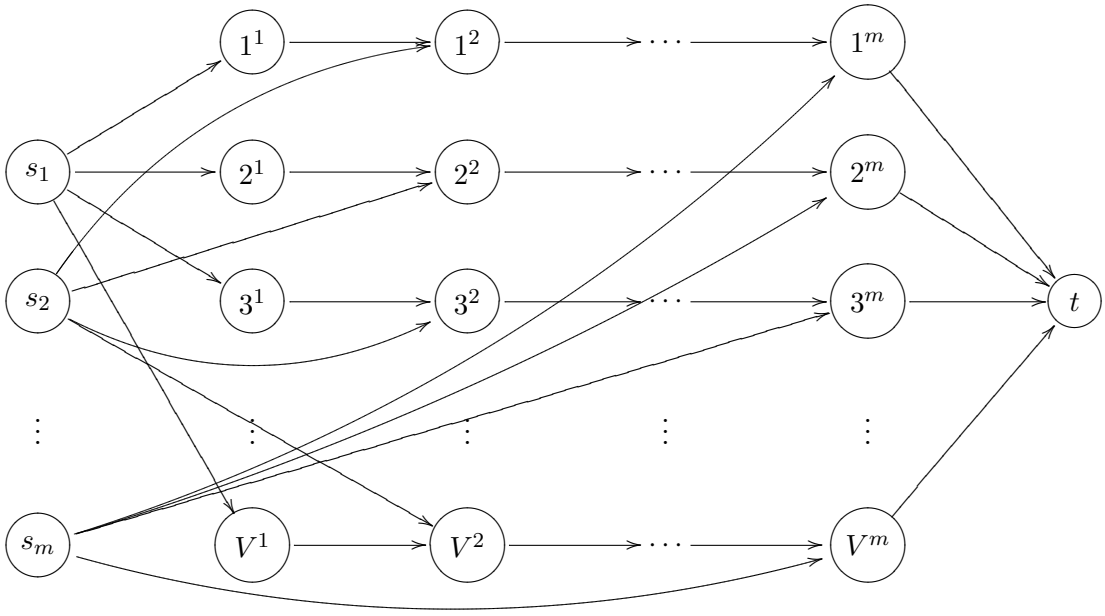


Figure 1: Network Model of $m$ Priority Problem

nodes, labelled $s_1, s_2, \ldots, s_m$, with node $s_i$ having supply $K_i$. There are $m$ transshipment nodes for each vendor $j$ (labelled $j^1, \ldots, j^m$) and a single sink node $t$ with demand $K = \sum_{i=1}^{m} K_i$. The network contains arcs $(s_i, j^i)$ with flow $x_{ij}$ and zero cost for $i = 1, \ldots, m$ and $j = 1, \ldots, V$; arcs $(j^i, j^{i+1})$ with flow $y_{ij}$ for $i = 1, \ldots, m-1$ and $j = 1, \ldots, V$; and arcs $(j^m, t)$ with flow $y_{mj}$ and cost $f_{mj}(y_{mj})$ for $j = 1, \ldots, V$. Note that we could give an equivalent formulation with $m - 1$ nodes for each vendor.

With this formulation, we can apply the successive shortest path algorithm for the minimum convex-cost network flow problem to this network with $mV + m + 1$ nodes and $2mV$ arcs to solve the $m$-priority problem. In the residual network, the cost of an arc with flow $y_{ij}$ will be $\delta f_{ij}(y_{ij}+1)$ and the cost of the reverse arc will be $-\delta f_{ij}(y_{ij})$; hence, in the single priority case this reduces to the greedy algorithm. Our approach to the $m$-priority problem is to push flow from $s_i$ to $t$ in the reduced network $G_i$, which omits the source nodes $s_1$ through $s_{i-1}$ and their incident arcs. This

procedure begins by pushing flow from $s_m$ to $t$ (using the greedy algorithm) and continues until we push the flow from $s_1$ to $t$ over the full network. We outline the algorithm to solve the $m$-priority problem below:

***m*-priority Successive Shortest Path Algorithm:**

initialize $\boldsymbol{x}$,$\boldsymbol{y} := \mathbf{0}$;
for $i = m$ down to 1

while $\sum_{j=1}^{V} x_{ij} < K_i$ do

- determine the shortest path $P$ from node $s_i$ to node $t$ in the residual network $G_i(\boldsymbol{x},\boldsymbol{y})$ constructed from the reduced network $G_i$;
- augment 1 unit of flow along the path $P$;
- update $\boldsymbol{x}$, $\boldsymbol{y}$ and $G_i(\boldsymbol{x},\boldsymbol{y})$;

end while loop;

end

If we use Dijkstra's algorithm (Ahuja, Magnanti & Orlin 1993, Chapter 4) to solve the shortest path problem at each iteration, this implementation of the successive shortest paths algorithm will have complexity $O(KmV \log mV)$. With typical parameter values ($m \leq 4$ and $V \leq 10$), the algorithm is quite efficient. For an example with $m = 4$, $V = 6$, and $K = 10,000$, the algorithm took about 6 seconds on a laptop machine with a pentium M processor and a 1 GB RAM. Using the capacity-scaling algorithm described in Ahuja, Magnanti, and Orlin (Ahuja et al. 1993) to solve the network flow formulation would reduce the complexity to $O(\log K\, m^2V^2 \log mV)$, but it should be noted that computing the functions $L_j(y)$ recursively takes $\Omega(K)$ time over the course of the algorithm.

### 5.2.2 Solution to 2-priority problem

For the 2-priority problem, the simple structure of the network model allows us to give a more explicit description of the successive shortest paths algorithm. In the first stage of the algorithm, $K_2$ units of flow are pushed from node $s_2$ to the sink using the greedy algorithm. In the second stage, there will be only two types of augmenting paths in the residual network: direct paths from $s_1$ to $t$ via vendor $j$ with cost $\delta f_{1j}(x_{1j} + 1) + \delta f_{2j}(x_{1j} + x_{2j} + 1)$ for $j = 1, \ldots, V$, and paths from $s_1$ to $t$ that go through node $s_2$. In the latter case, sub-path from $s_1$ to $s_2$ will be via a vendor $p$ such that

$$p \in \arg\min_{x_{2j}>0} \delta f_{1j}(x_{1j} + 1)$$

(There is no cost to push flow from vendor $j$ to $s_2$, but there will only be a reverse arc if $x_{2j} > 0$.) The sub-path from $s_2$ to $t$ will be via a vendor $q$ such that

$$q \in \arg\min_{j=1,\ldots,V} \delta f_{2j}(x_{1j} + x_{2j} + 1).$$

Augmenting a unit of flow along this path requires that we increase $x_{1p}$ and $x_{2q}$ by 1 unit and decrease $x_{2p}$ by 1 unit. Therefore, we have the following algorithm to solve the 2-priority problem:

**2-priority Successive Shortest Path Algorithm:**

initialize $\boldsymbol{x}:=\mathbf{0}$;
while $\sum_{j=1}^{V} x_{2j} < K_2$ do

- find $k \in \arg\min_{j=1,\ldots,V} \delta f_{2j}(x_{2j}+1)$;
- increment $x_{2k}$ by 1;

end;
while $\sum_{j=1}^{V} x_{1j} < K_1$ do

- find $k \in \arg\min_{j=1,\ldots,V} \{\delta f_{1j}(x_{1j}+1) + \delta f_{2j}(x_{1j}+x_{2j}+1)\}$, $p \in \arg\min_{x_{2j}>0} \delta f_{1j}(x_{1j}+1)$ and $q \in \arg\min_{j=1,\ldots,V} \delta f_{2j}(x_{1j}+x_{2j}+1)$;
- if $\delta f_{1k}(x_{1k}+1) + \delta f_{2k}(x_{1k}+x_{2k}+1) < \delta f_{1p}(x_{1p}+1) + \delta f_{2q}(x_{1q}+x_{2q}+1)$ then increment $x_{1k}$; else increment $x_{1p}$ and $x_{2q}$ by 1 and decrement $x_{2p}$ by 1 unit

end

This algorithm has the advantage of being very easy to code. It is very similar to the greedy algorithm, since it adds one item at a time, with the possibility of rearranging the lower priority items previously assigned. The complexity of the algorithm is $O(V + K\log V)$. For $m > 2$, this approach can be generalized to solve the $m$-priority problem in time $O(m^2V + Km^3\log V)$.

# 6 Computational Issues

In this section, we discuss some of the computational issues related to the outsourcing problem. First, we compute the expected queue length $L(N)$ in a $M/M/s/\cdot/N$ finite population queue. When $s = 1$, $L(N) = N - \rho + \rho B(\rho, N)$, where $\rho = \mu/\lambda$ and

$$B(\rho, N) = \frac{\rho^N/N!}{\sum_{j=0}^{N} \rho^j/j!}$$

is the Erlang loss function (see (Jagerman 1974)). When $N$ is large, directly calculating $B(\rho, N)$ is difficult. Therefore, we use the following recursion to compute $B(\rho, N)$:

$$B(\rho, N) = \frac{\rho B(\rho, N-1)/N}{1 + \rho B(\rho, N-1)/N} \tag{4}$$

for $N \geq 1$, where $B(\rho, 0) = 1$.

Opp et al. (2003) dedicate a section to discussing calculation issues when there are multiple servers at each vendor. They show that a queue with $s$ servers working at rate $\mu$ can be well approximated by a single server queue working at rate $s\mu$. This approximation works best when the servers for the particular vendor are rarely idle, which is common in practice. If the exact expected queue length is needed when $s > 1$, we can use an analysis of a two station closed queueing network (Gross & Harris 1985, Section 2.7). Opp (2003) gives a recursive method of

computing the expected queue length in this case. We provide those results in the appendix for the sake of completeness.

# 7 An Example

We provide an example to illustrate the successive shortest paths algorithm. Consider a manufacturer that has warrantied items of $m = 4$ different priority types. The number of items of each priority type is $K_1 = 150$, $K_2 = 250$, $K_3 = 200$, and $K_4 = 400$ (for a total of $K = 1000$ items). Each item has a common failure rate of $\lambda = 1$ failure per year. There are $V = 6$ vendors at her disposal. Each vendor $j$ has a single repairperson. The vendors' properties are summarized in Table I. Note that for every vendor $j$, $h_{ij} > h_{i+1,j}$ $(i = 1, \ldots, 3)$ and $h_{4j} > \lambda c_j$, so the convexity properties for the $f_{ij}$ functions are satisfied.

Table I: Costs and Service Rates for Each Vendor

| Vendor $j$ | $\mu_j$ | $c_j$ | $(h_{1j}, h_{2j}, h_{3j}, h_{4j})$ |
|---|---|---|---|
| 1 | 80 | 15 | (500, 350, 300, 175) |
| 2 | 62 | 19 | (500, 400, 250, 175) |
| 3 | 70 | 18 | (500, 350, 300, 160) |
| 4 | 50 | 15 | (500, 400, 250, 160) |
| 5 | 45 | 14 | (500, 400, 300, 175) |
| 6 | 25 | 9 | (500, 350, 300, 175) |

We apply the successive shortest paths algorithm to solve the 4-priority problem with these parameters. The allocation matrix below is the solution, where the row indicates the priority type and the column indicates the vendor. For example, the number of type 3 items assigned to vendor 4 is 80.

$$X = \begin{pmatrix} 39 & 34 & 31 & 24 & 21 & 1 \\ 62 & 33 & 56 & 30 & 33 & 36 \\ 0 & 120 & 0 & 80 & 0 & 0 \\ 0 & 0 & 300 & 95 & 5 & 0 \end{pmatrix}$$

The expected yearly cost of this allocation is $\$146,012.42$.

# 8 Cost Benefits of the Multi-Priority Approach

In this section, we discuss the benefits of giving priority in service to items that have higher holding costs. We first illustrate the benefits on the example in the previous section. Suppose there was no priority structure in place, but the holding costs are still given as in Section 7. Consider a specific vendor $j$. If $N$ items are randomly assigned to vendor $j$ ignoring priority levels, the expected number of type $i$ items assigned to vendor $j$ is $NK_i/K$. If there is no priority structure, the expected waiting time will be independent of the priority type. Therefore, the expected holding cost per item $\hat{h}_j$ assigned to vendor $j$ is

$$\hat{h}_j = \left( \frac{NK_1}{K} h_{1j} + \ldots + \frac{NK_V}{K} h_{Vj} \right) \Big/ N = \frac{\sum_{i=1}^{m} K_i h_{ij}}{K}. \tag{5}$$

**Example 1** We apply the formula in (5) to get the following average holding costs for the vendors in the Example in Section 7. Recall that this example had six vendors and four priority classes.

Table II: Average Holding Costs for Vendors

| Vendor $j$ | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| $\hat{h}_j$ | 292.5 | 295 | 286.5 | 289 | 305 | 292.5 |

Now we solve the single priority problem with the holding costs from Table II, and repair costs and service rates from Table I on $K = 1000$ items. The optimal allocation vector is (106, 83, 637, 73, 61, 40) and the long-run average yearly cost is $\$197,520.56$. This represents a 35.3% increase over the cost with the priority structure in place. The reason for the increase is that the higher priority items are in service for a longer portion of time and hence create a higher overall holding cost. ■

**Example 2** We next turn to an example where there are only two priority classes and their holding costs differ by only 10%. We consider five vendors and $10,000$ total items, with $K_1 = 2500$ and $K_2 = 7500$, so the majority of the items are low priority items. The items fail at rate two per year. The five vendors have the following properties:

Table III: Vendor Properties for Similar Cost Example

| Vendor $j$ | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| $c_j$ | 20 | 18 | 23 | 16 | 25 |
| $s_j\mu_j$ | 1000 | 200 | 400 | 600 | 700 |
| $h_{1j}$ | 220 | 203.5 | 198 | 231 | 209 |
| $h_{2j}$ | 200 | 185 | 180 | 210 | 190 |
| $\hat{h}_j$ | 205 | 189.62 | 184.5 | 215.25 | 194.75 |

When the type 1 items are not given priority in service, the allocation vector is (1063, 238, 7293, 644, 762) and the total long-run average cost is $\$1,374,210$ per year. On the other hand, when the type 1 items are given priority in service we apply the successive shortest paths algorithm, which produces the following allocation matrix:

$$X = \begin{pmatrix} 890 & 152 & 335 & 512 & 611 \\ 173 & 86 & 6958 & 132 & 151 \end{pmatrix}.$$

In this case, the long-run average cost is $\$1,342,645$ per year. The cost savings using the priority structure is 2.35%. Note that in this example the total number of items allocated to each vendor is the same regardless of the priority structure used. Clearly the priority structure is more beneficial when high priority items dramatically affect the holding cost. However, even in the case of similar holding costs among the different priority types, the priority structure can reduce cost by a small amount with relatively little effort. ■

## 9 Selecting the Values of $K_i$

Suppose that the manufacturer has $K$ items to sell. The manufacturer has the option of offering up to $m$ different warranty contracts. The contracts dictate the priority level: the most expensive contract will guarantee first priority in service, the second most expensive contract guarantees second priority in service, etc. The manufacturer charges the customer $r_i$ dollars for each item to have a type $i$ priority in service ($i = 1, \ldots, m$). The values of $r_i$ are fixed and are determined by the competition and the manufacturer cannot control these. We assume that the manufacturer can sell any number of items under any warranty contract. That is, he will sell all of the items regardless of the warranty contract(s) he offers. Typically, these assumptions are satisfied if the manufacturer is a small player in a large market.

Now, the manufacturer must decide on the values of $K_i$ ($i = 1, \ldots, m$) in addition to the optimal allocation matrix. The optimization problem is:

$$\begin{aligned}
\text{minimize} \quad & \sum_{j=1}^{V} f_j(x_{1j}, x_{2j}, \ldots, x_{mj}) - \sum_{i=1}^{m} r_i K_i \\
\text{subject to} \quad & \sum_{i=1}^{m} K_i = K \\
& \sum_{j=1}^{V} x_{ij} = K_i \quad (i = 1, \ldots, m) \\
& x_{ij} \geq 0 \text{ and integer} \quad (i = 1, \ldots, m;\ j = 1, \ldots, V) \\
& K_i \geq 0 \text{ and integer} \quad (i = 1, \ldots, m)
\end{aligned}$$

We can again model the problem as a minimum convex-cost network flow problem, similar to the problem discussed in Section 4 with the addition of a single super-source node $S$ with supply $K$ and arcs $(S, s_i)$ with flow $K_i$ and cost $-r_i$. The nodes $s_i$ ($i = 1, \ldots, m$) become transshipment nodes, but otherwise the rest of the network is the same as described in Section 4. We can then solve this problem using the successive shortest path algorithm, augmenting the flow one unit at a time along shortest paths from $S$ to $t$, with the same orders of complexity.

**Example 3** We illustrate this using the example of Section 7, with $r_i = (15, 10, 5, 0)$. Recall that the original values of $K_i$ were $(150, 250, 200, 400)$ and the expected yearly cost was $\$146,012.42$. If we use these values of $K_i$ with the above values of $r_i$, the expected yearly cost is now $\$140,262.40$ after deducting the additional policy costs that the manufacturer charges. However, using the successive shortest path algorithm gives us an optimal allocation of

$$X = \begin{pmatrix} 34 & 25 & 27 & 17 & 16 & 0 \\ 13 & 0 & 10 & 0 & 0 & 6 \\ 0 & 11 & 0 & 6 & 0 & 0 \\ 52 & 42 & 573 & 95 & 43 & 30 \end{pmatrix},$$

with values of $(119, 29, 17, 835)$ for $K_i$. The cost of allocation is $\$112,326.61$, a 20% reduction in cost over the previous solution. ■

## 10 Conclusions and Future Work

In this paper, we addressed the problem of outsourcing warranty repairs when items have priority in service. We provided the known result for the single priority case and used a network flow model to solve the multi-priority problem. We also provided a numerical comparison of the single-priority problem versus the multi-priority problem. We mention several suggestions for future work below:

- **Different Priority Structures:** We can consider other priority structures, such as non-preemptive priority. In this case, the computation of the expected queue lengths is more complicated and we lose the special structure of the objective.
- **Enforce Maximum Waiting Times:** If the contracts specify a maximum repair turnaround time for the different priority types, we should incorporate this into the model.
- **Simulation:** We can use simulation to determine if our allocation strategy performs well under different failure and service distributions.
- **Game Theoretic Problems:** In our research, we assumed that the manufacturer pays a fixed fee per repair to each vendor. Often these contracts are negotiated between the manufacturer and the vendor and may depend on the number of items allocated to that vendor.
- **Dynamic Allocation:** Another possible allocation strategy is dynamic allocation, i.e. the items are allocated only after a claim is made. Opp (2003) uses index policies to address the single priority case. We are interested in how index policies can be used to estimate the optimal solution in the multi-priority case.

## References


Ahuja, R. K., Magnanti, T. L. & Orlin, J. B. (1993), *Network Flows: Theory, Algorithms, and Applications*, Prentice Hall, New Jersey.

Blischke, W. R. & Murthy, D. N. P. (1994), *Warranty Cost Analysis*, Marcel Dekker, New York.

Blischke, W. R. & Murthy, D. N. P. (1996), *Product Warranty Handbook*, Marcel Dekker, New York.

Bretthauer, K. M. & Shetty, B. (1995), 'The nonlinear resource allocation problem', *Operations Research* **43**(4), 670–683.

Bretthauer, K. M. & Shetty, B. (2002), 'The nonlinear knapsack problem - algorithms and approaches', *European Journal of Operations Research* **138**(3), 459–472.

Bunday, B. D. & Scranton, R. E. (1980), 'The $G/M/r$ machine interference model', *European Journal of Operations Research* **4**, 399–402.

Dowdy, L. W., Eager, D. L., Gordon, K. D. & Saxton, L. V. (1984), 'Throughput concavity and response time convexity', *Information Processing Letters* **19**, 209–212.

Gross, D. & Harris, C. M. (1985), *Fundamentals of Queueing Theory*, second edn, Wiley.

Gross, O. (1956), A class of discrete type minimization problems, Technical Report RM-1644, RAND Corporation.

Ibaraki, T. & Katoh, N. (1988), *Resource Allocation Problems: Algorithmic Approaches*, MIT Press, Cambridge, MA.

Jagerman, D. L. (1974), 'Some properties of the Erlang loss function', *The Bell System Technical Journal* **53**(5), 525–551.

Jagers, A. A. & Doorn, E. A. V. (1986), 'On the continued Erlang loss function', *Operations Research Letters* **5**, 43–46.

Mamer, J. W. (1982), 'Cost analysis of pro rate and free-replacement warranties', *Naval Research Logistics Quarterly* **29**, 345–356.

Messerli, E. J. (1972), 'Proof of a convexity property of the Erlang B formula', *Bell System Technical Journal* **51**, 951–953.

Murthy, D. N. P. & Djamaludin, I. (2002), 'New product warranty: A literature review', *International Journal of Production Economics* **79**(2), 236–260.

Opp, M. (2003), Outsourcing Warranty Repair Services, PhD thesis, University of North Carolina at Chapel Hill.

Opp, M., Adan, I., Kulkarni, V. G. & Swaminathan, J. M. (2003), Outsourcing warranty repairs: Static alloation, Technical Report 03-01, University of North Carolina at Chapel Hill. Submitted to Management Science.

Zaporozhets, A. (1997), 'A short proof of optimality of the bottom up algorithm for discrete resource allocation problems', *Operations Research Letters* **21**(2), 81–85.

# A Closed Queueing Network Computations

Consider a two-station closed queueing network, station 1 an infinite server station with service rate $\lambda$ and station 2 is a $s$ server station with service rate $\mu$ per server. There are $N$ items circulating continuously in the network between stations 1 and 2 (assume that $N \geq s$). We calculate the expected number of customers at station 2. Let $\rho = s\mu/\lambda$. Then, the probability that all $s$ servers at station 2 are busy, and hence the probability that an incoming item is blocked, is given by

$$P_B(N) = \frac{\sum_{i=0}^{N-s} \rho^i/i!}{\sum_{i=0}^{N-s} \rho^i/i! + A(N)\rho^{N-s}/(N-s)!}, \tag{6}$$

where

$$A(N) = \sum_{i=1}^{s} \frac{\rho^i s!/(s-i)!}{s^i(N-s+i)!/(N-s)!}.$$

Note that $A(N)$ has only $s$ terms, so we need not compute it recursively. Clearly $P_B(N) = 0$ for $N < s$. From (6), we get the following expression for $P_B(N)$ in terms of $A(N)$ and $B(\rho, N)$:

$$P_B(N) = \frac{1}{1 + A(N)B(\rho, N-s)}.$$

Recall that $B(\rho, N-s)$ can be computed recursively from expression (4).

Let $W_q(N)$ by the mean waiting time in the queue at station 2 with $N$ customers circulating in the network. Similarly, define $L_q(N)$ as the expected number of customers waiting in the queue at station 2 and $L(N)$ as the expected number of customers in the queue at station 2. The following recursion holds (initialize $L_q(0) = 0$):

$$\begin{aligned} W_q(N) &= \frac{P_B(N-1) + L_q(N-1)}{s\mu}, \\ \Lambda(N) &= \frac{N}{1/\lambda + 1/\mu + W_q(N)}, \\ L_q(N) &= \Lambda(N) W_q(N), \text{ and} \\ L(N) &= L_q(N) + \Lambda(N)/\mu. \end{aligned}$$