Journal of the Korean
OR/ MS Society
Vol. 18, No. 2, August 1993

A Simulated Annealing Method for the Optimization Problem in a Multi-Server and Multi-Class Customer System

Seuck-Cheun Yoo*

Abstract

This paper addresses an optimization problem faced by a multi-server and multi-class customer system in manufacturing facilities and service industries. This paper presents a model of an integrated problem of server allocation and customer type partitioning. We approximate the problem through two types of models to make it tractable. As soution approach, the simulated annealing heuristic is constructed based on the general simulated annealing method. Computational results are presented.

1. Introduction

Minimizing work-in-process has traditionally been an important objective in manufacturing and production systems. Work-in-process is directly related to waiting time, as an elementary application of Little's Law shows. In addition, minimizing waiting time has also been an important criterion to evaluate the service quality of service industries such as banks, supermarkets and department stores. With the objective of minimizing waiting time in queue, this paper addresses an optimization problem faced by a multi-server and multi-class customer system that displays substantial queueing behavior.

Examples of this kind of queueing system are found in manufacturing or service systems in which many different customers compete for limited service capacity. In the manufacturing and production areas, a production facility manufactures a number of different products (multi-products and/or multi-models) with the same equipment and/or operators. Often, different service level requirements apply to different items, so that longer waiting time results from larger

* Department of Business Administration and Information Hong Ik University

variability of service time at a service station. Consider a flexible manufacturing system that consists of machine groups. Each group of machines has identical tools. To reduce the machine setup time that is required for switching one part type to another, the parts are grouped into families in such a way that no setup is required when processing parts within a family. Consequently, each family of parts is processed by a designated group of machines. In modern telecommunication systems, heterogeneous data types (*e.g.*, interactive messages, computer outputs, file transfers, facsimile, *etc*), compete with voice for the limited availability of shared transmission equipment. As a typical example in the service industry, consider supermarket checkout counters, where the checkout counters can be viewed as servers and arriving shoppers can be viewed as customers. Customers are categorized in terms of the number of purchased items, and the checkout counters are often divided into two major types : express and regular. The express checkout counter serves customers who purchase ten items or less. Other examples in the service domain include post offices and banks, in which each service window serves only certain types of customers.

For either the server allocation problem or the customer assignment problem, most of the models proposed in the literature have been based on the premise that the external factor is given as follows : In allocating servers the level of workload is fixed [9, 19, 22, 23, 30], and in assigning customers the service capacity is fixed [14, 29, 32]. These two problems are rarely considered jointly. They have been treated as independent problems. In particular situations where the environment of the system is likely to change frequently, the consideration of only one side is not adequate. This paper considers an integrated problem of server allocation and customer type partitioning in a multi-server and multi-class customer system. We present a queueing model of the integrated problem and develop the simulated annealing heuristic as a solution approach.

The organization of this paper is as follows. Section 2 focuses on formulating the problem with two types of approximation, the k M/G/1 model and the M/G/k approximation model. Even with these approximations, both formulations of the problem are non-convex, non-linear, and integer programming problems. As a basis for developing a solution approach, we separate the problem and obtain the dynamic programs in section 3. Section 4 describes the simulated annealing method in general and then constructs the simulated annealing heuristic approach for our problem. Computational results are presented in section 5.

2. Problem Formulation

2.1 Description of the Problem

We consider the assembly stage in a production line that has fabrication stages and then an assembly stage. The fabrication stage processes products and then sends them to the assembly stage. Producing a variety of products and models in the facility means that the production line has several different products in process at the same time [1]. Each in-process product requires different numbers of operations processed at the assembly stage that has s identical machines or operators in parallel. Types of products that arrive at the assembly stage are defined by the number of operations required at the assembly stage. For example, the product that requires x operations at the assembly stage is defined as type x. It is assumed that type 1 through z are processed at the assembly stage. Following conventional queueing terminology, the assembly stage has s identical servers in parallel and z types of arriving customers. To gain the advantages of grouping the servers and partitioning customer types, we allocate servers into m groups and partition the customer types into m groups. Each server group dedicates its service to the corresponding customer types. Therefore, s servers are allocated into m groups and z customer types are partitioned into m disjoint sets. Suppose that partitioning is non-contiguous. Then the model is not likely to be tractable, and the variance of the waiting time in a group is likely to be larger [21]. Therefore, we restrict our attention to partitioning the customer types in a contiguous way. It is assumed that m, s and z are given and that $s \leq z$.

The fabrication stations release products to the assembly stage after completion of processing. The material handling system moves them to the assembly stage. Due to the randomness of both the processing time at the fabrication stage and the availability of carriers, we assume that the arrival of customers is a Poisson process with a rate λ. There are several studies that suggest that the Markovian arrivals assumption leads to fairly robust conclusions, even when there is no underlying reason to believe that the arrivals are Poisson [21, 25]. Each arriving customer requires N operations to be processed at the assembly stage, where N is a discrete random variable with a probability density

$$P(N = x) = p_x \quad \text{for} \quad x = 1, \ldots, z.$$

Note that z is the maximum number of operations required by any customer.

The processing time of each operation i takes t_i units of time, where the t_i's are identically

and independently distributed exponential random variables with a mean $1/\mu$. Let T be the service time for customer that requires N operations at the assembly stage. Hence, the service time T can be expressed as

$$T = \sum_{i=1}^{N} t_i.$$

In this case, the first and second moments of T can be expressed as

$$E[T] = E[\sum_{i=1}^{N} t_i] = E[E[\sum_{i=1}^{N} t_i|N]] = E[N \cdot \frac{1}{\mu}] = \frac{E[N]}{\mu},$$

$$Var(T) = Var(\sum_{i=1}^{N} t_i) = E[Var(\sum_{i=1}^{N} t_i|N)] + Var(E[\sum_{i=1}^{N} t_i|N])$$

$$= E[N \cdot \frac{1}{\mu}] + Var(N \cdot \frac{1}{\mu}) = E[N] \cdot \frac{1}{\mu} + \frac{1}{\mu^2} \cdot Var(N)$$

$$= \frac{E[N] + Var(N)}{\mu^2}, \text{ and}$$

$$E[T^2] = E^2[T] + Var(T) = \frac{E[N] + E[N^2]}{\mu^2}.$$

Notice that the service time of a customer has a general probability distribution even when the processing time of an operation has an exponential distribution.

We have to determine two sets of decision variables : the server allocation and the customer type partitioning. More formally, a server allocation is denoted by

$$\vec{n} = (n_1, \ldots, n_m),$$

where n_i is the number of servers in the ith group. Notice that $\sum_{i=1}^{m} n_i = s$ and $n_i \geq 1$ for any i. Similarly, a partitioning of the customer types is denoted by

$$\vec{x} = (x_1, \ldots, x_m), \quad \text{s.t} \quad x_i < x_{i+1},$$

where x_i and $x_{i+1} - 1$ are the smallest and the largest customer types in the ith group, respectively. Thus, servers in group i process customer types x_i through $x_{i+1} - 1$. The types of customers to be processed in group i are represented as follows :

$$\{x_i, \ldots, x_{i+1} - 1\} \quad \text{for} \quad i = 1, \ldots, m.$$

For notational convenience, $x_{i+1} - 1$ is denoted by y_i. Notice that $x_1 = 1$ and $y_m = z$.

In the ith group, let $F(x_i, y_i)$ be the probability that a customer belongs to group i, where

$$F(x_i, y_i) = \sum_{k=x_i}^{y_i} p_k \qquad i=1,\ldots,m.$$

Let N_i denote the number of operations required by a customer that belongs to group i. For notational convenience, we define the following:

$$F1(x_i, y_i) = \sum_{k=x_i}^{y_i} k \cdot p_k \qquad \text{and} \qquad F2(x_i, y_i) = \sum_{k=x_i}^{y_i} k^2 \cdot p_k.$$

In other words, $F1(x_i, y_i)$ and $F2(x_i, y_i)$ are the sums of the expected values of random variables N and N^2, respectively, when N is between x_i and y_i. Let T_i be the service time of a customer in group i. It is easy to verify that

$$\begin{aligned} E[T_i] &= \frac{E[N_i]}{\mu} = \frac{F1(x_i, y_i)}{\mu \cdot F(x_i, y_i)} \\ E[T_i^2] &= \frac{E[N_i]+E[N_i^2]}{\mu^2} = \frac{F1(x_i, y_i)+F2(x_i, y_i)}{\mu^2 \cdot F(x_i, y_i)}. \end{aligned} \tag{1}$$

In summary, given m groups, s servers and z customer types, the problem is to determine the m server groups and the m customer groups so that the expected waiting time of customer in queue is minimized. Hence, the assembly stage has m different queueing systems, each of which has Poisson arrivals, a general service time distribution, and multiple servers.

2.2 Modelling

Now consider queueing models used to describe the behavior of groups of servers and customers. With the problem description, it is appropriate to model each group of the assembly stage with an M/G/k queueing system. However, there is no known closed form expression for the mean waiting time in an M/G/k system. Thus, we need to approximate the M/G/k system by using other systems or to use known approximation formulas for the mean waiting time in the M/G/k system. One possibility is to approximate the group having n_i servers by n_i M/G/1 independent systems. In this case, we can use the well known Pollaczeck-Khinchin for-

mula for computing the expected waiting time. Alternatively, we can use known results for the heuristics and approximation formulas for the mean waiting time in the M/G/k system.

The k M/G/1 Model

The approximation of k M/G/1 system can be justified by Rothkopf and Rech [21]. Rothkopf and Rech question the efficiency of the combining queue, especially queues of people, that merges separate queues into a single queue. While combining queues has been widely used, they argue that it is not necessarily effective to merge queues into a single combining queue.

The queue discipline of group i can be described as follows : the customers are assigned in cyclical order to the n_i servers (*i.e.* the first to server one, the second to server two,..., the (n_i+1)st to server one, *etc.*), with no jockeying allowed. Thus the input to the ith group is equally rationed among the n_i servers. Let λ_i denote the input rate to each server in the ith group, where

$$\lambda_i = \frac{\lambda \cdot F(x_i, y_i)}{n_i}. \tag{2}$$

By approximating the ith group as n_i M/G/1 systems, the mean queue waiting time in the i the group, denoted by W_i, is given as

$$W_i = \frac{\lambda_i}{2 \cdot (1 - \lambda_i \cdot E[T_i])} \frac{E[T_i^2]}{}. \tag{3}$$

Substituting the input rate of Eq(2) and the mean and second moments of service time of Eq (1) into Eq(2), we have

$$W_i = \frac{\lambda \cdot (F1(x_i, y_i) + F2(x_i, y_i))}{2\mu \cdot (n_i \mu - \lambda \cdot F1(x_i, y_i))}. \tag{4}$$

To guarantee that utilization of each server is less than one, the following stability condition must be satisfied :

$$\lambda \cdot F1(x_i, y_i) < n_i \cdot \mu. \tag{5}$$

The expected waiting time in the system, denoted by W, can be expressed as

$$W = \sum_{i=1}^{m} F(x_i, y_i) \cdot W_i$$

$$= \sum_{i=1}^{m} F(x_i, y_i) \cdot \frac{\lambda \cdot (F1(x_i, y_i) + F2(x_i, y_i))}{2\mu \cdot (n_i \cdot \mu - \lambda \cdot F1(x_i, y_i))} \tag{6}$$

In this case, the problem of allocating servers and partitioning customer types can be formulated in the following non-linear integer program:

Problem(P)

$$\min_{n,x,y} \quad \sum_{i=1}^{m} F(x_i, y_i) \cdot \frac{\lambda \cdot (F1(x_i, y_i) + F2(x_i, y_i))}{2\mu \cdot (n_i \cdot \mu - \lambda \cdot F1(x_i, y_i))}$$

subject to

(1) $\sum_{i=1}^{m} n_i = s$, $n_i \geq 1$, and n_i integer

(2) $x_i < x_{i+1}$, $y_i = x_{i+1} - 1$, and x_i, y_i integers

Central to our development in the k M/G/1 model has been the use of k M/G/1 queueing systems to model an M/G/k system in favor of using the exact formula for the mean waiting time.

The M/G/k Approximation Model

There are now several heuristic methods and approximations of analyzing the mean waiting time in the M/G/k system. Recently, many researchers have examined simple approximations for multi-server queueing systems. In particular, considerable attention has been given to the development of approximations for various operating characteristics of the M/G/k system. Several good approximations for the mean queue size have been developed [4, 7, 18, 27]. Approximations for the state probabilities have been studied by Hokstad [11] and Tijms *et al.* [27]. For special cases of the M/G/k system exact methods for calculating the waiting time have been studied [6], but these methods have not been practically useful. A brief review is summarized in Karmarkar *et al.* [12].

The approximation formula of Nozaki and Ross [18] can be used for the purposes of this paper. Moreover, their approximations for the mean queue size coincide with the approximation for the mean queue size given in Tijms *et al.* [27]. The Tijms' approximation has been shown to be generally superior to others based on extensive numerical comparisons (*cf.* [28]). Based on these facts, we use the Nozaki and Ross approximation formula for our model. The expected waiting time in group i, denoted by W_i, is expressed as follows:

$$W_i^q = \frac{\gamma_i^{n_i} \cdot E[T_i^2] \cdot (E[T_i])^{n_i-1}}{2 \cdot (n_i-1)! \cdot (n_i-\gamma_i \cdot E[T_i])^2 \cdot \left[\sum_{k=0}^{n_i-1} \frac{(\gamma_i \cdot E[T_i])^k}{k!} + \frac{(\gamma_i \cdot E[T_i])^{n_i}}{(n_i-1)! \cdot (n_i-\gamma_i \cdot E[T_i])}\right]} \quad (7)$$

where $\gamma_i = \lambda \cdot F(x_i, y_i)$. In other words, γ_i represents the input rate to group i. The above approximation has been shown to be quite close to the exact value when the service time (T_i) has a gamma distribution. It is also exact when the service time distribution is exponential [20].

Instead of the Pollaczeck-Khinchin formula for the M/G/1 system in the k M/G/1 model, we now use the Nozaki and Ross approximation formula in the M/G/k approximation model. The expected waiting time in queue can be expressed as

$$W = \sum_{i=1}^{m} F(x_i, y_i) \cdot W_i^q,$$

where W_i^q is given in Eq(7), $E[T_i]$ and $E[T_i^2]$ are given in Eq(1). Stability of group i is satisfied if $n_i - \gamma_i \cdot E[T_i] > 0$. This lead to the same stability condition as Eq(5).

As the case of the k M/G/1 model, the problem is also formulated as the complicated nonlinear integer program (*cf.* Problem (P)).

Problem (P′)

$$\min_{\vec{n}, \vec{x}} \sum_{i=1}^{m} F(x_i, y_i)\ W_i^q$$

subject to

(1) W_i^q in Eq(7)

(2) $E[T_i]$ and $E[T_i^2]$ in Eq(1)

(3) $\sum_{i=1}^{m} n_i = s, \quad n_i \geq 1, \quad$ and n_i integer

(4) $x_i < x_{i+1}, \quad y_i = x_{i+1} - 1, \quad$ and x_i, y_i integers

While Problem (P′) is more complicated than Problem (P), the basic structure of the problem is essentially the same.

3. Solution Approach

We approximate the problem in two ways : 1) approximate the M /G /k system by k M /G /1 systems ; 2) use the approximation formula for he mean waiting time of the M /G /k system. With the problem formulations (P) and (P′), we cannot take advantage of any classical optimization method and use it to find optimal s lutions. With above points in mind, we develop a heuristic that finds a near optimal solution wi hout explicitly finding all possible pairs of $\vec{n}$ and $\vec{x}$ in Problems (P) and (P′). Non-conve: mathematical programming problems such as Problems (P) and (P′) are likely to possess loca optimal solutions which are often far from the global optimum. Hence, this paper develops a l euristic for finding good solutions by applying the simulated annealing method which is a pro nising approach for solving complex and large combinatorial problems (*c.f.* [8]).

3.1 Separating the problem

Our problem is integrated of a server alloca ion problem and a customer type partitioning problem. The underlying concept of the solution approach is to separate the integrated problem into two independent sub-problems and then to olve one sub-problem optimally based on given values from the other sub-problem.

Dynamic Programming for Server Allocation

Given a customer type partitioning, the remai ing part of Problem (P) is a simple resource allocation problem. The problem is to allocate a of the s servers to the m groups so that the expected waiting time is minimized. It is well k nown that any deterministic resource allocation problem can be formulated as a dynamic pro ram. Let $f_i(r)$ be defined as the minimum expected waiting time obtainable from groups 1 t rough i when r servers remain to be allocated to groups i through m. We can restrict the stat space for n_i by using the following facts : (i) each group must have at least one server (ii) e ch group satisfies the stability condition in Eq (5). Let A_n denote the state space for the numb r of servers in group i.

$$A_n = \{ n \mid n=1,\ldots, r-m+i \text{ and } \lambda \cdot F1(x_i, y_i) < n \cdot \mu \},$$

where $r-m+i$ guarantees the remaining groups of having at least one server. We can write the forward dynamic programming corresponding to the resource allocation problem as follows. Find $f_{m+1}(0)$ where,

$$\begin{aligned} f_0(\cdot) &\equiv 0 \\ f_i(r) &\equiv \min_{n_i \in A_{n_i}} \{ W_i(n_i) + f_{i-1}(r+n_{i-1}) \} \\ & \quad i=1,\ldots, m+1 \\ W_i(n_i) &= F(x_i,y_i) \cdot \frac{\lambda \cdot F1(x_i, y_i) + F2(x_i,y_i))}{2\mu \cdot (n_i \quad u-\lambda \cdot F1(x_i, y_i))}. \end{aligned} \tag{8}$$

$W_i(n_i)$ represents the contribution of the ith group to the expected waiting time when n_i servers are assigned to the ith group.

Dynamic Programming for Customer Type Partitioning

Given a server allocation, the remaining part of Problem (P) is to find the shortest path from node 1 to node $z+1$, on which $m-1$ intermediate nodes are included. Given the server allocation $(\vec{n}_i)$, a location of the intermediate nodes provides the m partitions of the customer types. This shortest path problem can be formulated as a dynamic program. Let $g_i(y_i)$ be the minimum expected waiting time obtainable from groups 1 through i when group i processes customer types x_i through y_i. Since $x_{i+1} = y_i+1$, we know that the $(i+1)$st group begins with customer type y_i+1. We can restrict the state space for y_i by using the following facts : (i) each group processes at least one customer type ; (ii) each group satisfies the stability condition in Eq(5). Let A_i denote the state space of the largest customer type in each group.

$$A_i = \{ y \mid y = x_i,\ldots, z-m+i \text{ and } \lambda \cdot F1(x_i, y) < n_i \cdot \mu \},$$

where $z-m+i$ guarantees that the remaining groups have at least one customer type. We can write the dynamic program corresponding to the shortest path problem as follows. Find $g_m(z)$ where,

$$\begin{aligned} g_0(\cdot) &\equiv 0 \\ g_i(y_i) &= \min_{\substack{y_i \in A_{y_i} \\ x_i = y_{i-1}+1 \text{ and } i=1,\ldots, m}} \{ g_{i-1}(y_{i-1}) + W_i(x_i, y_i) \} \end{aligned} \tag{8}$$

$$W_i(x_i, y_i) = F(x_i, y_i) \cdot \frac{\lambda \cdot (F1(x_i, y_i) + F2(x_i, y_i))}{2\mu \ (n_i \cdot \mu - \lambda \cdot F1(x_i, y_i))}.$$

$W_i(x_i, y_i)$ represents the contribution of the ith group to the expected waiting time when the ith group processes customer types x_i through y_i.

4. The Simulated Annealing Heuristic

Recently, the simulated annealing method has been applied to large and difficult combinatorial problems and has had promising computational results. Kirkpatrick *et al.* [13] used the simulated annealing approach to solve a circuit board layout and wiring problem, as well as travelling salesman problems. Thereafter, many articles have reported successful applications of this method: the travelling salesman problem [3], the quadratic assignment problem [5, 31], the single machine scheduling problem with weighted tardiness [15], general job shop problems [16], and the operation partitioning problem [2]. These successful applications motivate us to apply the simulated annealing method to determine a near-optimal solution of our problem.

4.1 General Desciption of the Simulated Annealing Method

The simulated annealing method is a randomized local search method that has been used to approximately solve optimization problems. Simulated annealing was first developed as a simulation model to describe a physical annealing process of condensed matter [17]. Metropolis *et al.* [17] devised a simple Monte Carlo approach to simulate the behavior of a collection of atoms in achieving thermal equilibrium at a given temperature. The main feature of simulated annealing is the addition of random ascent moves in order to escape local minima which are not global minima. These ascent moves are used to supplement a normal descent algorithm. Therefore, a global minima will probably be attained after a large number of iterations.

The main procedure of the simulated annealing method consists of a loop over a random displacement generator that consequently makes changes in objective value and an adaptive mechanism. In a minimization problem, the adaptive mechanism leads to the displacement being accepted if the change is less than zero, and even in the case of a non-negative change, the displacement is accepted probabilistically.

Consider the following general optimization problem: min $f(v)$ s.t. $v \in V$, where f is a real-valued function that is defined on a finite set of elements V. To present a random displacement generator, let us associate each element $v \in V$ with a set $N(v) \in V$, where the set $N(v)$ is called the neighborhood of v. The random displacement generator can move element v to element w only when $w \in N(v)$. For each element $v, w \in V$, we assume that $w \in N(v)$ if and only if $v \in N(w)$.

To present an adaptive mechanism, let $T_k : T_k > 0$, $k = 1, \ldots$ be a monotone decreasing sequence of positive numbers that converges to zero. Given the current element v, the random displacement generator will move from v to $w \in N(v)$ with an "acceptance probability" ap, where

$$ap = e^{\frac{-\Delta f}{T_k}} \quad \text{for} \quad \Delta_f = f(w) - f(v). \tag{10}$$

This implies that each element $w \in N(v)$ has a certain probability to be chosen even when $f(w) > f(v)$. The temperature T_k is regarded as an external parameter which controls fluctuations of the random walks, *i.e.*, the flexibility to accept changes that increase the objective function value. At the beginning of the procedure, T_k is set to a reasonably high value so that the search path is able to pass local minima near the initial solution. As T_k gets smaller, the probability of accepting a displacement that causes increase in the objective value decreases.

4.2 The Simulated Annealing Algorithm

In this section, we describe how to apply the simulated annealing method to our problem. Our problem has two sets of decision variables, server allocation and customer type partitioning. For the convenience of random displacement, our simulated annealing algorithm displaces a server allocation randomly. Corresponding customer type partitioning is obtained by solving the dynamic program in Eq (9).

Initialization Procedure

It is known that the simulated annealing method performs better with a good initial solution [2, 15, 16]. Between two sets of decision variables, the initialization procedure determines a customer type partitioning. Assigning a balanced workload across the server groups, which is

proportional to the capacity of the group, is the underlying concept in the initialization procedure. It is assumed that the capacity of each group is the same. The workload in the system is defined as the expected number of operations of the customer.

$$E[N] = \sum_{1}^{z} x \cdot p_x$$

The initialization procedure is formally presented as follows.

Initialization Procedure

step 1. Set $\Delta L = \frac{E[N]}{m}$ and $y_m = z$

step 2. $y_{m-1} = \{\, y \mid F1(0,y) \le E[N]-\Delta L < F1(0,y+1) \,\}$

step 3. $y_{m-2} = \{\, y \mid F1(0,y) \le F1(0,y_{m-1}) - \Delta L < F1(0,y+1) \,\}$.

⋮ ⋮

step m. $y_1 = \{\, y \mid F1(0,y) \le F1(0,y_2) - \Delta L < F1(0,y+1) \,\}$.

Since y_i denotes the largest number of operations in the ith group, y_i+1 is equal to x_{i+1}. This scheme results in each group having almost an equal number of customer types. Let $\vec{x}_0$ denote the initial $\vec{x}$ obtained in the initialization procedure. Given $\vec{x}_0$, we generate the corresponding server allocation by solving the dynamic program in Eq (8), which will be used as an inital solution in the simulated annealing heuristic.

Annealing Schedule

The success of the simulated annealing method depends largely on the selection of the annealing schedule. As a matter of fact, the simulated annealing schedule is highly problem specific. Therefore, it is determined empirically. In this paper, we use the annealing schedule $T_k = (10)(0.9)^{k-1}$, for $k = 1,\ldots,K$ that has shown some promising results for generating good solutions for other combinatorial problems [2, 13, 31].

Pairwise Interchange

Given m groups and s servers, any server allocation is denoted as

$$\vec{n} = (n_1,\ldots,n_m) \quad \text{s.t.} \quad \sum_{i=1}^{m} n_i = s.$$

To displace server allocation randomly and to enhance the performance of the simulated annealing algorithm, a pairwise interchange is completed after the following steps. First two groups are chosen randomly, u and v. Ths sum of the two groups in n_u+n_v. Second, we enumerate all possible cases that can be allocated into two groups with n_u+n_v. There are (n_u+n_v-1) possible cases. Third, for each of the possible server allocations we solve the dynamic program for the customer type partitioning in Eq (9) and calculate its objective value. Finally, we select the server allocation that has the minimum objective value.

Stopping Rule

Regarding the system equilibrium at each temperature T_k, we run the simulation for at most E epochs. D exchanges are generated in each epoch e, where $e = 1,\ldots,E$. Let AC_e denote the objective value of the best server allocation out of those D exchanges generated in epoch e. Let AC denote the average value of all allocations generated until epoch e. For any value of T_k, our algorithm reaches its equilibrium when

$$|AC_e - AC| / AC \leq \epsilon, \quad \text{where} \quad \epsilon > 0.$$

The following parameters are used in our simulated annealing heuristic :

ϵ : the error tolerance in equilibrium;
D : the number of simulation run in each epoch;
E : the maximum number of epochs at each temperature; and
K : the number of time periods in the annealing schedule.

Our simulated annealing algorithm can be formally described as following:

step 1. (Initialization)

a. Let S be the initial solution generated by the initialization procedure, $S = (\vec{n}, \vec{x}, W)$.

b. Set $T_k = (10)(0.9)^{k-1}$ for $k=1,\ldots,K$. Set $e=1$, Set $d=1$.

step 2. a. (Pairwise Interchange) Randomly select two groups u and v. Set $sum = n_u + n_v$. Enumerate all possible allocations for two groups with sum. For each server allocation

$\vec{n}^i$, solve the dynamic programming for customer type partitioning. Next, calculate the objective value W^i, where $i=1,\ldots,sum-1$. Let $W(d)$ be the minimum objective value among $(sum-1)$ case.

$$W(d)= \min_{i=1,\ldots,sum-1} \{W^i\}.$$

b. (Epoch Simulation) If $d = D$, go to step 2 (c); otherwise, $d = d+1$ and go to step 2 (a).

c. Let $\vec{n}^*$ be the optimal allocation that has the minimum objective value $W^* = \min_{d=1,\ldots,D} \{W(d)\}$ in epoch e. Evaluate the consequent change in the objective value, $\Delta W = W^* - W$.

step 3. (Acceptance Probability) If $\Delta W \geq 0$, go to step 3 (a); otherwise, go to step 3 (b).

a. Generate a random number x that is uniformly distributed on $[0,1]$. If $x < e^{\frac{-\Delta W}{T}}$, then go to step 3 (b); otherwise go to step 4.

b. Accept the new solution as the optimal solution $S = \{\vec{n}^*, \vec{x}^*, W^*\}$, then go to step 4.

step 4. (Stability Test) Go to step 5 if $|AC - AC|/AC \leq \epsilon$. Set $e = e+1$. Go to step 5 if $e > E$; otherwise, set $d=1$ and go to step 2.

step 5. (Annealing) Set $k=k+1$. Stop if $k > K$. Otherwise, go to stop 1 (b).

A schematic description of the simulated annealing heuristic is given in Figure 4.1.

5. Computational Experience

To investigate the effectiveness of the simulated annealing heuristic, we need to experiment with the simulated annealing heuristic. Three basic approaches to the analysis of heuristics are worst case analysis, probabilistic analysis, and empirical testing[10]. The best approach would be the worst case analysis that finds a lower bound on the optimal objective value and establishes an absolute guarantee on the performance of the heuristic. Thus, it is essential to establish a tight lower bound on the optimal objective value. However, it is difficult to find a tight lower bound for even a simple problem. In this paper, we take an empirical testing approach.

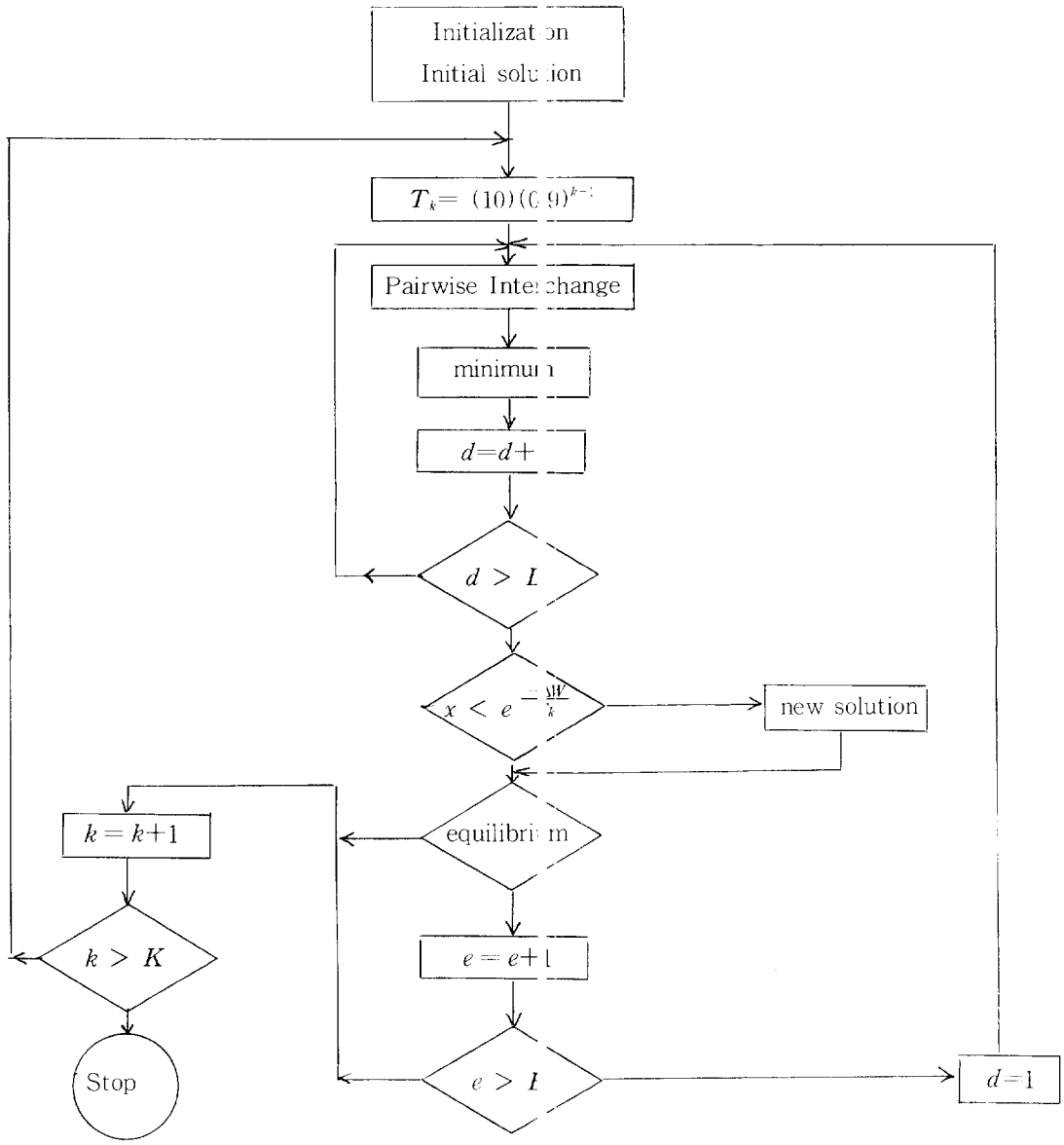

Figure 4.1 Schematic Diagram of th Simulated Annealing heuristic

5.1 Computer Experimentation

In order to evaluate the effectiveness of the sim ulated annealing heuristic, we generate a ran

dom problem instance and then solve the probl m by using the simulated annealing heuristic. We generate randomly a set of solutions for eac h random problem and find the solution having the minimum objective value. Then we compar the objective value of the solution from the simulated annealing heuristic to the minimum bjective value of solutions for the randomly generated problem (see Tang and Denardo[26]).

We generate three groups of random problems in terms of the size of the system. The size of the system is defined as the number of custome types at the assembly stage. For each system size, the program generates 50 problems randon ly. In experimenting of each random problem, the program generates 100, 200, 300 solutions r ndomly for small, medium, and large problem size, respectively. The number of random soluti ns is large enough to test the effectiveness of the heuristic. The following are the parameter necessary for generating random problems in small, medium, and large systems, respectively.

1. s, the number of servers is uniformly distrib uted over [2,10], [5,20] and [5,25].
2. m, the number of groups is uniformly distrib uted over [2,5], [2,10] an [2,15].
3. z, the number of customer types is (15 and 20), (25 and 30) and (35 and 40).
4. λ, the input rate is uniformly distributed ove r [5,10] for all sizes.
5. μ, the output rate is uniformly disributed ov r [5,15] for all sizes.

For each random problem, the computer progr m chooses s, m, λ, μ, and the probability function p_i randomly by using the above parameter ran ges depending on the problem size. Of course each problem guarantees its stability by satisfy ng the condition that the utilization factor is less than one. We have tried different sets of p rameters and found the set of parameters work well when $\epsilon = 0.001$, $D = 20$, $E = 10$, $K = 20$, and $T_k = (10)(0.9)^{k-1}$ for $k = 1, \ldots, K$. These simulation experiments have been done using a com uter code written in PASCAL, and have been implemented on an HP 9000/850 UNIX System.

5.2 Result and Discussion

First, we examine the k M/G/1 model that is based on the M/G/1 system formula. The program calculates the objective function value for each random solution and selects the minimum objective value, denoted by W_{min}. Next, he program solves the problem by using the simulated annealing heuristic and then calculate the objective value of the heuristic solution. Let W_{SA} denote the value obtained by using the simulated annealing heuristic. The performance ratio $R_1 = W_{SA}/W_{min}$ is calculated. Therefore, the euristic fares well when R_1 is far less than 1.

To evaluate the overall performance of the heur stic, the program calculates the mean value of 50 ratios in each system size.

Second, we examine the M/G/k approximati n model. Let A_{min} denote the minimum objec tive value with random solutions. Let A_{SA} d note the objective value after applying the simulated annealing heuristic. The performance atio $R_2=A_{SA}/A_{min}$ is also calculated. Because of the complexity of the M/G/k approximation f rmula, it took longer computer time to exper iment the simulated annealing heuristic for th M/G/k approximation model. Therefore, we could not experiment with 50 problems for the ase of the large problem. Table 5.1 summarizes computational results of the simulated annealing euristic.

Table 5.1 Computational Results o the Simulated Annealing Heuristic

	Smal		Medium		Large	
number of problems	50		50		50	
number of randon solutions	100		200		300	
number of customer types(z)	15	20	25	30	35	40
men of 50 ratios($R_1=W_{SA}/W_{min}$)	0.87	0.78	0.71	0.73	0.65	0.62
men of 50 ratios($R_2=A_{SA}/A_{min}$)	0.70	0.67	0.30	0.34	0.23*	0.28**

* : the average of 42 random probems

** : the average of 45 random probems

It seems that the simulated annealing heurist c is an efficient and practical method to find a near optimal server allocation and customer par tion for both models. In addition, the simulated annealing heuristic shows better performance ir the case of the M/G/k model when compared to the k M/G/1 model as shown in Table 5.1. This observation implies that when the formulaton of a combinatorial problem is very complicated, the simulated annealing method is a good solution approach.

6. Concluding Remarks

We have presented a queueing model of the integrated problem of server allocation and cus-

tomer type partitioning arising in a multi-server and multi-class customer system. We have formulated the problem with two models and have constructed a heuristics as solution approach: the simulated annealing heuristic. Motivation of applying the simulated annealing method was based on some successful applications for solving large and difficult combinatorial problems. Computational results of the heuristic for each model have been presented to show the performances of the heuristic. This paper uses the simulated annealing method as a solution approach to our problem and thus expands the applications for the simulated annealing method.

Several future research directions have been identified in this paper. First, even though our model has assumed that servers are identical it is common that each server has a different level of ability to handle the jobs. While Lin and Kumar [14] showed that the optimal policy is of the threshold type in the M/M/2 system it is an open problem to investigate the optimal policy for more general cases: for example, the case of more than two servers or the case of general service time distribution. Second, our model has given equal weight to the expected waiting time of each customer type. A potential extension of the research is to consider a weight for each customer type according to the system objective. Then the relationships between system configuration and systme performance as give in this paper may be incorporated to optimize the system performance. We hope that this research provides some insight to these questions as well as a basis for future investigation of the problem.

Reference

[1] Ahmadi,R.Z. and H. Matsuo, "The Line Segmentation Problem," Working Paper, Anderson Graduate School of Management, UCLA, 1988.

[2] _____ and C. S. Tang, "An Operation Partitioning Problem for Automated Assembly System Design," Working Paper, Anderson Graduate School of Management, UCLA, 1989.

[3] Bonomi, E. and J. Lutton, "The *N*-city Travelling Salesman Problem: Statistical Mechanics and the Metropolis Algorithm," *SIAM Review*, Vol. 26 (1984), pp.551-568.

[4] Boxma, O.J.,J.W.Cohen and N.Huffels, "Approximations of the Mean Waiting Time in an M/G/c Queueing System," *Operations Research*, Vol. 27 (1980),pp.1115-1127.

[5] Burkard, R. E. and F. Rendl, "A thermodynamically Motivated Simulation Procedure for Combinatorial Optimization Problems," *European Journal of Operational Research*, Vol. 17 (1984), pp. 169-174.

[6] Cohen, J.W., "On the M/G/2 Queueing Model," *Stochastic Process. Appl.*, Vol. 12(1982), pp.231-248.

[7] Cosmetatos, G.P., "Some Approximate Equi ibrium Results for the Multiserver Queue," *Oper. Res. Quart.*, Vol. 27 (1976), pp.615-620

[8] Committee on the Next Decade in Operations Research (CONDOR), "Operations Research : The Next Decade," *Operations Research*, Vol. 36 (1988), pp.619-637.

[9] Dyer, M. E. and L. G. Proll, "On the Validi y of Marginal Analysis for allocating Servers in M/M/*c* Queues," *Mannagement Science*, Vol. 23(1977), pp.1019-1022.

[10] Fisher, M.L. and H. G. Rinnooy Kan, "The Design, Analysis and Implementation of Heuristics," *Mannagement Science*, Vol. 34(1988), pp.263-265

[11] Hokstad, P., "Approximations for the M/G/*m* Queue," *Operations Research*, Vol. 26 (1978), pp.551-523

[12] Karmarkar, U.S.,S.Kekre and S.Kekre, "Lot Sizing in Multi-item Multi-machine Job Shops," *IIE Trans.*, Vol. 17(1985), pp.290-298.

[13] Kirkpatrick, S., C. D. Gelatt, Jr. and M. P. Vecchi, "Optimization by Simulated Annealing," *Science*, Vol. 220(1983), pp.671-680

[14] Lin, W. and P.R. Kumar, "Optimal Control of a Queueing System with Two Heterogeneous Servers," *IEEE Trans. on Automatic Control*, Vol. AC-29(1984), pp.694-703

[15] Matsuo, H., C.Suh and R. S. Sullivan, "A Controlled Search Simulated Annealing Method for the Single Machine Weighted Tardiness Problem," Working Paper, Graduate School of Business, University of Taxas at Austin, 1987

[16] _____, _____ and _____, "A Controlled Search Simulated Annealing Method for the General Jobshop Scheduling Problem," Working Paper, Graduate School of Business, University of Taxas at Austin, 1988.

[17] Methopolis, N., A. Rosenbluth, M. Rosenbluth, A. Teller and E. Teller, "Equation of State Calculations by Fast Computing Machines," *J. Chem. Phys.*, Vol. 21(1953), pp.1087-1092

[18] Nozaki, S. A. and S. M. Ross, "Approximations in Finite Capacity Multiserver Queues with Poisson Arrivals," *Journal of Applied Probability*, Vol. 15(1978), pp.826-834.

[19] Rolfe, A. J, "A Note on Marginal Allocation in Multiple Service Systems," *Management Science*, Vol. 17(1971), pp.656-658.

[20] Ross, S. M, *Introduction to probability Models 2nd Ed.*, Academic Press Inc. N.Y.,1980.

[21] Rothkopf, M. H. and P. Rech "Perspectives On Queues : Combining Queues is Not Always Benefical," *Operations Research*, Vol. 35(1987), pp.906-909.

[22] Shanthikumar, G. J. and D. D. Yao, "Optimal Server Allocation A System of Multi-Server Stations," *Management Science*, Vol. 33(1987), pp.1173-1180.

[23] _____ and _____, "On Server Allocation in Multiple Center Manufacturing Systems," *Operations Research*, Vol. 36(1988), pp.333-342

[24] Solberg, J.J, "A Mathematical Model of C mputerized Manufacturing Systems," *Proc. 4th International Conference on Production R search*, Tokyo, Japan, 1977.

[25] Suri, R, "New Techniques for Modelling ar d Control of Flexible Automated Manufacturing Systems," *Proc. IFAC 1981*, Kyoto, Japan, 1981.

[26] Tang, C. S. and E. V. Denardo, "Models rising from a Flexible Manufacturing Machine, Part 1: Minimization of the Number of Toc Switches," *Operations Research* Vol. 36(1988), pp. 767-777.

[27] Tijms, H. C., M. H. van Hoorn and A. Fe ergruen, "Approximations for the Steady-State Probabilities in the M/G/c Queue," *Adv. in Appl. Probab.*, Vol. 13(1981), pp.186-206.

[28] van Hoorn, M. H. and H. C. Tijms, "Appr oximations for the Waiting Time Distribution of the M/G/c Queue," *Performance Evaluat on*, Vol. 2(1982), pp.22-28.

[29] Weber, R. R, "On the Optimal Assignmen of Customers to Parallel Servers," *Journal of Applied Probability*, Vol. 15(1978), pp.406-413.

[30] _____, "On the Marginal Benefit of Addi ng Servers to G/GI/*m* Queues," *Management Science*, Vol. 26(1980), pp.946-951.

[31] Wilhelm, M. R. and T. L. Ward, "Solving Quadratic Assignment Problems by 'Simulated Annealing'," *IIE Transactions*, (1987) pp.1 7-117.

[32] Winston, W. L. "Assignment of Customer to Servers in a Heterogeneous Queueing System with Switching," *Operations Research*, Vol. 25(1977), pp.468-483.