# IMPROVED ALGORITHMS FOR MACHINE ALLOCATION IN MANUFACTURING SYSTEMS

**HANS FRENK**
*Erasmus University, Rotterdam, The Netherlands*

**MARTINE LABBÉ**
*University of Brussels, Brussels, Belgium*

**MARIO VAN VLIET**
*Erasmus University, Rotterdam, The Netherlands*

**SHUZHONG ZHANG**
*Erasmus University, Rotterdam, The Netherlands*

(Received June 1991; revisions received June, October 1992; accepted November 1992)

In this paper we present two algorithms for a machine allocation problem occurring in manufacturing systems. For the two algorithms presented we prove worst-case performance ratios of 2 and 3/2, respectively. The machine allocation problem we consider is a general convex resource allocation problem, which makes the algorithms applicable to a variety of resource allocation problems. Numerical results are presented for two real-life manufacturing systems.

An important design problem in manufacturing concerns the optimal allocation of machines (servers) to workstations within a manufacturing system. One can think, for example, of the problem of allocating a fixed number of machines among different workstations such that the performance of the system (e.g., in terms of work-in-process or lead times of products) is optimal. Another problem concerns the minimum cost allocation of machines such that the performance of the system meets a certain target. The latter problem is the subject of this paper.

The production process we consider can be modeled as an open network of queues with different product classes. This means that the production process consists of workstations through which each product follows its own deterministic route. A workstation consists of several parallel identical machines (servers). In the sequel we assume that product routings, the amount of traffic offered, the location of machines, and technology (e.g., processing times) are specified.

Several authors have considered server allocation problems in manufacturing queueing networks. For closed queueing networks, which are a popular means to model a flexible manufacturing system (FMS), Vinod and Solberg (1985), Dallery and Frein (1986), Shanthikumar and Yao (1987, 1988), and Dallery and Stecke (1990) considered various server allocation problems. The server allocation issues addressed in these papers differ according to the optimization problem treated, the kind of manufacturing system analyzed, and the queueing network used for modeling. Boxma et al. (1990) and Van Vliet and Rinnooy Kan (1991) treated server allocation problems in open queueing networks in which each workstation is modeled as an $M/M/m$ and a $GI/G/m$ queue, respectively. Boxma et al. present the so-called *server allocation* problem. For this problem they propose a greedy algorithm which generates undominated solutions. Furthermore, their algorithm provides bounds to check how close the heuristic solution is to the optimal one.

In this paper, we study the same problem setting. The algorithms we present are improved versions of the greedy algorithm presented in Boxma et al. Whereas Boxma et al. do not give any worst-case analysis of their algorithm, we prove our algorithms to have worst-case ratio performances of 2 and 3/2,

*Subject classifications:* Manufacturing: allocation of machines in an FMS. Manufacturing, performance/productivity: optimization of steady-state performance. Queues, networks: optimization of queueing networks.
*Area of review:* OPTIMIZATION.

respectively. Although the problem is treated here in the context of server allocation, it represents a general class of resource allocation problems. Therefore, the algorithms are applicable to a wider class of problems.

In Section 1 we describe the underlying queueing network and the server allocation problem treated. In Section 2 we discuss the algorithms and their theoretical analysis. Numerical results of the algorithms applied to two real-life manufacturing systems are presented in Section 3. Conclusions and suggestions for further research are stated in Section 4.

## 1. SYSTEM ANALYSIS

The manufacturing system we consider consists of $J$ workstations. Each workstation $j$ has $m_j$ identical parallel servers with independent exponentially distributed service times with mean $1/\mu_j$; $N$ product types are produced by the system. Products of type $i$ arrive at the first workstation they visit according to a Poisson process with parameter $\lambda^i$, and then follow a deterministic route through a subset of workstations. A product may visit a workstation more than once, but for simplicity we will not allow two successive stages of a product route to be identical. Furthermore, we assume that the arrival and service processes are independent. The assumptions that service times are independently and exponentially distributed, and the arrival process is Poisson, allow us to make an exact analysis of the steady-state behavior of the queueing network. Since the focus in this paper is on combinatorial optimization problems, rather than on the queueing network analysis, we prefer to model the manufacturing system as a queueing network that can be analyzed exactly. However, for some practical environments the exponential and Poisson assumptions might not be valid. Van Vliet (1991) shows how the algorithms presented in this paper can be, under mild additional assumptions, applied to general queueing networks for which the exponential and Poisson assumptions do not hold.

For further analysis we can treat the different product types as one aggregate product with an aggregate arrival rate $\lambda_j$ at each workstation $j$. The joint equilibrium queue length distribution in the system has a product form (cf. Kelly 1979, corollary 3.4). Each workstation $j$ behaves as an $M/M/m_j$ queue in the steady state. This leads to the following well known formula for the average number of products present (in queue and in process) at workstation $j$ (cf. Tijms 1986, p. 332).

$$L_j(m_j, \mu_j, \lambda_j)$$
$$= \frac{\left(\frac{\lambda_j}{\mu_j}\right)^{m_j}\left(\frac{1}{\mu_j m_j}\right)}{m_j!\left(1-\frac{\lambda_j}{\mu_j m_j}\right)^2} \cdot \left\{\sum_{k=0}^{m_j-1}\frac{\left(\frac{\lambda_j}{\mu_j}\right)^k}{k!}+\frac{\left(\frac{\lambda_j}{\mu_j}\right)^{m_j}}{m_j!\left(1-\frac{\lambda_j}{\mu_j m_j}\right)}\right\}^{-1}+\frac{\lambda_j}{\mu_j}. \tag{1}$$

In the sequel we assume that the arrival and service rates are given, while the number of servers at each workstation are the decision variables. This means that $L_j(m_j, \mu_j, \lambda_j)$ can be regarded as a function of $m_j$ only: $L_j(m_j)$. Dyer and Proll (1977) proved that $L_j(m_j)$ as given by (1) is a convex decreasing function in $m_j$.

We will measure the steady-state performance of the system by the work-in-process (WIP) of the system. The WIP (inventory) is the total value of all the products that are in the system. Without loss of generality, we make the assumption that the value of a product at workstation $j$, either in queue or in process, is independent of the type of product and equal to $v_j$. In other words, $v_j$ represents the value of inventory per unit at workstation $j$. For example, products waiting at the end of the production process, i.e., the last workstation they visit, have more added value in terms of material and manpower than the products waiting at the beginning of the production process. The formulation for WIP then becomes

$$WIP(m_1, \dots, m_J) = \sum_{j=1}^{J} v_j L_j(m_j).$$

Furthermore, we assume that the allocation of $m_j$ servers at workstation $j$ generates investment costs of $F_j(m_j)$, with $F_j(m_j)$ a convex and nondecreasing function in $m_j$. To prevent the system from becoming instable, we have to require the *traffic intensity* at a workstation $j(\equiv \rho_j = \lambda_j/\mu_j m_j)$ to be less than one. It is easy to verify that this results in requiring that $m_j \geqslant m_j^L = \lfloor \lambda_j/\mu_j \rfloor + 1$, where $\lfloor\cdot\rfloor$ represents the integer round-down operation.

For convenience we use the following notation.

$$m = (m_1, \ldots, m_J);$$
$$S = \{m | m_j \geq m_j^L\};$$
$$F(m) = \sum_{j=1}^{J} F_j(m_j);$$
$$L(m) = \sum_{j=1}^{J} v_j L_j(m_j);$$
$$\Delta F_j(m_j) = F_j(m_j) - F_j(m_j - 1) \qquad (m_j > m_j^L);$$
$$\Delta L_j(m_j) = v_j(L_j(m_j - 1) - L_j(m_j)) \qquad (m_j > m_j^L).$$

From the convexity of $F_j$ and $L_j(j \in \{1, \ldots, J\})$ it follows that

$$\frac{\Delta F_j(m_j + 1)}{\Delta L_j(m_j + 1)} \geq \frac{\Delta F_j(m_j)}{\Delta L_j(m_j)}, \text{ and} \tag{2}$$
$$\Delta L_j(m_j + 1) \leq \Delta L_j(m_j). \tag{3}$$

The optimization problem we consider is to allocate servers to workstations in such a way that the WIP is below a target WIP level $W_T$. The configuration we are looking for is a minimum cost configuration. The mathematical formulation is as follows.

### Problem SA

Minimize$_m$ $F(m)$

subject to $L(m) \leq W_T$

$m_j \geq m_j^L$, $m_j$ integer $(j \in \{1, \ldots, J\})$.

## 2. ALGORITHMS AND THEIR ANALYSIS

Since problem **SA** can be regarded as a generalization of the knapsack problem it is NP-hard. Therefore, our focus will be on algorithms to find approximately optimal solutions. We will discuss two such algorithms. However, to make this paper self-contained we first briefly mention the results by Boxma et al., because the algorithms and results presented in their study serve as a starting point for our analysis.

The algorithm by Boxma et al. to approximately solve **SA** starts with the smallest possible allocation, that is, $m_j^L$ for each workstation $j$. At every iteration it then adds a server at that workstation where the quotient of the increase of the objective function and the decrease of WIP is the smallest. The algorithm terminates as soon as adding a server makes the allocation feasible.

### Algorithm SA1

*STEP 1.* Start with $c^0$ where $c_j^0 = m_j^L$.

*STEP 2.* $k := 1$.

*STEP 3.* Set $c^k := c^{k-1} + e_i$, where $e_i$ is the $i$th unit vector, and

$$i = \text{Arg} \min_{j \in \{1, \ldots, J\}} \frac{\Delta F_j(c_j^{k-1} + 1)}{\Delta L_j(c_j^{k-1} + 1)}.$$

*STEP 4.* If $L(c^k) \leq W_T$, $c^{SA1} := c^k$, stop; else $k := k + 1$, go to Step 3.

**Definition 1.** An allocation $x$ is called *undominated* (efficient) (cf. Fox 1966) if for all $y \in S$:

$$F(y) < F(x) \Rightarrow L(y) > L(x)$$
$$F(y) = F(x) \Rightarrow L(y) \geq L(x).$$

Boxma et al. prove the following results on algorithm **SA1**.

**Lemma 1.** *Allocations generated by algorithm* **SA1** *are undominated.*

**Lemma 2.** *If* $c^0, \ldots, c^p$ *are the allocations generated by algorithm* **SA1** *and* $c^*$ *is an optimal allocation for* **SA**, *then it holds that* $F(c^{p-1}) < F(c^*) \leq F(c^p)$.

Concerning the complexity of **SA1**, the following can be shown. Let the maximum number of servers among all undominated allocations be $m$, then the total number of operations needed by **SA1** is $\mathcal{O}(mJ)$.

Lemma 2 shows that the solution generated by **SA1** provides bounds to check whether the allocation found by **SA1** is sufficiently close to the optimal allocation. The difference (if any) between the heuristic and the optimal solution is created by the server which is added to a workstation ($j'$, say) in the final step of the algorithm. If this 'final server' causes the 'final WIP' to be substantially larger than $W_T$, the heuristic solution might be far from the optimal one. If, however, the final WIP is very close to $W_T$, chances are high that the heuristic solution cannot be much improved upon.

The following algorithm tries to improve upon the allocation generated by algorithm **SA1** by making use of the above observation. Algorithm **SA2** generates $J$ allocations. The first allocation is as given by **SA1**. To get the second allocation, **SA2** takes the number of servers at each workstation equal to those as given in the first allocation, except for workstation $j'$, where the number of servers is decreased by one. Note that this allocation is not feasible. Given this allocation, servers are added in the same greedy manner as in **SA1**. The procedure stops as soon as a feasible allocation is found. The resulting allocation is the second of the $J$ allocations. The number of servers at $j'$ is now kept fixed in all the following steps of the algorithm. The third allocation is found by the same procedure, with $j'$ now equal to the workstation at which the last server was added to get the previous allocation. This procedure is repeated until $J$ allocations have been generated. The allocation with the lowest objective function value is the heuristic allocation as given by **SA2**. This procedure can be stated as follows.

## Algorithm SA2

```
(Initialization) Set $c^{H_1} := c^{SA1}$; $C := \{c^{H_1}\}$; $A := \{1, \ldots, J\}$; $k := 1$.
STEP 1. Let $j_k$ be the index of the workstation at which a server is added in the final iteration to obtain heuristic allocation $H_k$.
    $A := A \setminus \{j_k\}$;
    $c^{H_{k+1}} = (c_1^{H_k}, \ldots, c_{j_k}^{H_k} - 1, \ldots, c_J^{H_k})$; $k := k + 1$.
STEP 2. Set $c^{H_k} := c^{H_k} + e_i$, where $e_i$ is the $i$th unit vector and
    $i = \text{Arg}\min_{j \in A} \frac{\Delta F_j(c_j^{H_k}+1)}{\Delta L_j(c_j^{H_k}+1)}$.
STEP 3. If $L(c^{H_k}) \leqslant W_T$
    then $C := C \cup \{c^{H_k}\}$. If $k = J$ go to Step 4, else go to Step 1.
    else go to Step 2.
STEP 4. Choose $c^{SA2} := \text{Arg}\min_{c \in C} F(c)$.
```

Before presenting the worst-case analysis for **SA2**, note that the complexity of **SA2** is $\mathcal{O}(mJ^2)$ (cf. the complexity of algorithm **SA1** presented before).

We can now prove the following theorem.

**Theorem 1.** *Let $c^*$ be the optimal allocation for* **SA**, *then it holds that*

$$\frac{F(c^{SA2}) - F(m^L)}{F(c^*) - F(m^L)} \leqslant 2$$

*and this bound is tight.*

**Proof.** We relabel the indices such that $i$ equals the index $j_i$ of the last workstation at which a server is added to obtain the heuristic solution $c^{H_i}$, $i \in \{1, \ldots, J\}$. Note that $c_i^{H_J} = c_i^{H_i} - 1$ for $i = 1, \ldots, J - 1$. Furthermore, $(c_1^{H_J}, \ldots, c_{J-1}^{H_J}, c_J^{H_J} - 1)$ is infeasible with regard to **SA**. Hence, there exists at least one index $j$ for which $c_j^* \geqslant c_j^{H_j}$. Let $d$ be the smallest index for which this is satisfied, i.e., $c_d^* \geqslant c_d^{H_d}$ and $c_j^* < c_j^{H_j}$ for $j = 1, \ldots, d - 1$.

In the proof we use the two relations:

$$\sum_{j=1}^{J} \sum_{i=m_j^L+1}^{m_j} \Delta F_j(i) = \sum_{j=1}^{J} F_j(m_j) - \sum_{j=1}^{J} F_j(m_j^L) = F(m) - F(m^L)$$

$$\sum_{j=1}^{J} \sum_{i=m_j^L+1}^{m_j} \Delta L_j(i) = \sum_{j=1}^{J} \nu_j L_j(m_j^L) - \sum_{j=1}^{J} \nu_j L_j(m_j) = L^L - L(m).$$

When applying **SA2**, servers are successively added to the workstations in a nondecreasing order of the ratio $\Delta F_j(m_j)/\Delta L_j(m_j)$. Hence, (2) and the fact that $d$ is the index of the last workstation at which a server is added to obtain $c^{H_d}$ imply that

$$\frac{\Delta F_j(i)}{\Delta L_j(i)} \leqslant \frac{\Delta F_d(c_d^{H_d})}{\Delta L_d(c_d^{H_d})} \quad \text{for } j = 1, \ldots, J \text{ and } i = 1, \ldots, c_j^{H_d} \tag{4}$$

$$\frac{\Delta F_j(i)}{\Delta L_j(i)} \geqslant \frac{\Delta F_d(c_d^{H_d})}{\Delta L_d(c_d^{H_d})} \quad \text{for } j = d, \ldots, J \text{ and } i > c_j^{H_d}. \tag{5}$$

We know that

$$\begin{aligned} F(c^{SA2}) - F(m^L) &\leqslant F(c^{H_d}) - F(m^L) \\ &= F(c^*) - F(m^L) \\ &\quad + \sum_{j: c_j^{H_d} > c_j^*} \sum_{i=c_j^*+1}^{c_j^{H_d}} \Delta F_j(i) \\ &\quad - \sum_{\substack{j: c_j^{H_d} < c_j^* \\ j \neq d}} \sum_{i=c_j^{H_d}+1}^{c_j^*} \Delta F_j(i) \\ &\quad - \sum_{i=c_d^{H_d}+1}^{c_d^*} \Delta F_d(i). \end{aligned} \tag{6}$$

Furthermore, (4) implies that

$$\sum_{j: c_j^{H_d} > c_j^*} \sum_{i=c_j^*+1}^{c_j^{H_d}} \Delta F_j(i) \leqslant \frac{\Delta F_d(c_d^{H_d})}{\Delta L_d(c_d^{H_d})} \sum_{j: c_j^{H_d} > c_j^*} \sum_{i=c_j^*+1}^{c_j^{H_d}} \Delta L_j(i) \tag{7}$$

Now, since $(c_1^{H_d}, \ldots, c_{d-1}^{H_d}, c_d^{H_d} - 1, c_{d+1}^{H_d}, \ldots, c_J^{H_d})$ is not feasible with regard to **SA** we get

$$\begin{aligned} L(c_1^{H_d}, \ldots, c_{d-1}^{H_d}, c_d^{H_d} - 1, c_{d+1}^{H_d}, \ldots, c_J^{H_d}) &= L(m^L) - \sum_{j: c_j^{H_d} > c_j^*} \sum_{i=m_j^L+1}^{c_j^*} \Delta L_j(i) \\ &\quad - \sum_{j: c_j^{H_d} > c_j^*} \sum_{i=c_j^*+1}^{c_j^{H_d}} \Delta L_j(i) \\ &\quad - \sum_{\substack{j: c_j^{H_d} \leqslant c_j^* \\ j \neq d}} \sum_{i=m_j^L+1}^{c_j^{H_d}} \Delta L_j(i) \\ &\quad - \sum_{i=m_j^L+1}^{c_d^{H_d}-1} \Delta L_d(i) > W_T. \end{aligned} \tag{8}$$

Moreover, since the optimal solution $c^*$ is feasible, we obtain

$$\begin{aligned} L(c^*) = L(m^L) &- \sum_{j:c_j^{H_d} > c_j^*} \sum_{i=m_j^L+1}^{c_j^*} \Delta L_j(i) \\ &- \sum_{\substack{j:c_j^{H_d} \le c_j^* \\ j \ne d}} \sum_{i=m_j^L+1}^{c_j^{H_d}} \Delta L_j(i) \\ &- \sum_{\substack{j:c_j^{H_d} < c_j^* \\ j \ne d}} \sum_{i=c_j^{H_d}+1}^{c_j^{H_d}} \Delta L_j(i) \\ &- \sum_{i=m_j^L+1}^{c_d^{H_d-1}} \Delta L_d(i) \\ &- \sum_{i=c_d^{H_d}}^{c_d^*} \Delta L_d(i) \le W_T. \qquad (9) \end{aligned}$$

Using (8) and (9) in (7) we obtain

$$\begin{aligned} &\sum_{j:c_j^{H_d} > c_j^*} \sum_{i=c_j^*+1}^{c_j^{H_d}} \Delta F_j(i) \\ &\le \frac{\Delta F_d(c_d^{H_d})}{\Delta L_d(c_d^{H_d})} \cdot \left[ \sum_{\substack{j:c_j^{H_d} < c_j^* \\ j \ne d}} \sum_{i=c_j^{H_d}+1}^{c_j^*} \Delta L_j(i) + \sum_{i=c_d^{H_d}}^{c_d^*} \Delta L_d(i) \right] \\ &\le \sum_{\substack{j:c_j^{H_d} < c_j^* \\ j \ne d}} \sum_{i=c_j^{H_d}+1}^{c_j^*} \Delta F_j(i) + \sum_{i=c_d^{H_d}}^{c_d^*} \Delta F_d(i), \qquad (10) \end{aligned}$$

where the second inequality follows from the definition of $d$ which implies that (5) holds for $d$ and for all $j$ such that $c_j^{H_d} > c_j^*$.

Finally, substituting (10) in (6) yields

$$\begin{aligned} F(c^{SA2}) - F(m^L) &\le F(c^*) - F(m^L) \qquad (11) \\ &+ \Delta F_d(c_d^{H_d}) \le 2(F(c^*) - F(m^L)). \end{aligned}$$

To prove that the above bound is tight we use a similar example as in Csirik et al. (1990), where the same bound was proven for a similar algorithm for the 0–1 min-knapsack problem.

Take the problem instance:

$$\begin{aligned} J &= 3 \\ L(m^L) - W_T &= C + 1 \\ F_j(m_j^L) &= 0 \ (j = 1, 2, 3) \\ \Delta L_1(m_1^L + 1) &= \Delta F_1(m_1^L + 1) = 1 \\ \Delta L_2(m_2^L + 1) &= C \\ \Delta F_2(m_2^L + 1) &= C + \epsilon (\epsilon > 0) \\ \Delta L_3(m_3^L + 1) &= \Delta F_3(m_3^L + 1) = C - 1. \end{aligned}$$

It is easy to verify that $c^{SA2} = (m_1^L + 1, m_2^L + 1, m_3^L + 1)$ and $c^* = (m_1^L + 1, m_2^L + 1, m_3^L)$ with $F(c^{SA2}) = 2C + \epsilon$ and $F(c^*) = C + 1 + \epsilon$. This implies that

$$\begin{aligned} \frac{F(c^{SA2}) - F(m^L)}{F(c^*) - F(m^L)} &= \frac{2C + \epsilon}{C + 1 + \epsilon} \\ &= \frac{2(C + 1 + \epsilon)}{C + 1 + \epsilon} - \frac{1 + \epsilon}{C + 1 + \epsilon} \\ &= 2 - \frac{1 + \epsilon}{C + 1 + \epsilon} \end{aligned}$$

which can be arbitrarily close to 2 for large $C$ and small $\epsilon$.

From Theorem 1 it follows that **SA2** has a worst-case performance ratio of 2. To improve upon this performance ratio, we introduce the following algorithm, which is again an improvement algorithm that uses allocations from the set of possible allocations, generated by algorithm **SA2**, as initial allocations. For each of the $J$ allocations $c^{H_k}$ $(k \in \{1, \ldots, J\})$, as generated by **SA2**, **SA3** creates one new problem **SA**$_k$ by introducing the additional restriction that the workstation $k$ (which is the last workstation to which a server has been added to obtain $c^{H_k}$) has at least $c_k^{H_k}$ servers. Then, **SA2** is applied to each such new problem **SA**$_k$, $(k \in \{1, \ldots, J\})$ to generate new feasible solutions. The heuristic solution returned by **SA3** is the allocation generated by **SA2** when applied to **SA** or **SA**$_k$ $(k \in \{1, \ldots, J\})$, which minimizes the objective function.

## Algorithm SA3

*STEP 1.* Apply **SA2** and denote the set of possible allocations by $C = (c^{H_1}, \ldots, c^{H_J})$.

*STEP 2.* For $k := 1$ to $J$ construct the following problem.

## Problem SA$_k$

$$\underset{m_j}{\text{Minimize}} \sum_{j=1}^{J} F_j(m_j)$$

subject to

$$\sum_{j=1}^{J} v_j L_j(m_j) \le W_T$$

$$\begin{aligned} m_j &\ge m_j^L, \quad m_j \text{ integer } (j \in \{1, \ldots, J\} \backslash \{k\}) \\ m_k &\ge c_k^{H_k}, \quad m_k \text{ integer}. \end{aligned}$$

Apply algorithm **SA2** to problem **SA**$_k$ and denote the heuristic solutions by $\bar{c}^{H_k} = (\bar{c}_1^{H_k}, \ldots, \bar{c}_J^{H_k})$.

*STEP 3.* Let $\bar{C} = (\bar{c}^{H_1}, \ldots, \bar{c}^{H_J})$. Choose

$$c^{SA3} = \text{Arg} \min_{c \in C \cup \bar{C}} F(c).$$

Note that the complexity of **SA3** is $\mathcal{O}(mJ^3)$ (cf. the complexities of **SA1** and **SA2** discussed before).

**Theorem 2.** *Let $c^*$ be the optimal allocation for* **SA**, *then it holds that*

$$\frac{F(c^{SA3}) - F(m^L)}{F(c^*) - F(m^L)} \leq 3/2 .$$

**Proof.** We use the same notation as in the proof of Theorem 1. So, let $d$ be the index such that $c_d^* \geq c_d^{H_d}$ and $c_j^* < c_j^{H_j}$ for $j = 1, \dots, d - 1$.

We now distinguish between two cases.

**Case a**

$$\Delta F_d(c_d^{H_d}) \leq 1/2\,(F(c^*) - F(m^L)).$$

In this case it follows directly from (11) that

$$F(c^{SA2}) - F(m^L) \leq 3/2\,(F(c^*) - F(m^L)),$$

and the result follows from $F(c^{SA3}) \leq F(c^{SA2})$.

**Case b**

$$\Delta F_d(c_d^{H_d}) > 1/2\,(F(c^*) - F(m^L)).$$

Let $\bar{c}^*$ denote the optimal allocation of problem $\mathbf{SA}_d$. We then have

$$\begin{aligned}
&F(c^{SA3}) - F(m^L)\\
&\quad \leq F(\bar{c}^{H_d}) - F(m^L)\\
&\quad = F(\bar{c}^{H_d}) - \sum_{\substack{j=1\\ j\neq d}}^{J} F_j(m_j^L) - F_d(c_d^{H_d})\\
&\qquad + F_d(c_d^{H_d}) - F_d(m_d^L)\\
&\quad \leq 2\Bigg(F(\bar{c}^*) - \sum_{\substack{j=1\\ j\neq d}}^{J} F_j(m_j^L) - F_d(c_d^{H_d})\Bigg)\\
&\qquad + F_d(c_d^{H_d}) \;- F_d(m_d^L)\\
&\quad \leq 2\Bigg(F(c^*) - \sum_{\substack{j=1\\ j\neq d}}^{J} F_j(m_j^L) - F_d(c_d^{H_d})\Bigg)\\
&\qquad + F_d(c_d^{H_d}) \;- F_d(m_d^L)\\
&\quad = 2(F(c^*) - F(m^L))\\
&\qquad - (F_d(c_d^{H_d}) - F_d(m_d^L))\\
&\quad \leq 2(F(c^*) - F(m^L)) - \Delta F_d(c_d^{H_d})\\
&\quad \leq 3/2\,(F(c^*) - F(m^L)).
\end{aligned}$$

The second inequality is obtained by applying Theorem 1 to problem $\mathbf{SA}_d$ and the third inequality follows from the fact that $c^*$ is feasible to $\mathbf{SA}_d$ given the definition of $\mathbf{SA}_d$.

Note that the situation in which $\Delta F_d(c_d^{H_d}) > 1/2(F(c^*) - F(m^L))$ is only likely to occur when the number of workstations is small. In most practical environments the number of workstations is fairly large ($J \geq 4$), which may well imply that $\Delta F_d(c_d^{H_d}) \leq 1/2(F(c^*) - F(m^L))$ for all $d \in J$. In this case, it follows from the above that **SA2** also has a worst-case performance ratio of 3/2. In Section 3 we show that the numerical results obtained for the practical settings we considered are similar for **SA2** and **SA3**. The above observation, together with the numerical results, strongly suggests that the average performance of **SA2** and **SA3** will be very similar.

## 3. NUMERICAL RESULTS

We applied the algorithms to two manufacturing systems which were taken from Van Vliet and Rinnooy Kan (1991). The first system is a manufacturing system producing semiconductor devices and consists of 13 workstations. Up to 10 different semiconductor devices (product types) are produced. The second system consists of 11 workstations and produces 2 product types.

The two manufacturing systems were modeled in Van Vliet and Rinnooy Kan as queueing networks with independent $GI/G/m$ queues. Since we focus on the nonlinear optimization problems rather than on the queueing network aspects, we model each workstation as an $M/M/m$ queue. This makes the queueing network analysis exact (see Section 1). For how to apply the above optimization problems to (more realistic) non-Markovian queueing networks we refer to Van Vliet and Rinnooy Kan.

To compare the performance of heuristics **SA1**, **SA2**, and **SA3** we use a relative error indicator. For each heuristic **SA*i*** we use the upperbound ($UB_i$) and the lowerbound ($LB_i$) as given by the algorithms. The relative error is then calculated by:

$$\text{Relative error of } \mathbf{SA}i = \frac{UB_i - LB_i}{\dfrac{UB_i + LB_i}{2}}. \tag{12}$$

We performed the heuristics for a wide range of WIP values. It appeared that the results for **SA2** and **SA3** did not differ for the two manufacturing systems we examined. This means that **SA2** provides a solution that has a WIP value which is extremely close to the target value. Hence, there is no space left for any improvement when the additional steps of **SA3** are performed. This might indicate that for the two manufacturing systems examined, **SA2** provides an optimal solution in most cases. Note that **SA3** has a complexity which is a factor $J$ higher than **SA2**. Therefore, in practice, **SA2** would be the preferred algorithm. However, in some special cases (see the discussion at the end of Section 2), **SA3** can indeed provide better solutions than **SA2**. Figures 1 and 2 show the results of heuristics **SA1**, **SA2**, and **SA3** for system 1.

From Figures 1 and 2 we see that the relative errors decrease substantially when the improved algorithms **SA2** and **SA3** are used. Especially when **SA1** produces large relative errors (up to 30%), the improved algorithms are able to cut the relative errors substantially. Heuristic **SA1** produces large relative errors when the last 'greedy' server added by the algorithm increases the WIP by a large amount (relative to the existing difference with $W_T$). If this is the case, the improved algorithms have a lot of 'space' between $W_T$ and the WIP produced by **SA1** to find improvements. This is not the case when the relative errors produced by **SA1** are small. In most cases, where **SA1** produced small relative errors (<2%), **SA2** produced the same solution.

Another improvement of **SA2** and **SA3** over **SA1** is the monotonic behavior of the relative errors. Although the general trend of (12) for **SA1** is decreasing when $W_T$ decreases (this is to be expected because the constraints get tighter because of the convex

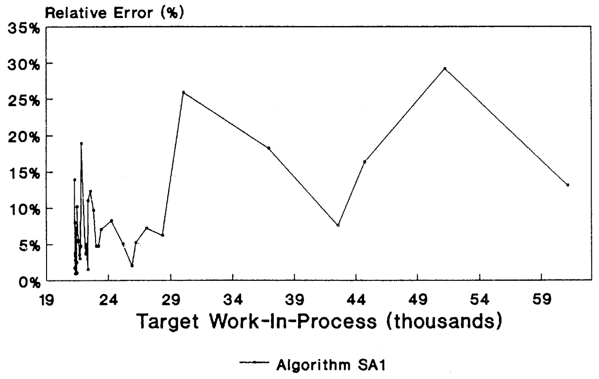

**Figure 1.** Relative error algorithm **SA1** for system 1.

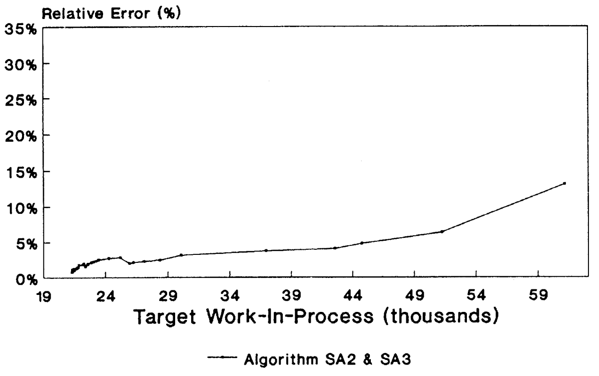

**Figure 2.** Relative error algorithms **SA2** and **SA3** for system 1.

**Table I**
Behavior of Heuristics for Systems 1 and 2

| Heuristic | Average Relative Error (%) | Standard Deviation |
|---|---|---|
| System 1 | | |
| **SA1** | 7.44 | 0.064 |
| **SA2** & **SA3** | 2.13 | 0.014 |
| System 2 | | |
| **SA1** | 4.71 | 0.028 |
| **SA2** & **SA3** | 1.61 | 0.006 |

behavior of $L(m)$), the specific behavior of (12) for **SA1** is unpredictable. The relative errors produced by **SA2** and **SA3**, however, show an almost monotonic decreasing behavior. Hence, when $W_T$ decreases, **SA2** and **SA3** are almost surely to give a better solution. Table I shows the average relative errors over all target WIP values and the corresponding standard deviations for the heuristics. We see that **SA2** and **SA3** improve the quality of the solution by a considerable amount. The results for system 2 (see Figures 3 and 4) show a similar behavior of the different heuristics (see Table I).

## 4. CONCLUSIONS AND SUGGESTIONS FOR FURTHER RESEARCH

We have presented two algorithms for approximately solving a machine allocation problem that arises in the area of manufacturing system design. The optimization problem is particularly relevant within the context of flexible manufacturing, where the issue of optimal capacity allocation is important and prevalent. The optimization problem formulated is a general resource allocation problem in which both the objective function and the constraint functions are not linear and the decision variables show a discrete nature.

The algorithms presented are improvements over a greedy algorithm by Boxma et al. Whereas they did not give any worst-case performance guarantees for their algorithm, we prove our algorithms to have worst-case bounds of, respectively, 2 and 3/2. We have applied the algorithms to two manufacturing systems taken from practice. For these two manufacturing systems we compare the relative error made by the algorithm of Boxma et al. and the two algorithms presented. The results show that the average relative error made by the improved algorithms is substantially smaller than the average relative error made by the algorithm of Boxma et al.

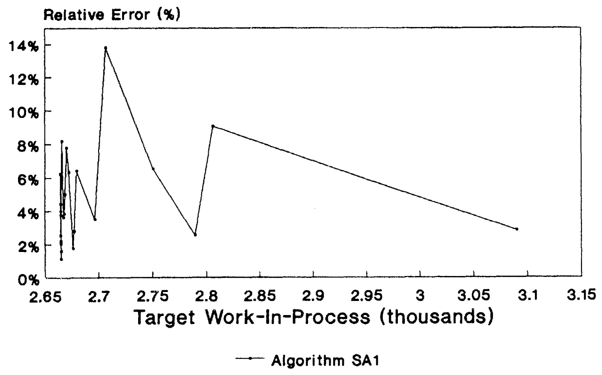

**Figure 3.** Relative error algorithm **SA1** for system 2.

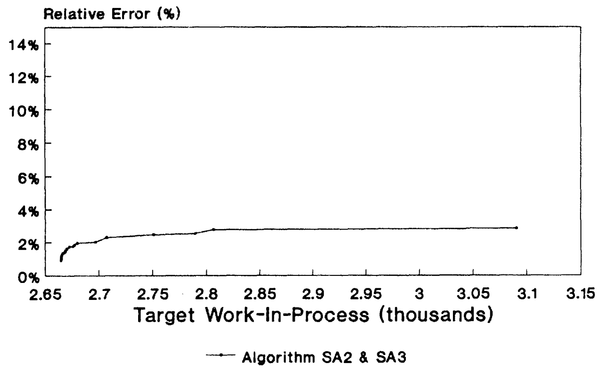

**Figure 4.** Relative error algorithms **SA2** and **SA3** for system 2.

Although outside the scope of this paper, the above analysis can be extended by presenting an algorithm for which we can prove an $\epsilon$ approximation scheme. This indicates that the resource allocation problem treated, although belonging to the class of NP-hard problems, is relatively easy to solve. This is of particular interest, because the presented optimization problem is applicable to a variety of resource allocation problems. We see it as an interesting challenge to investigate the performance of the presented algorithms for other resource allocation problems.

## ACKNOWLEDGMENT

We are grateful to Charles Corbett of INSEAD, France, for the careful reading of an earlier version of the manuscript. Furthermore, two anonymous referees, the associate editor, and the area editor are gratefully acknowledged for their constructive remarks.

## REFERENCES

BOXMA, O. J., A. H. G. RINNOOY KAN AND M. VAN VLIET. 1990. Machine Allocation Problems in Manufacturing Networks. *Eur. J. Opnl. Res.* **45,** 47–54.

CSIRIK, J., J. B. G. FRENK, M. LABBÉ AND S. ZHANG. 1990. Heuristics for the 0–1 Knapsack Problem. Technical Report 9013/A. Econometric Institute, Erasmus University, Rotterdam, The Netherlands.

DALLERY, Y., AND K. E. STECKE. 1990. On the Optimal Allocation of Servers and Workloads in Closed Queueing Network. *Opns. Res.* **38,** 694–703.

DALLERY, Y., AND Y. FREIN. 1986. An Efficient Method to Determine the Optimal Configuration of a Flexible Manufacturing System. In *Proceedings of the Second ORSA/TIMS Conference on Flexible Manufacturing Systems: Operations Research Models and Applications*, K. E. Stecke and S. Suri (eds.). North-Holland, Amsterdam.

DYER, M. E., AND L. G. PROLL. 1977. On the Validity of Marginal Analysis for Allocating Servers in *M/M/c* Queues. *Mgmt. Sci.* **23,** 1019–1022.

FOX, B. 1966. Discrete Optimization via Marginal Analysis. *Mgmt. Sci.* **13,** 210–216.

KELLY, F. P. 1979. *Reversibility and Stochastic Networks*. John Wiley, New York.

SHANTHIKUMAR, J. G., AND D. D. YAO. 1987. Optimal Server Allocation in a System of Multiserver Stations. *Mgmt. Sci.* **33,** 1173–1180.

SHANTHIKUMAR, J. G., AND D. D. YAO. 1988. On Server Allocation in Multiple Center Manufacturing Systems. *Opns. Res.* **36,** 333–342.

TIJMS, H. C. 1986. *Stochastic Modelling and Analysis*. John Wiley, New York.

VAN VLIET, M. 1991. Optimization of Manufacturing System Design. Ph.D Thesis, Subseries B, No. 10, Tinbergen Institute, Erasmus University Rotterdam, The Netherlands.

VAN VLIET, M., AND A. H. G. RINNOOY KAN. 1991. Machine Allocation Algorithms for Job Shop Manufacturing. *J. Intell. Mfg.* **2,** 83–94.

VINOD, B., AND J. J. SOLBERG. 1985. The Optimal Design of Flexible Manufacturing Systems. *Int. J. Prod. Res.* **23,** 1141–1151.