Improved Algorithms for Machine Allocation in Manufacturing Systems

Hans Frenk* Martine Labbé† Mario van Vliet‡
Shuzhong Zhang§

October, 1992

*Econometric Institute, Erasmus University Rotterdam, the Netherlands.

†Econometric Institute, Erasmus University Rotterdam, the Netherlands. Current Address: Université Libre de Bruxelles, B-1050 Bruxelles, Belgium.

‡Tinbergen Institute, Erasmus University Rotterdam, the Netherlands. Current Address: Nederlandse Aardolie Maatschappij B.V., P.O. Box 28000, 9400 HH Assen, the Netherlands.

§Department of Econometrics, University of Groningen, the Netherlands. This research was finished while the author was working at Tinbergen Institute, Erasmus University Rotterdam, the Netherlands.

Abstract

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 $\frac{3}{2}$ respectively. The machine allocation problem considered is a general convex resource allocation problem, which makes the algorithms applicable to a large variety of resource allocation problems. Numerical results are presented for two real-life manufacturing systems.

Keywords: Queueing Network, Generalized Knapsack Problems, Machine Allocation, Heuristics, Worst-case Analysis.

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 to allocate a fixed number of machines among different workstations such that the performance of the system (e.g. in terms of Work-In-Process or leadtimes 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 individual deterministic route. A workstation consists of several parallel identical machines (servers). In the sequel we assume that product routings, amount of traffic offered, location of machines, and technology (e.g. processing times) are already 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), Shanthikumar and Yao (1987, 1988), Vinod and Solberg (1985), Dallery and Frein (1986), and Dallery and Stecke (1990) considered various different server allocation problems. The server allocation issues addressed in these papers differ according to the optimization problem treated, the kind of manufacturing system analysed 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 respectively as an $M/M/m$ and a $GI/G/m$ queue. In Boxma et al. (1990) the so-called *server allocation* problem is presented. For this problem Boxma et al. 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 as presented in Boxma et al. (1990). 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 $\frac{3}{2}$ 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 $\frac{1}{\mu_j}$. 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 λ^i, and then follow a deterministic route through a subset of the set of workstations. A product may visit a workstation more than once, but for simplicity we shall not allow two successive stages of a product route to be identical. Furthermore, it is assumed that the arrival processes and service processes are independent. The assumptions that service times are independently and exponentially distributed, and that 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 the combinatorial optimization problems rather than on the queueing network analysis, we prefer to model the manufacturing system as a queueing network that can be analysed exactly. However, for some practical environments the exponential and Poisson assumptions might not be valid. In Van Vliet (1991), it is shown 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 λ_j at each workstation j. The joint equilibrium queue length distribution in the system has a product form (cf.

Kelly (1979), corollary 3.4). In the steady state each workstation j behaves as an $M/M/m_j$ queue. 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), pp. 332).

$$L_j(m_j, \mu_j, \lambda_j) =$$

$$\frac{(\frac{\lambda_j}{\mu_j})^{m_j}(\frac{1}{\mu_j m_j})}{m_j!(1-\frac{\lambda_j}{\mu_j m_j})^2}\{\sum_{k=0}^{m_j-1}\frac{(\frac{\lambda_j}{\mu_j})^k}{k!}+\frac{(\frac{\lambda_j}{\mu_j})^{m_j}}{m_j!(1-\frac{\lambda_j}{\mu_j m_j})}\}^{-1}+\frac{\lambda_j}{\mu_j} \tag{1}$$

In the sequel we assume that the arrival and service rates are given, while the numbers 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 shall 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 much 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

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

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 non-decreasing function in m_j. In order to prevent the system from becoming

instable, we have to require *traffic intensity* at a workstation $j(\equiv \rho_j = \frac{\lambda_j}{\mu_j m_j})$ to be less than one. It is easy to verify that this results in requiring that $m_j \geq m_j^L = \lfloor \frac{\lambda_j}{\mu_j} \rfloor + 1$, where $\lfloor \cdot \rfloor$ represents the integer rounddown operation.

For convenience we use the following notation.

$$m = (m_1, \cdots, m_J)$$

$$S = \{m \mid m_j \geq m_j^L\}$$

$$F(m) = \sum_{j=1}^{J} F_j(m_j)$$

$$L(m) = \sum_{j=1}^{J} v_j L_j(m_j)$$

$$\Delta F_j(m_j) = F_j(m_j) - F_j(m_j - 1) \qquad (m_j > m_j^L)$$

$$\Delta L_j(m_j) = v_j(L_j(m_j - 1) - L_j(m_j)) \qquad (m_j > m_j^L)$$

From the convexity of F_j and L_j ($j \in \{1, \cdots, J\}$) 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)

$$\min_m F(m)$$

$$\text{s.t. } L(m) \leq W_T$$

$$m_j \geq m_j^L, \quad m_j \text{ integer } (j \in \{1, \cdots, J\})$$

2 Algorithms and Their Analysis

Since problem (SA) can be regarded as a generalization of the knapsack problem it is an NP-hard problem and therefore our focus shall be on algorithms to find approximately optimal solutions. We shall discuss two such algorithms. However, to make this paper self-contained we first briefly mention the results by Boxma et al. (1990), since the algorithms and results presented in their study serve as a starting point for our analysis.

The algorithm by Boxma et al. (1990) 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 Work-In-Process 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

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

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. (1990) prove the following results on algorithm SA1.

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

Lemma 2 *If $c^0, \cdots, c^p$ 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 algorithm 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 algorithm SA1 provides bounds to check whether the allocation found by algorithm 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 the allocation as

given by SA1. To get the second allocation, algorithm 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 algorithm SA2. This procedure can be stated as follows.

Algorithm SA2

Initialization Set $c^{H_1} := c^{SA1}$; $C := \{c^{H_1}\}$; $A := \{1, \cdots, J\}$; $k := 1$.

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. $A := A \backslash \{j_k\}$; $c^{H_{k+1}} = (c_1^{H_k}, \cdots, c_{j_k}^{H_k} - 1, \cdots, c_J^{H_k})$; $k := k + 1$.

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 $L(c^{H_k}) \leq W_T$

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 algorithm SA2, we 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*

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

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}, $i \in \{1, \cdots, J\}$. Note that $c_i^{H_J} = c_i^{H_i} - 1$, for $i = 1, \cdots, J-1$. Furthermore, $(c_1^{H_J}, \cdots, c_{J-1}^{H_J}, c_J^{H_J} - 1)$ is infeasible with regard to (SA). Hence, there exists at least one index j for which $c_j^* \geq c_j^{H_j}$. Let d be the smallest index for which this is satisfied, i.e.,

$$c_d^* \geq c_d^{H_d} \text{ and } c_j^* < c_j^{H_j} \text{ for } j = 1, \cdots, d-1$$

In the proof we use the following two relations.

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

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

When applying SA2, servers are successively added to the workstations in a non-decreasing order of the ratio's $\frac{\Delta F_j(m_j)}{\Delta L_j(m_j)}$. 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

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

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

We know that

$$F(c^{SA2}) - F(m^L) \leq F(c^{H_d}) - F(m^L) =$$

$$F(c^*) - F(m^L) + \sum_{j:c_j^{H_d} > c_j^*} \sum_{i=c_j^*+1}^{c_j^{H_d}} \Delta F_j(i) -$$

$$\sum_{\substack{j:c_j^{H_d} < c_j^* \\ j \neq d}} \sum_{i=c_j^{H_d}+1}^{c_j^*} \Delta F_j(i) - \sum_{i=c_d^{H_d}+1}^{c_d^*} \Delta F_d(i) \tag{6}$$

Furthermore, (4) implies that

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

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

$$L(c_1^{H_d}, \cdots, c_{d-1}^{H_d}, c_d^{H_d} - 1, c_{d+1}^{H_d}, \cdots, c_J^{H_d}) =$$

$$L(m^L) - \sum_{j:c_j^{H_d} > c_j^*} \sum_{i=m_j^L+1}^{c_j^*} \Delta L_j(i) - \sum_{j:c_j^{H_d} > c_j^*} \sum_{i=c_j^*+1}^{c_j^{H_d}} \Delta L_j(i) -$$

$$\sum_{\substack{j:c_j^{H_d} \leq c_j^* \\ j \neq d}} \sum_{i=m_j^L+1}^{c_j^{H_d}} \Delta L_j(i) - \sum_{i=m_j^L+1}^{c_d^{H_d}-1} \Delta L_d(i) > W_T \tag{8}$$

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

$$L(c^*) = L(m^L) - \sum_{j:c_j^{H_d} > c_j^*} \sum_{i=m_j^L+1}^{c_j^*} \Delta L_j(i) - \sum_{\substack{j:c_j^{H_d} \leq c_j^* \\ j \neq d}} \sum_{i=m_j^L+1}^{c_j^{H_d}} \Delta L_j(i) -$$

$$\sum_{\substack{j:c_j^{H_d} < c_j^* \\ j \neq d}} \sum_{i=c_j^{H_d}+1}^{c_j^*} \Delta L_j(i) - \sum_{i=m_j^L+1}^{c_d^{H_d}-1} \Delta L_d(i) -$$

$$\sum_{i=c_d^{H_d}}^{c_d^*} \Delta L_d(i) \leq W_T \tag{9}$$

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

$$\sum_{j:c_j^{H_d} > c_j^*} \sum_{i=c_j^*+1}^{c_j^{H_d}} \Delta F_j(i) \leq$$

$$\frac{\Delta F_d(c_d^{H_d})}{\Delta L_d(c_d^{H_d})} [\sum_{\substack{j:c_j^{H_d} < c_j^* \\ j \neq d}} \sum_{i=c_j^{H_d}+1}^{c_j^*} \Delta L_j(i) + \sum_{i=c_d^{H_d}}^{c_d^*} \Delta L_d(i)] \leq$$

$$\sum_{\substack{j:c_j^{H_d}<c_j^* \\ j\neq d}} \sum_{i=c_j^{H_d}+1}^{c_j^*} \Delta F_j(i) + \sum_{i=c_d^{H_d}}^{c_d^*} \Delta F_d(i) \tag{10}$$

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, substuting (10) in (6) yields

$$F(c^{SA2}) - F(m^L) \leq F(c^*) - F(m^L) + \Delta F_d(c_d^{H_d}) \leq$$

$$2(F(c^*) - F(m^L)) \tag{11}$$

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 following problem instance

$$J = 3 \quad L(m^L) - W_T = C + 1 \quad F_j(m_j^L) = 0 \ \ (j = 1, 2, 3)$$

$$\Delta L_1(m_1^L + 1) = \Delta F_1(m_1^L + 1) = 1$$

$$\Delta L_2(m_2^L + 1) = C \quad \Delta F_2(m_2^L + 1) = C + \epsilon \ \ (\epsilon > 0)$$

$$\Delta L_3(m_3^L + 1) = \Delta F_3(m_3^L + 1) = C - 1$$

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

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

$$\frac{2(C+1+\epsilon)}{C+1+\epsilon} - \frac{1+\epsilon}{C+1+\epsilon} = 2 - \frac{1+\epsilon}{C+1+\epsilon}$$

which can be arbitrarily close to 2 for large C and small ϵ.

□

From Theorem 1 it follows that algorithm 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, as generated by algorithm SA2, as initial allocations. For each of the J allocations c^{H_k} $(k \in \{1, \cdots, J\})$, as generated by algorithm SA2, algorithm 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, algorithm SA2 is applied to each such new problem (SA_k), $(k \in \{1, \cdots, J\})$ to generate new feasible solutions. The heuristic solution returned by algorithm SA3 is the allocation generated by SA2 when applied to (SA) or (SA_k) $(k \in \{1, \cdots, J\})$ which minimizes the objective function.

Algorithm SA3

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

Step 2 For $k := 1$ to J do

construct the following problem.

Problem (SA_k)

$$\min_{m_j} \sum_{j=1}^{J} F_j(m_j)$$

$$\text{s.t.} \sum_{j=1}^{J} v_j L_j(m_j) \leq W_T$$

$$m_j \geq m_j^L, \quad m_j \text{ integer } (j \in \{1, \cdots, J\} \backslash \{k\})$$

$$m_k \geq c_k^{H_k}, \quad m_k \text{ integer}$$

Apply algorithm SA2 to problems (SA_k) and denote the heuristic solutions by

$$\bar{c}^{H_k} = (\bar{c}_1^{H_k}, \cdots, \bar{c}_J^{H_k})$$

Step 3 Let

$$\bar{C} \ = \ (\bar{c}^{H_1}, \cdots, \bar{c}^{H_J})$$

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

Note that the complexity of algorithm 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*

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

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 $j = 1, \cdots, d-1$.

We now distinguish between the following two cases:

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

In this case it follows directly from (11) that

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

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

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

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

$$F(c^{SA3}) - F(m^L) \leq F(\bar{c}^{H_d}) - F(m^L) =$$

$$F(\bar{c}^{H_d}) - \sum_{\substack{j=1 \\ j \neq d}}^{J} F_j(m_j^L) - F_d(c_d^{H_d}) + F_d(c_d^{H_d}) - F_d(m_d^L) \leq$$

$$2(F(\bar{c}^*) - \sum_{\substack{j=1 \\ j \neq d}}^{J} F_j(m_j^L) - F_d(c_d^{H_d})) + F_d(c_d^{H_d}) - F_d(m_d^L) \leq$$

$$2(F(c^*) - \sum_{\substack{j=1 \\ j \neq d}}^{J} F_j(m_j^L) - F_d(c_d^{H_d})) + F_d(c_d^{H_d}) - F_d(m_d^L) =$$

$$2(F(c^*) - F(m^L)) - (F_d(c_d^{H_d}) - F_d(m_d^L)) \leq$$

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

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

The second inequality is obtained by applying Theorem 1 to problem (SA_d) and the third inequality follows from the fact that c^* is feasible to (SA_d) given the definition of problem (SA_d).

□

Note that the situation in which $\Delta F_d(c_d^{H_d}) > \frac{1}{2}(F(c^*) - F(m^L))$ 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 very well imply that $\Delta F_d(c_d^{H_d}) \leq \frac{1}{2}(F(c^*) - F(m^L))\ \forall d \in J$. In this case, it follows from the above that algorithm SA2 also has a worst-case performance ratio of $\frac{3}{2}$. In section 3 we show that the numerical results obtained for the practical settings we considered are similar for algorithms 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 (1991) 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). How to apply the above optimization problems to (more realistic) non-Markovian queueing networks we refer to Van Vliet and Rinnooy Kan (1991).

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

$$\text{Relative Error of SA}i = \frac{UB_i - LB_i}{\frac{UB_i + LB_i}{2}} \tag{12}$$

We performed the heuristics for a wide range of WIP values. It appeared that the results for heuristics SA2 and SA3 did not differ for the two manufacturing systems we examined. This means that SA2 provides us with a solution that has a WIP value which is already 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 in most cases an optimal solution. 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, and SA2 & SA3 for System 1.

Figure 1: System 1 - Relative Error Algorithm SA1

Figure 2: System 1 - Relative Error Algorithm SA2 & SA3

From figures 1 and 2 we see that the relative errors decrease substantially when the improved algorithms SA2 & 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 & 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 since the constraints get tighter because of the convex behavior of $L(m)$), the specific behavior of (12) for SA1 is very unpredictable. The relative errors produced by SA2 & SA3, however, show an almost monotonic decreasing behavior. Hence, when W_T decreases, SA2 & SA3 are almost surely to give a better solution. In table 1 we show the average relative errors over all Target WIP values and the corresponding standard deviations for the heuristics. We see that SA2 & SA3 improve the quality of the solution by a considerable amount.

System 1		
Heuristic	Average Relative Error	Standard Deviation
SA1	7.44 %	0.064
SA2 & SA3	2.13 %	0.014

Table 1

The following results for System 2 show a similar behavior of the different heuristics.

Figure 3: System 2 - Relative Error Algorithm SA1

Figure 4: System 2 - Relative Error Algorithm SA2 & SA3

System 2		
Heuristic	Average Relative Error	Standard Deviation
SA1	4.71 %	0.028
SA2 & SA3	1.61 %	0.006

Table 2

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 an important and prevalent one. The optimization problem formulated is in fact 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 as presented by Boxma et al. (1990). Whereas Boxma et al. did not give any worst-case performance guarantees for their algorithm, we prove our algorithms to have worst-case bounds of respectively 2 and $\frac{3}{2}$. 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 by Boxma et al. and the two algorithms presented. The results show that the average relative error made by the improved algorithms are substantially smaller than the average relative error made by the algorithm of Boxma et al.

Although outside the scope of this paper, the above analysis can be extended by presenting an algorithm for which we can prove an ϵ approximation scheme. This indicates that the resource allocation problem treated, although belonging to the class of NP-hard problems, is a relatively easy problem to

solve. This is of particular interest, since the presented optimization problem is applicable to a large 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.

Acknowledgements.

We are very 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

[1] Boxma, O. J., Rinnooy Kan, A. H. G. and Van Vliet, M. 1990. Machine Allocation Problems in Manufacturing Networks. *European Journal of Operational Research* 45, 47-54.

[2] Csirik, J., Frenk, J. B. G., Labbé, M. and Zhang, S. 1990. Heuristics for the 0-1 Knapsack Problem. Technical Report 9013/A. Econometric Institute, Erasmus University Rotterdam.

[3] Dallery, Y. and Frein, Y. 1986. An Efficient Method to Determine the Optimal Configuration of a Flexible Manufacturing System. In Stecke and Suri, editors, *Proceedings of the Second ORSA/TIMS Conference on Flexible Manufacturing Systems: Operations Research Models and Applications.* North-Holland, Amsterdam.

[4] Dallery, Y. and Stecke, K. E. 1990. On the Optimal Allocation of Servers and Workloads in Closed Queueing Network. *Operations Research* 38, 694-703.

[5] Dyer, M. E. and Proll, L. G. 1977. On the Validity of Marginal Analysis for Allocating Servers in M/M/c Queues. *Management Science* 23, 1019-1022.

[6] Fox, B. 1966. Discrete Optimization via Marginal Analysis. *Management Science* 13, 210-216.

[7] Kelly, F. P. 1979. *Reversibility and Stochastic Network*. John Wiley, New York.

[8] Shanthikumar, J. G. and Yao, D. D. 1988. On Server Allocation in Multiple Center Manufacturing Systems. *Operations Research* 36, 333-342.

[9] Shanthikumar, J. G. and Yao, D. D. 1987. Optimal Server Allocation in a System of Multi-server Stations. *Management Science* 33, 1173-1180.

[10] Tijms, H. C. 1986. *Stochastic Modelling and Analysis*. John Wiley, New York.

[11] Van Vliet, M. 1991. *Optimization of Manufacturing System Design.* Ph.D Thesis Subseries B, No. 10, Tinbergen Institute, Erasmus University Rotterdam, The Netherlands.

[12] Van Vliet, M. and Rinnooy Kan, A. H. G. 1991. Machine Allocation Algorithms for Job Shop Manufacturing. *Journal of Intelligent Manufacturing* 2, 83-94.

[13] Vinod, B. and Solberg, J. J. 1985. The Optimal Design of Flexible Manufacturing Systems. *International Journal of Production Research* 23, 1141-1151.