A novel approach for the design of multi-class reentrant manufacturing systems

Dong Joon Yoo*, Jae Hak Jung**, In-Beum Lee§, Euy Soo Lee†, and Gyeongbeom Yi‡

*Department of Chemical Engineering, POSTECH, Pohang, Korea
(Tel : +82-54-279-5967; E-mail: djyoo@postech.ac.kr)
**School of Chemical Engineering and Technology, Yeungnam University, Gyeongsan, Korea
(Tel : +82-53-810-3241; E-mail: jhjung@yu.ac.kr)
§Department of Chemical Engineering, POSTECH, Pohang, Korea
(Tel : +82-54-279-2274; E-mail: iblee@postech.ac.kr)
†Department of Chemical Engineering, Dongguk University, Seoul, Korea
(Tel : +82-2-2260-3706; E-mail: eslee@dongguk.edu)
‡Division of Chemical Engineering, Pukyong National University, Busan, Korea
(Tel : +82-51-620-1499; E-mail: gbyi@pknu.ac.kr)

Abstract: The design problem of manufacturing system is addressed, adopting the closed queueing network model with multiple loops and re-entrant flows. The entire design problem is divided into two hierarchical sub-problems of (1) determining the station configuration and (2) optimizing the lot constitution; then they are tackled by neighbor search algorithm (NSA) and greedy mean value analysis (GMVA), respectively. Unlike the conventional MVA concerning multi-class closed queueing networks, the GMVA doesn't stick to a fixed lot proportion; rather it tries to find the optimal balance. The NSA, on the other hand, improves the object function value by altering the station configuration successively with its superior neighbor. The moderate time complexity, presented in big-*O* notation, enables us to apply the method even to the large-size practical cases, and the CPU time of an enlarged problem can be approximated by the same equation. The validity of our analytic approach is backed up by simulation studies with a widespread simulation package.

Keywords : manufacturing system, design, queueing network, mean value analysis

1. INTRODUCTION

Queueing network, or network of queues, is a useful tool for modeling, analyzing, and designing a wide range of systems including computer architectures, manufacturing systems, and communication/transportation networks. It can be described as a set of nodes (standing for servers, processors, or delayers) in which customers (representing data, jobs, or transporters) visit one after another in either deterministic or stochastic order until all of their requirements are fulfilled. Concerning flexible manufacturing system (FMS), semiconductor manufacturing system, or a certain type of chemical plant, some nodes need to be visited more than once, as well as more than one class of customers, differing in their routes and/or processing times, must be taken into account.

There were a series of *evaluation models* for this type of system, yielding relatively accurate performance measures such as throughput, production cycle time, and so on [6]. For example, Narahari and Khan [5] applied the mean value analysis (MVA) to the system with re-entrant lines under a variety of scheduling policies. Park et al. [7-8] generalized the above work by incorporating batch machines and multi-class jobs. Despite their accuracy and swiftness in computation, these models are not for finding the optimal configuration; rather the configuration must be given as input parameters. On the other side, a number of *generative models* have been proposed since Vinod and Solberg [10] formulated the optimal configuration problem for FMSs [1, 3-4]. However, these researches have some limitations: (1) they assumed universal pellets rather than more general special pellets, (2) the production amount for each product was given as a parameter rather than regarded as a decision variable and adjusted for the maximum profit of the system, (3) the results were not compared with those of simulation studies, hence the validities of models were not checked.

Therefore in this research, combining the advantages of the evaluation and the generative model, we develop a novel method that optimizes the system configuration as well as the production amounts subject to the constraints considering the performance measures with accuracies. In section 2, we describe a closed queueing network model of the manufacturing system with re-entrances and multi-class jobs, and specify the assumptions. In section 3, a bi-level optimization scheme for the design of the manufacturing system and the worst-case time complexity of the algorithm are presented. In section 4, our method is applied to a medium-size design problem. In section 5, we conclude and present some plans for future studies.

2. PROBLEM DEFINITION

We consider a system composed of several stations for producing a number of products simultaneously (see Fig. 1). Each station is equipped with one or more parallel processors of identical performance and each product is produced by visiting and taking services from a number of stations in a predetermined manner. The route of production for each product makes a closed loop in the system. On its visit to a station, a lot is put into a buffer, a space for both waiting and processing, and waited until all the preceding lots are done and at least one processor becomes available. Each buffer in a station corresponds to a specific product or a specific reentrance stage of a product. The processing times in stations are assumed to be independent and exponentially distributed, where the means may differ by buffer. We assume that the setup times for changing the lot classes are negligible and all the buffer capacities are unlimited. The transportation between stations are carries out by automated guided vehicles (AGV), which is regarded as a pseudo-station holding as many parallel processors as the number of AGVs. The processing time for each pass of AGV transportation must include the times for the following consecutive stages:

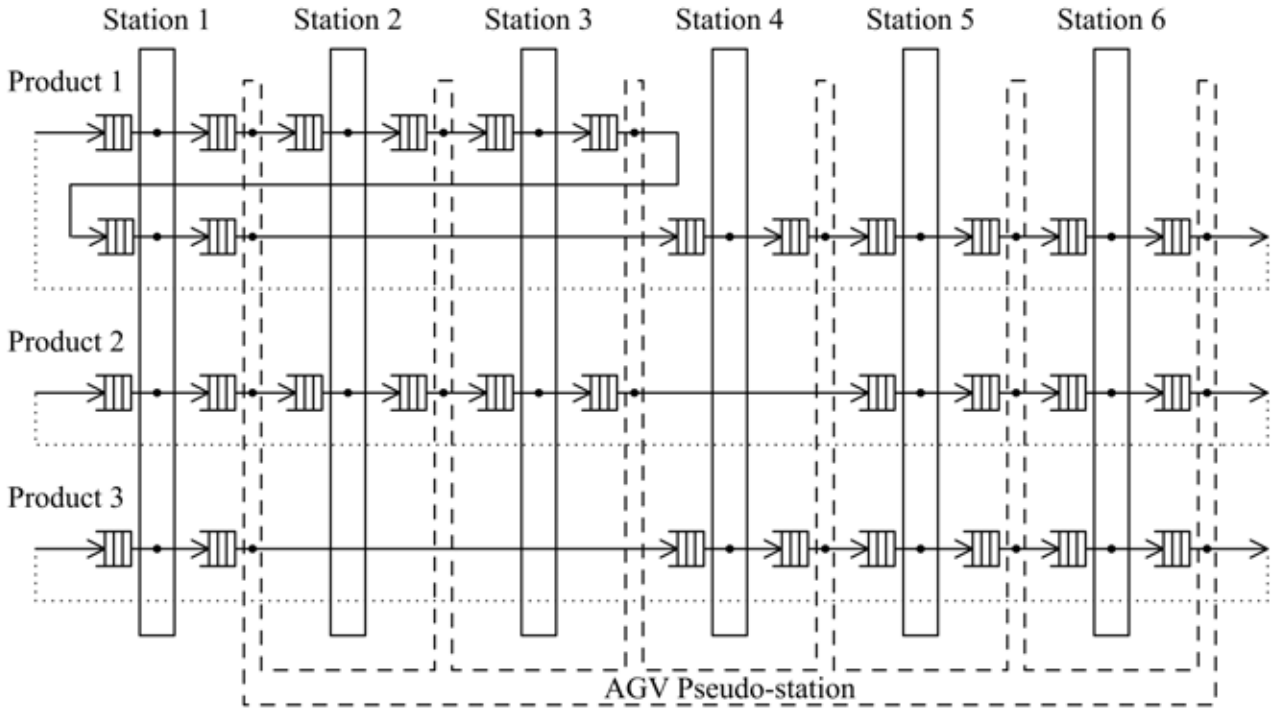

Fig. 1 The closed queueing network model of a manufacturing system.

i. Locate the AGV in the departure station.
ii. Load the lot onto the AGV.
iii. Transfer the AGV to the arrival station.
iv. Unload the lot from the AGV.

Since, having completed a production cycle, each lot returns to the first processing station and restarts another cycle, the number of lots, or work in processes (WIP), in each closed loop always remains unchanged. Stations may have multiple buffers, where first come first served (FCFS) scheduling policy are prevailed; the lot that has arrived earliest to the station will be served next regardless of buffer priorities. For the design problem, we assume that the following information is given beforehand:

$f_i^{\text{profit}}(P_i)$	annual profit [$/yr]
$P_i^{\min}$, $P_i^{\max}$	bounds on annual production [ea/yr]
$f_j^{\text{cost}}(m_i)$	annual cost for lot [$/yr]
$m_i^{\max}$	maximum number of lots [lot]
$f_j^{\text{cost}}(n_j)$	annual cost of station [$/yr]
$n_j^{\min}$, $n_j^{\max}$	bounds on number of parallel processors [unit]

The decision variables are the number of parallel processors in every station, referred to as *station configuration*, and the number of cycling lots in every closed loop, or *lot constitution*. The objective of the optimization is to maximize the annual profit while satisfying a number of constraints. In this research, we attempt a bi-level approach to find a near optimum solution of practical-size problem within an acceptable computational time, where the station configuration is successively improved by *neighbor search algorithm*, and the lot constitution is optimized by *greedy mean value analysis*.

3. DESIGN PROCEDURE

3.1 Greedy Mean Value Analysis

The mean value analysis (MVA) is an iterative procedure, developed by Reiser and Lavenberg [9], to evaluate the mean performance measures such as mean queue lengths, mean residence times, and mean throughputs of closed queueing networks. So far it has been applied to the performance evaluations of FMSs, semiconductor manufacturing systems, and so on, but there's been a critical assumption, i.e. the proportion of the lots in each closed loop to the total lots, or

$$r_i = M_i / \sum_{i \in I} M_i \qquad \forall i \in I, \tag{1}$$

must be given *a priori*. Then using the MVA, after defining the initial conditions as

$$L_{jk}(0) = 0 \qquad \forall j \in J, k \in K_j, \tag{2}$$

we can compute the performance measures by successively applying Eqs. (3) ~ (6) for $m = 1, 2, \ldots, M\ (= \Sigma M_i)$:

$$W_{jk}(m) = \sum_{k' \in K_j} \frac{L_{jk'}(m-1)\, p_{jk'}}{n_j} + \frac{p_{jk}}{n_j} \quad \forall j \in J, k \in K_j, \tag{3}$$

$$W_i(m) = \sum_{j \in J_i} \sum_{k \in K_{ij}} W_{jk}(m) \qquad \forall i \in I, \tag{4}$$

$$\lambda_i(m) = \frac{m \cdot r_i}{W_i(m)} \qquad \forall i \in I, \tag{5}$$

$$L_{jk}(m) = \lambda_i(m) W_{jk}(m) \qquad \forall i \in I,\ j \in J_i, k \in K_{ij}. \tag{6}$$

To apply the MVA to design problems, however, we have to introduce a procedure for finding the optimal lot composition in place of the above assumption; which has motivated us to develop the greedy mean value analysis (GMVA). Before describing the procedure of GMVA, we have to classify the constraints into three categories with respect to a closed loop *i*:

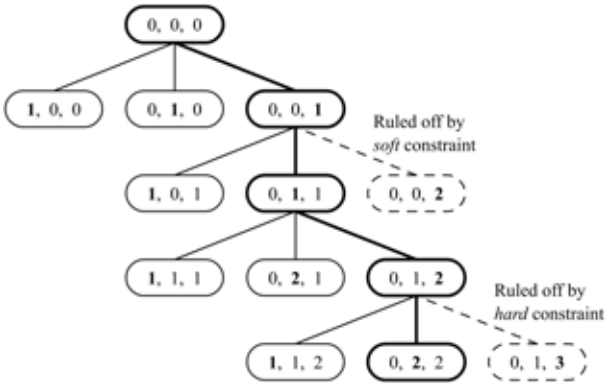

Fig. 2 The greedy mean value analysis.

i. *Hard constraint*: likely to be violated as a lot for product *i* is augmented.
ii. *Soft constraint*: likely to be fulfilled as a lot for product *i* is augmented.
iii. *Non-constraint*: indifferent to the augmentation of product *i* lot.

Then, starting with an empty lot constitution, all the candidates of one lot increased are tried one after another (refer to Fig. 2). Every time a candidate that improves the current best object function value and satisfies all the hard constraints is found, it replaces the current best lot constitution. If at least one replacement is occurred in the current lever of the tree, the search is kept going in the next level; otherwise the current best lot constitution is returned as the optimal, along with the object function value and the performance measures. During the process, the soft constraints play an important role in branching the candidates; if there are products with respect to which not all soft constraints are satisfied by the current best lot constitution, the lots for these products have priorities to be augmented. As a result, the soft constraints have strong tendencies to be satisfied but may not always be met. On the other hand, the hard constrains strictly check the feasibility of every candidate and discard one unless all of them are fulfilled. The detailed procedure of GMVA is presented in Fig. 3.

3.2 Neighbor Search Algorithm

The main purpose of neighbor search algorithm (NSA) is to successively improve the station configuration by replacing the current choice with one of its neighbors that results in a better object function value. The "neighbor" in this context implies any other station configuration whose difference in the number of parallel processors in each station is less than or equal to the search radius r. There exist at most $|J|^{2r+1} - 1$ neighbors, and this number may decrease if the number of parallel processors in some station is bounded by constraint. In this research, we only consider the case of $r = 1$, accordingly it is possible to revise a station in two ways, i.e. augmenting or removing a single parallel processor, and the maximum number of neighbors is restricted to $|J|^3 - 1$.

Between the two alternatives, adding a new parallel processor is usually effective in a station of high utilization, because the alleviation of the load directly leads to an increase in the production throughput, as well as the annual production, which can overwhelm the cost of the additional processor. On

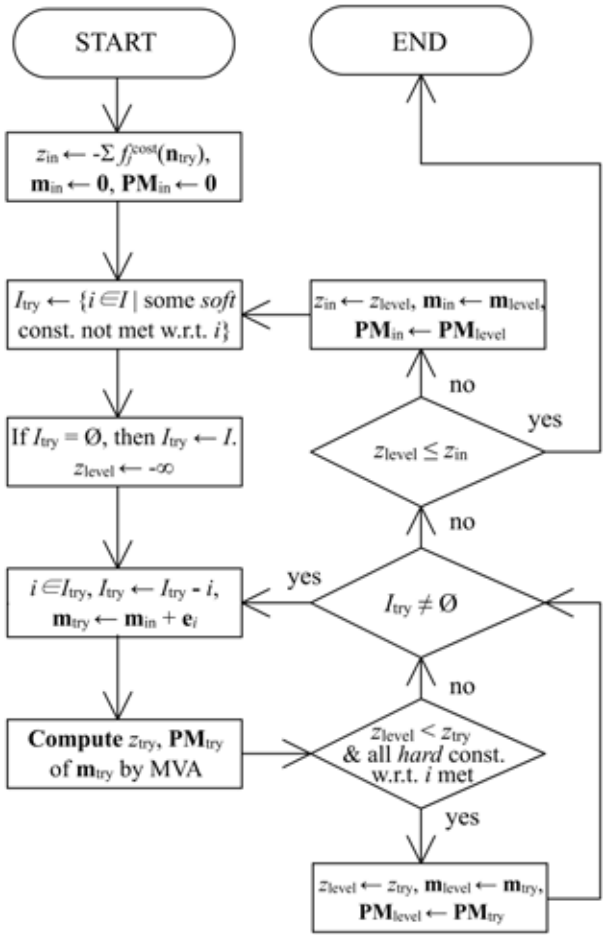

Fig. 3 The flow chart of GMVA.

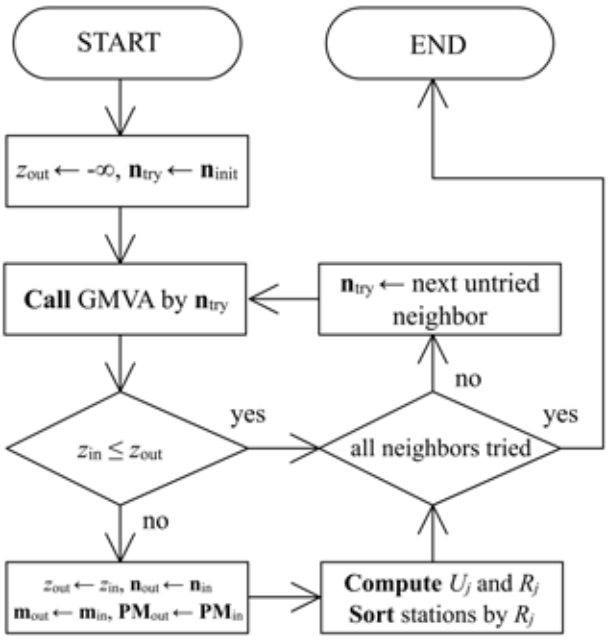

Fig. 4 The flow chart of NSA.

the other hand, the strategy of removing an existing parallel processor is successful in a station of low utilization, for we

Table 1. Binary codes for NSA.

(a) $s = 1$

Code	Revise	Way
	$S_{(0)}$	$S_{[0]}$
$C(1,0,1)$	0	F
$C(1,1,1)$	1	U

(b) $s = 2$

Code	Revise		Way	
	$S_{(1)}$	$S_{(0)}$	$S_{[1]}$	$S_{[0]}$
$C(2,0,1)$	0	0	F	F
$C(2,1,1)$	0	1	U	F
$C(2,1,2)$	1	0	F	U
$C(2,2,1)$	1	1	U	U

(c) $s = 3$

Code	Revise			Way		
	$S_{(2)}$	$S_{(1)}$	$S_{(0)}$	$S_{[2]}$	$S_{[1]}$	$S_{[0]}$
$C(3,0,1)$	0	0	0	F	F	F
$C(3,1,1)$	0	0	1	U	F	F
$C(3,1,2)$	0	1	0	F	U	F
$C(3,1,3)$	1	0	0	F	F	U
$C(3,2,1)$	0	1	1	U	U	F
$C(3,2,2)$	1	0	1	U	F	U
$C(3,2,3)$	1	1	0	F	U	U
$C(3,3,1)$	1	1	1	U	U	U

Table 2. Neighbor sequencing (s = 3).

No.	r	u	w	q	$C(r, u, w)$	$C(s, r, q)$	Neighbor
1	1	0	1	1	F	001	--F
2	1	0	1	2	F	010	-F-
3	1	0	1	3	F	100	F--
4	1	1	1	3	U	100	U--
5	1	1	1	2	U	010	-U-
6	1	1	1	1	U	001	--U
7	2	0	1	1	FF	011	-FF
8	2	0	1	2	FF	101	F-F
9	2	0	1	3	FF	110	FF-
10	2	1	1	3	UF	110	UF-
11	2	1	1	2	UF	101	U-F
12	2	1	1	1	UF	011	-UF
13	2	1	2	1	FU	011	-FU
14	2	1	2	2	FU	101	F-U
15	2	1	2	3	FU	110	FU-
16	2	2	1	3	UU	110	UU-
17	2	2	1	2	UU	101	U-U
18	2	2	1	1	UU	011	-UU
19	3	0	1	1	FFF	111	FFF
20	3	1	1	1	UFF	111	UFF
21	3	1	2	1	FUF	111	FUF
22	3	1	3	1	FFU	111	FFU
23	3	2	1	1	UUF	111	UUF
24	3	2	2	1	UFU	111	UFU
25	3	2	3	1	FUU	111	FUU
26	3	3	1	1	UUU	111	UUU

can save the cost of the uninstalled processor with a minor decrease in production. Here, the utilization of the parallel processors in each station is approximated by the equation

$$\rho_j = \frac{1}{n_j}\left[\frac{\lambda_{j1}}{\mu_{j1}} + \Lambda + \frac{\lambda_{jk}}{\mu_{jk}}\right] \quad \forall j \in J. \tag{7}$$

We assume that a "recommended" utilization is given for each station, and define a "relative" utilization as

$$\rho_j^{\text{rel}} = \frac{\rho_j}{\rho_j^{\text{rec}}} \quad \forall j \in J. \tag{8}$$

The relative utilization greater than unity indicates that the addition of a parallel processor is more preferable than the subtraction, and *vice versa*. Therefore, both the *favorable* and the *unfavorable* way of revision, in each station, are determined straightforwardly by the relative utilization.

There are at most "s choose r", or $C(s, r)$, ways to choose r stations to revise from s stations, each can be represented by an s-digit binary number; e.g. 011, 101, and 110 are those revising 2 stations in a 3-station set, where the digit 1 implies that the corresponding station is to be revised. We sort each set of $C(s, r)$ binary numbers by value, and introduce a notation $C(s, r, q)$ to indicate the qth smallest binary code of revising r stations in an s-station set. To represent the way of revision, we use the above binary code once more after replacing each digit of 0/1 with F(favorable)/U(unfavorable), respectively, and reflect the entire code horizontally. Then, the code $C(r, u, w)$ corresponds to the wth binary code with u unfavorable revisions among r total revisions. The codes for s = 1, 2, and 3 are listed in Table 1.

To facilitate the progress of NSA we give priority to the neighbor of the following attribute:

i. those with fewer station revisions.
ii. those with fewer unfavorable revisions.
iii. those with favorable revision in a higher-relative-utilization station, or unfavorable revision in a lower-relative-utilization station.

When the Rule i(ii) is failed to pick a single candidate, ii(iii) is applied further to get one, respectively. At the beginning of each NSA step, the stations are permutated in order of relative utilization so that $S_{(j)}$ denotes the station of the $(j + 1)$st largest relative utilization. Since the number of stations, s, is invariant, the outermost loop of our algorithm is to increasing the number of revision stations, i.e. $r = 1, 2, \cdots, s$ (Rule i). In the second nested loop, the number of unfavorable revisions is varied as $u = 0, 1, \cdots, r$ (Rule ii). In the third, a binary code $C(r, u, w)$ is selected for $w = 1, 2, \cdots, C(r, u)$ (Rule iii). Finally, in the inner most loop, a binary code $C(s, r, q)$ is chosen for $q = 1, 2, \cdots, C(s, r)$, in *reciprocating* order starting with the increasing direction. In each pass of the innermost loop, the neighbor is obtained by substituting the F's and U's of $C(r, u, w)$ for the 1's of $C(s, r, q)$. The 0's are replaced with minus signs, meaning no revisions in the corresponding stations. The whole procedure for $s = 3$ is demonstrated in Table 2.

The NSA is set off by calling its first MVA with the initial station configuration. The returned object function value, as well as the lot constitution, is assigned to the current optimal,

Table 3. Computational results.

Step	z^* [$/yr]	Calls	Number of Parallel Processors							Number of MV's			Ann. Production [ea/yr]		
			S0	S1	S2	S3	S4	S5	S6	P1	P2	P3	P1	P2	P3
0	-79,647	1	3	3	3	3	3	3	3	1	4	3	2,128	11,826	16,506
1	212,919	1	4	3	3	3	3	3	3	2	3	3	4,909	10,008	22,509
2	440,152	1	5	3	3	3	3	3	3	2	4	3	4,887	12,824	26,161
3	562,658	1	6	3	3	3	3	3	3	2	5	3	4,707	14,954	29,061
4	856,309	1	6	3	4	3	3	3	3	3	8	4	5,834	19,168	29,246
5	946,288	3	6	3	5	3	3	3	3	5	7	5	9,431	16,604	29,689
6	980,060	6	6	2	5	3	3	3	3	5	7	5	9,360	16,631	29,468
7	1,024,842	7	6	2	5	3	3	2	3	5	7	5	9,355	16,631	29,177
8	1,032,022	7	6	2	5	3	3	1	3	6	8	6	9,898	16,500	28,126
9	1,051,243	2	6	2	5	4	3	1	3	5	8	7	9,095	19,244	28,996
10	No Imp.	1,457													
	Total	1,487													24.7 CPU sec

Table 4. Comparison with simulation results.

Product	P1	P2	P3
No. of Movable Vessels	5	8	7
Mean Cycle Time [hr/lot]			
Simulation	15.95	12.67	7.38
GMVA	16.49	12.47	7.24
Relative Error [%]	3.3	-1.6	-1.9
Mean Throughput [lot/hr]			
Simulation	0.313	0.631	0.948
GMVA	0.303	0.641	0.966
Relative Error [%]	-3.3	1.6	1.9
Annual Production [ea/yr]			
Simulation	9,399	18,934	28,445
GMVA	9,095	19,244	28,996
Relative Error [%]	-3.3	1.6	1.9

and the stations are arranged in order of relative utilization. From this initial station configuration, up to $|J|^3$ - 1 neighbors are generated according to the precedence rules, and examined by MVA in sequence. If a better station configuration is found in the course of the neighbor search, the optimal solution is replaced with the returned one, and all the candidates remaining behind are discarded. Then, another step of neighbor search is started after the stations are rearranged by the new relative utilization. If all the neighbors are proved to be inferior to the current optimal, the procedure is terminated by returning the current optimal solution (see Fig. 4 for detailed procedure). The worst case time complexity for a single step of NSA is $O(m^2pq(2r+1)^s)$, where m is the maximum number of lots, p is the number of products, q is the number of queueing buffers, r is the search radius, and s is the number of stations.

4. APPLICATION

The proposed method is applied in the design of a manufacturing system, schematized in Fig. 1. It is composed of 7 stations, including the AGV pseudo-station, and produces three products, referred to as P1, P2, and P3, simultaneously. We assume that the annual working hour is 6,000 [hr/yr], 5 products are produced for every lot regardless of product, and calculate the annual production as

$$P_i = B_i \lambda_i H \qquad \forall i \in I. \tag{9}$$

Therefore, the object function value is evaluated by equation

$$z(m) = \sum_{\forall i \in I} v_i P_i - \sum_{\forall i \in I} c_i m_i - \sum_{\forall j \in J} c_j n_j. \tag{10}$$

As shown in Table 3, on Step 0, the first GMVA is called by the initial station configuration, where every station is allocated three parallel processors, and results in a loss of $79,647 per annum. On the next step, an improved solution is found on the first trial among as many as 2,186 (= 3^7 – 1) neighbors, which is adding a processor to the highest-relative-utilization station, here AGV pseudo-station. Continuing in this fashion, the optimal solution as profitable as $1,051,243 per annum is obtained on Step 9. The solution is proved to be the optimal on the next step for every neighbor within a unit search radius fails to improve the object function value. The effectiveness of our neighbor sequencing is supported by the fact that the improvements are usually achieved in a very early stage of GMVA calls. The soft and the hard constraints seem to work well for the productions are filled out in order of profit-per-investment, here P3 > P 2 > P1, but never exceed the maximum bound. It takes 24.7 CPU seconds, coded in Microsoft Visual Basic for Applications (VBA) and executed on our PC equipped with Intel Pentium 4 Processor (2.53GHz).

In addition to the analytic approaches, simulation studies are also performed using a general-purpose simulation package, Arena [2]. The simulation is maintained until a steady-state is reached, where the results are regarded as of steady-state if the mean cycle times and the mean throughputs fall within 0.1% range from those of the previous 300,000-hour, or 50-working-year, interval. Both analytic and simulation results are summarized in Table 4.

5. CONCLUSION

The complicated design problem of multi-class reentrant closed queueing network has been efficiently tackled, split into two decision levels, viz. mean value analysis (GMVA) and neighbor search algorithm (NSA), by which the decision variables are refined successively and interactively. The GMVA itself could be well combined with such evolutionary programming schemes as genetic algorithm (GA), tabu search (TS), etc., as it returns the object function value of a specific

station configuration, usually represented by some code, along with a well balanced composition of products, within a minimal time required. The precedence rules for the NSA seem to be working nicely for, in most of the steps, a better neighbor is found within a couple of GMVA calls. Especially, from the worst case time complexity of the NSA, it is possible to measure the impact of any extensions in problem dimensions upon the CPU time. As further studies, we are planning to incorporate the diverse scheduling policies other than first come first served (FCFS), for example last come first served (LCFS) or general buffer priorities. To enhance the validity of the model, the assumption that every buffer capacity is infinite must be generalized to properly deal with finite-sized buffers. Finally, the NSA should be upgraded to handle an arbitrary integer search radius.

NOMENCLATURE

Sets

I	set of products
I_j	set of products visiting station j
J	set of stations
J_i	set of stations visited by product i
K	set of buffers
K_j	set of buffers in station j
K_{ij}	set of buffers for product i in station j

Indices

i	product
j	station
k	buffer

Parameters & Variables

B_i	lot size of product i [ea/lot]
c_i	annual unit cost of lot for product i [$/lot/yr]
c_j	annual unit cost of parallel processor in station j [$/unit/yr]
H	annual working hour [hr/yr]
$L_{jk}(m)$	mean length of lots in buffer k of station j provided lot population is m [lot]
$m_i^{\max}$	maximum number of lots for product i [lot]
m_i, M_i	number of lots for product i [lot]
$\mathbf{m}$, $\mathbf{M}$	vector of lot constitution, i.e. $(m_1, m_2, \cdots, m_{\|I\|})$
m, M	total number of lots, or lot population [lot]
$n_j^{\max}$	maximum number of parallel processors in station j [unit]
$n_j^{\min}$	minimum number of parallel processors in station j [unit]
n_j	number of parallel processors in station j [unit]
$\mathbf{n}$	vector of station configuration, i.e. $(n_1, n_2, \cdots, n_{\|J\|})$
$P_i^{\max}$	maximum annual production of product i [ea/yr]
$P_i^{\min}$	minimum annual production of product i [ea/yr]
P_i	annual production of product i [ea/yr]
p_{jk}	mean processing time of lots in buffer k of station j [hr/lot]
$\mathbf{PM}$	vector of performance measures
r_i	proportion of product i lots to total lots
r	search radius
$S_{(j)}$	station indicating $(j+1)$st highest relative utilization
$S_{[j]}$	station to be revised indicating $(j+1)$st highest relative utilization
v_i	unit profit of product i [$/ea]
$W_i(m)$	mean cycle time of product i provided lot population is m [hr]
$W_{jk}(m)$	mean residence time of lots in buffer k of station j provided lot population is m [hr]
z	annual net profit [$/yr]

(Greek letters)

$\lambda_i(m)$	mean throughput of product i provided lot population is m [lot/hr]
λ_{jk}	mean throughput of buffer k in station j [lot/hr]
μ_{jk}	mean processing rate of processor for lots in buffer k of station j, $\mu_{jk} = p_{jk}^{-1}$ [lot/hr]
ρ_j	mean utilization of processors in station j
ρ_j^{rec}	recommended utilization of processors in station j
ρ_j^{rel}	relative utilization of processors in station j

Subscripts

try	of current trial, or lot augmentation
level	of best lot augmentation in a level
in	the optimal solution returned by GMVA
out	the current optimal solution in NSA

REFERENCES

[1] Y. Dallery and Y. Frein, "An efficient method to determine the optimal configuration of a flexible manufacturing system," *Annals of Operations Research*, Vol. 15, pp. 207-225, 1988.

[2] W. D. Kelton, R. P. Sadowski, and D. A. Sadowski, *Simulation with Arena*, McGraw-Hill, New York, 2002.

[3] H. F. Lee and M. M. Srinivasan, and C. A. Yano, "The optimal configuration and workload allocation problem in flexible manufacturing systems," *International Journal of Flexible manufacturing Systems*, Vol. 3, pp. 213-230, 1991.

[4] H. F. Lee, "Optimal design for flexible manufacturing systems: generalized analytical methods," *IIE Transactions*, Vol. 31, pp. 965-976, 1999.

[5] Y. Narahari and L. M. Khan, "Performance analysis of scheduling policies in re-entrant manufacturing systems," *Computers & Operations Research*, Vol. 23, No. 1, pp. 37-51, 1996.

[6] H. T. Papadopoulos and C. Heavey, "Queueing theory in manufacturing systems analysis and design: A classification of models for production and transfer lines," *European Journal of Operational Research*, Vol 92, No. 1, pp. 1-27, 1996.

[7] Y. Park, S. Kim, and C.-H. Jun, "Performance analysis of re-entrant flow shop with single-job and batch machines using mean value analysis," *Production Planning & Control*, Vol. 11, No. 6, pp. 537-546, 2000.

[8] Y. Park, S. Kim, and C.-H. Jun, "Mean value analysis of re-entrant line with batch machines and multi-class jobs," *Computers & Operations Research*, Vol. 29, No. 8, pp. 1009-1024, 2002.

[9] M. Reiser and S. S. Lavenberg, "Mean value analysis of closed multichain queueing networks," *Journal of the Association for Computing Machinery*, Vol. 27, No. 2, pp. 313-322, 1980.

[10] B. Vinod and J. J. Solberg, "The optimal design of flexible manufacturing systems," *International Journal of Production Research*, Vol. 23, pp. 1141-1151, 1985.