Proceedings of the 1987 Winter Simulation Conference
A. Thesen, H. Grant, W. David Kelton (eds.)

SINGLE RUN OPTIMIZATION OF A SIMAN MODEL FOR CLOSED LOOP FLEXIBLE ASSEMBLY SYSTEMS

Rajan Suri
Ying Tat Leung
University of Wisconsin - Madison
Department of Industrial Engineering
1513 University Avenue
Madison, WI 53706, U.S.A.

ABSTRACT

We are interested in maximizing the throughput of a closed loop flexible assembly system by distributing the total cycle time across workstations in an optimal way. The algorithm we present optimizes the system in a single simulation run and is a multi-dimensional extension of the single run optimization algorithm proposed recently. A complete implementation, of both the assembly system simulation and the algorithm, is given in the SIMAN language. Numerical examples show that the algorithm converges very quickly.

1. INTRODUCTION

Flexible assembly systems (FAS's) possess important advantages such as reduced lead time, lower assembly cost, consistent product quality, and increased flexibility (e.g. Owen, 1984). However, the fixed cost of acquiring such an assembly system is often high, and thus it is important to design and operate it at or near its 'optimal' point. In early design stages, mathematical models of FAS's may be used to get 'rough' configurations (e.g. Kamath et al., 1986). However, for detailed design or operation stages, detailed mathematical modeling and hence optimization of FAS's are difficult, because of their complexity and stochastic nature. Discrete event simulation is, in almost all cases, the only way to model FAS's in detail. Refining the detailed design of a FAS thus reduces to optimizing a discrete event simulation model.

Common methods for optimization of a discrete event simulation model invariably require a large number of simulation runs to be made. These include the popular direct search methods and response surface methodology. Less widely used in the simulation community is a family of algorithms named stochastic approximation methods (Robbins and Monro, 1951; Kiefer and Wolfowitz, 1952). These methods are based on noisy observations of the response function rather than estimates of expected values, thus requiring less computational effort. Nevertheless, a large number of simulation runs are still required in the optimum search process.

In this study we apply a recently proposed stochastic optimization algorithm known as single run optimization to a specific type of FAS's, namely closed loop flexible assembly systems. Unlike most other methods, this algorithm gives an estimate of the optimum in a single simulation run. Numerical results show that this algorithm is very efficient. A second contribution of this paper is to show the source code for the implementation of perturbation analysis and the single run optimization algorithm, all in the SIMAN language, along with the SIMAN implementation of the FAS simulation model.

Section 2 gives a brief review of closed loop flexible assembly systems and states the optimization problem under study. Section 3 describes the single run optimization algorithm. Numerical examples are shown in section 4. The source code of the complete SIMAN implementation of the algorithm is contained in Appendix A.

2. CLOSED LOOP FLEXIBLE ASSEMBLY SYSTEMS (CLFAS)

2.1 A Brief Overview

A closed loop flexible assembly system is basically a number of assembly workstations connected together in a loop by an automated material handling system (see Fig. 1). The number of pallets in the system is fixed. Workpieces enter or leave the system through a load/unload station. A limited amount of buffer space is provided between adjacent stations. These systems often work at very high speeds, typically having a cycle time of less than 10 seconds. They are used mainly for high demand products but are flexible in the sense that it is easy to retool the system to produce another product (in the same product family).

We shall focus on automatic CLFAS's where there is no manual station and no rework loops. A characteristic of these systems is that the total operation time in a station consists of two parts: the deterministic cycle time which is the actual operation time and a jam clear time if the station happens to jam during that particular operation. Jams appear in a random manner and are usually cleared manually. Jam clear times are thus also random. A detailed treatment of flexible assembly systems can be found in Owen (1984) and automatic assembly systems in general are described at length in Boothroyd et al. (1982).

2.2 A Cycle Time Optimization Problem

Let M = number of stations in the CLFAS
N = number of pallets in the CLFAS
λ = throughput of the CLFAS
B_i = buffer size before station i (excluding the station capacity, which is assumed to be 1)
t_i = transport time from the previous station to station i
T_i = random total operation time at station i
θ_i = deterministic cycle time at station i

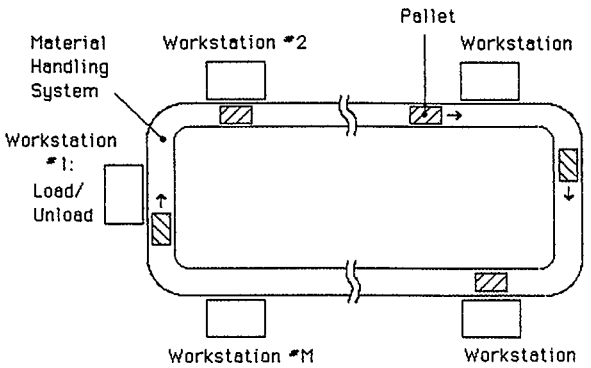

Fig. 1 Schematic Diagram of a Closed Loop Flexible Assembly System

$\theta = (\theta_1, \cdots, \theta_M)$
R_i = random jam clear time at station i

$$X_i = \begin{cases} 1 & \text{if a jam occurs at station } i \\ 0 & \text{otherwise} \end{cases}$$

$\alpha_i = EX_i$ = jam rate at station i

Then

$$T_i = \theta_i + X_i R_i \quad (i=1,\cdots,M)$$

We shall number the workstations such that station 1 is the load/unload station.

We are interested in searching for the value of θ, where throughput is maximized subject to some constraints. Formally, we have

$$\max_{\theta} \ \lambda$$

$$\text{s.t.} \ \sum_{j=1}^{M} \theta_j = \text{constant}$$

$$\theta_i \geq 0, \quad i=1,\cdots,M$$

Here M, N, B_i, t_i, α_i, distribution of R_i, $i=1,\cdots,M$, are fixed. An interpretation of the first constraint is that the total time to assemble a part is given and our aim is to distribute this total time to the (given) M stations in an optimal way. Denote the true optimum by $\theta^* = (\theta_1^*,\cdots,\theta_M^*)$.

We have made two important assumptions in this problem:

(1) The assembly task can be split in any way desired. This assumption makes the

problem continuous in the decision variable θ. In general, the assumption is, of course, not true. Permissible values of θ may be determined by technical considerations of the available workstations, the nature of the assembly task, etc.

(2) M, N, B_i, α_i, distribution of R_i, i=1,•••,M, are fixed, independent of the value of θ. In general, α_i and R_i may depend on θ, and M may depend on the way the assembly task is split. The value of M in turn affects the value of N.

Since we are dealing with a flexible assembly system, we can hypothesize that its configuration is given and we are trying to re-design it for a new product. So we can take M, N, B_i, and t_i (i=1,•••,M) as given. Still, from a practical point of view, the remaining assumptions are not realistic. However, since the single run optimization algorithm is new and not thoroughly understood at present, only if we can show that it works for simple problems can we hope that it will work for more complex real life problems. Obviously, more work has to be done before it can be applied to real life problems associated with CLFAS's.

2.3 A CLFAS Simulation Model

A simulation model of a fairly general CLFAS is written in SIMAN for the IBM PC and compatibles. The values of M, N, B_i, t_i, θ_i, α_i, and the distribution of R_i (i=1,•••,M) can be easily changed. Such a model turns out to be very short in length - about 35 lines in SIMAN, independent of all the values of the parameters described above. All standard statistics, e.g. station utilizations, queue lengths, buffer utilizations, etc., are collected.

The perturbation analysis algorithm for estimating $d\lambda/d\theta_i$ (i=1,•••,M) is then added to the simulation model (see section 3.2). With perturbation analysis, the total length of the program is about 80 lines. Finally, the single run optimization algorithm is added and the total length is about 125 lines. (Such a model for 6 workstations is given in Appendix A.)

The simulation model was verified in the following ways. By setting B_i=N, t_i=0, α_i=0, and θ_i to be exponentially distributed (i=1,•••,M), the CLFAS simulation model is reduced to a standard closed exponential network which is analytically tractable (e.g. Buzen, 1973). Hence we can compare the simulation results with the analytic solution. Next, if we allow general distribution of R_i, and allow α_i>0 (i=1,•••,M), but still assume no blocking or transportation delays, then we can compare the simulation results with that of an approximate analytical model. We chose to use the model proposed in Kamath et al. (1986). Both these methods were used to (partially) verify the simulation model.

3. SINGLE RUN OPTIMIZATION OF CLFAS

3.1 Background

As discussed above, optimization of discrete event simulation models is computationally very intensive in virtually all situations. Hoping to save some computational effort, Meketon (1983) proposed a 'single run optimization' method. The basic idea is that rather than running a whole new experiment each time after θ is updated in the search process, short iterations are made within a *single* simulation run. In this way, an estimate of the optimum can be obtained at the end of a single simulation run. Some preliminary work of applying this method to a single parameter M/M/1 queue optimization problem has been done in Suri and Zazanis (1985). Their promising results led to an extensive empirical study in Suri and Leung (1987). Results obtained there are encouraging, inspiring this study where a multiple parameter optimization problem of a more complex system, namely a CLFAS, is examined.

3.2 Perturbation Analysis of Queueing Networks

We shall use the perturbation analysis gradient estimator for estimating $d\lambda/d\theta_i$, $i=1,\cdots,M$, whose values are required in the proposed single run optimization algorithm. This perturbation analysis algorithm can be applied to general open and closed queueing networks with a single class of customers (and hence to the CLFAS's under study which are essentially closed cyclic networks). Here we will just state the algorithm; details can be found in Ho and Cao (1983) or Suri (1987).

Algorithm 1: Perturbation Analysis Algorithm for estimating $d\lambda/d\theta_i$:

(0) Initialize:
$A_{ij} \leftarrow 0$; $i,j=1,\cdots,M$ (These are the accumulator variables for the gradient calculations.)
(1) At the end of an operation at station i with total operation time T_i, $A_{ii} \leftarrow A_{ii} + dT_i/d\theta_i$, where we have used $dT_i/d\theta_i$ to denote the *sample gradient* of the random variable T_i (see the Appendix of Suri (1987)).
(2) If a pallet leaving station i going to station k terminates an idle period of station k, $A_{kj} \leftarrow A_{ij}$, $j=1,\cdots,M$
(3) If a pallet leaving station i going to station k terminates a blocked period of station i, $A_{ij} \leftarrow A_{kj}$, $j=1,\cdots,M$
(4) At the end of the simulation, let
P = total number of parts completed
S = total length of simulation in time units
λ_e = estimate of λ = P/S
Then,
$d\lambda_{ei}$ = estimate of $d\lambda/d\theta_i = -(\lambda_e/S)\cdot A_{Mi}$, $i=1,\cdots,M$
[End of algorithm]

Note that for our CLFAS system, since θ_i is a location parameter of the distribution of T_i, $dT_i/d\theta_i = 1$, $i=1,\cdots,M$ (Suri, 1987).

3.3 A Single Run Optimization Algorithm

Referring to the formal statement of the optimization problem given in section 2.2, we propose the following single run optimization algorithm. A discussion of the steps follows the algorithm.

Algorithm 2: Single run optimization algorithm for CLFAS's

(0) Choose initial values θ_i^1, $i=1,\cdots,M$
$n \leftarrow 1$
(1) Simulate at θ_i^n, $i=1,\cdots,M$, until L parts are completed. Using Algorithm 1 stated in section 3.2, we have:
Let P = total number of parts completed up to the current simulated time (= n·L)
S = current simulated time
λ_e = estimate of λ
$d\lambda_{ei}$ = estimate of $d\lambda/d\theta_i$
$\lambda_e \leftarrow P/S$
$d\lambda_{ei} \leftarrow -(\lambda_e/S)\cdot A_{Mi}$ $\quad i=1,\cdots,M$
(2) Update:

$$\theta_i^{n+1} \leftarrow \theta_i^n + a_n\cdot(d\lambda_{ei} - (1/M)\sum_{j=1}^{M} d\lambda_{ej}),$$

$i=1,\cdots,M$
where a_n is one of a sequence of positive numbers such that $\lim_{n\to\infty} a_n = 0$, $\sum_{n=1}^{\infty} a_n = \infty$, and $\sum_{n=1}^{\infty} a_n^2 < \infty$
(3) If $\theta_j^{n+1} \leq 0$ for some j, let $v \leftarrow \arg\min_j \{\theta_j^{n+1}\}$, and u be a uniform[0,1] random variable,

$$f \leftarrow \left| \frac{u * \theta_v^n}{\theta_v^{n+1} - \theta_v^n} \right|$$

Recompute θ_i's by:

$$\theta_i^{n+1} \leftarrow \theta_i^n + f\cdot a_n\cdot(d\lambda_{ei} - (1/M)\sum_{j=1}^{M} d\lambda_{ej}),$$

$i=1,\cdots,M$
(4) If a stopping criterion is satisfied, stop. Values of θ_i^n are estimates of θ_i^*, $i=1,\cdots,M$.
(5) $n \leftarrow n + 1$, Go to step (1)
[End of algorithm]

Step (2) is based on the multi-dimensional Robbins-Monro procedure for optimizing a stochastic system (Robbins and Monro, 1951; Blum, 1954), with the following modification: the terms $(d\lambda_{ei} - (1/M)\sum_{j=1}^{M} d\lambda_{ej})$ $(i=1,\bullet\bullet\bullet,M)$ are the projected gradients on the constraint hyperplane $\sum_{j=1}^{M} \theta_j$ = constant. Because of the simple nature of the constraint this allows us to adapt the unconstrained procedure in Blum (1954) for our problem.

Step (3) is a precaution against cases where some updated values θ_i^{n+1} are negative. This step is necessary since for small values of L (which we intend to use), very noisy $d\lambda_{ei}$'s may be obtained and negative θ_i^{n+1}'s values may result in step (2). In such cases we randomly shorten the step length to ensure non-negative θ_i values.

As noted in Suri and Leung (1987), by changing the values of θ_i's during a single simulation run, transient phenomena are introduced into the system. Two kinds of bias are in fact present in the perturbation analysis gradient estimator: bias due to initial transients, and bias due to the effects carried over from the old values of θ_i's; the values of λ_e and $d\lambda_{ei}$ in step (1) contain these biases. Hence the theoretical results of both perturbation analysis of queueing networks and the Robbins-Monro procedure do not hold. Thus the algorithm proposed here remains a heuristic at this stage of development.

3.4 Some Algorithm Implementation Issues

The sequence of real numbers a_n is chosen to be in the form of a harmonic sequence, i.e. $a_n = A/n$, where A is a constant. Choice of A is somewhat arbitrary; we have chosen it in such a way that the first few steps taken at the beginning of the simulation-optimization process are of the same order of magnitude of the actual values of θ_i's. A simple way to determine the value of A is to make a preliminary run of the simulation model with perturbation analysis so that the orders of magnitude of the projected gradients on the constraint hyperplane are known and hence the orders of magnitude of the step sizes taken in the first few iterations can be estimated. For the numerical examples in this study, we chose the value of A to be 800.

Following Suri and Leung (1987), the stopping criterion is chosen to be:
The algorithm stops when

$$\left[a_n \bullet \max_{i=1,\bullet\bullet,M}\{sg_i\}\right] < \epsilon,$$

where

$$sg_i = \begin{cases} (d\lambda_{ei} - (1/M)\sum_{j=1}^{M} d\lambda_{ej}) \ , \ n = 1 \\ 0.8 \bullet sg_i + 0.2 \bullet (d\lambda_{ei} - (1/M)\sum_{j=1}^{M} d\lambda_{ej}) \ , \ n > 1 \end{cases}$$

$d\lambda_{ei}$ $(i=1,\bullet\bullet\bullet,M)$ denotes its value in the n^{th} iteration,
ϵ is a constant.

This form of stopping criterion was explored and justified in Suri and Leung (1987). An intuitive interpretation of sg_i is that it is the exponentially smoothed projected gradient on the constraint hyperplane. The value of ϵ represents a tradeoff between run length and the accuracy of the estimated optimum. It is best chosen by trying out different values through a few preliminary runs with a system whose optimum is known. We used ϵ=0.005 in the subsequent numerical examples.

The value of L, the number of parts to be simulated in one iteration, represents a tradeoff between error in the estimated gradients $d\lambda_{ei}$ $(i=1,\bullet\bullet\bullet,M)$ and the frequency of iterations. Essentially, for a given run length, a small value of L will result in larger errors in $d\lambda_{ei}$ $(i=1,\bullet\bullet\bullet,M)$, but more iterations can be carried out. For the M/M/1 queue, Suri and Leung (1987) explored a range of values of L. Here we will just present results for L=20; other cases are left for a more extensive study.

3.5 Implementation in the SIMAN language

The single run optimization algorithm is implemented in SIMAN, on top of the simulation model described in section 2.3. A listing of the program for 6 workstations is given in Appendix A.

The CLFAS is modeled using a general station in SIMAN. Station 1 represents the load/unload station and station 6 is the last station visited by a part before it leaves the system. Since the number of pallets is fixed, they are all created at the very beginning of the simulation and then sent to the buffer of station 1. Buffers are modeled as resources in SIMAN. The effect of blocking due to finite buffers is implemented by seizing the following buffer before releasing a machine.

Computations for perturbation analysis and the single run optimization algorithm are also accomplished using SIMAN. Loops for calculating values of an array are done by sending an entity (i.e. a pallet) through a set of computational blocks repeatedly without advancing the simulation clock. The two dimensional array A_{ij} in Algorithm 1 is mapped into the one dimensional array $D(\cdot)$ available in SIMAN. Another way to achieve these computations would be to link the basic SIMAN simulation model to a FORTRAN sub-program.

A SIMAN counter is used to stop the simulation. When the stopping criterion is satisfied, this specific counter is incremented from zero to one.

4. NUMERICAL EXAMPLES

We now present experimental results for two CLFAS optimization problems of the type stated in section 2.2.

4.1 Test Runs on a Balanced System

We investigate the balanced case where the characteristics (e.g. B_i, θ_i, α_i, distribution of R_i, etc.) of all stations are identical. By symmetry, the maximum throughput occurs when all $\theta_i^*=\theta_j^*$ $(i,j=1,\cdots,M)$. Assuming that the optimum is not known, Algorithm 2 is used to estimate the optimum. System configuration data are: $M=6$, $B_i=B$ (see Table 1), $t_i=0$, $R_i \sim$ uniform[6,66] (in seconds), $\alpha_i=0.005$, $i=1,\cdots,M$. These values have been determined to be typical of CLFAS's (Kamath et al, 1986). We set the constraint $\sum_{i=1}^{6} \theta_i = 36$ (seconds), and hence we know *a priori* that $\theta_i^*=6$ (seconds), $i=1,\cdots,6$. We start our simulation with the initial values: $\theta = (3,6,4,8,10,5)$. Results of running our single run optimization algorithm (Algorithm 2) are shown in Tables 1 and 2.

For the two cases in the table, we display the results of five independent runs made using Algorithm 2. Run lengths are given in terms of number of completed parts for the CLFAS. Each run yields, of course, slightly different estimates of θ^*. Throughput at each estimated optimum is obtained by additional conventional simulations (i.e. without the optimization algorithm) run at the estimated θ^* value. The tables show the 95% confidence intervals of the mean throughputs, each obtained by making five additional independent replications at the estimated θ^* value. Each replication has a run length of 5000 completed parts after a 'warm-up' period of 5000 simulated seconds (consisting of about 400 completed parts). Since the width of the confidence intervals is less than 4% of the corresponding mean in all cases, the throughput means are expected to be reasonably good estimates. We shall use these means to discuss the results.

For N=6, the largest deviation of the throughput at the estimated optima from the true optimum occurs at replication 1 and is less than 6%. This replicate has increased the system throughput from 0.094 to 0.134 (parts per second) - an increase of 42%. The

Table 1: Results for single run optimization of balanced system

Case: N = 6, B = 1

Replicate number	Estimated θ^*	Run length	Thruput at estimated θ^*	Thruput % deviation from optimum
(Starting value)	3.00,6.00,4.00, 8.00,10.00,5.00	---	0.0944 ± 0.0007	33.7
1	4.91,6.09,6.34, 5.94,6.46,6.26	2380	0.1342 ± 0.0012	5.8
2	6.07,5.58,6.01, 6.41,6.18,5.75	4900	0.1353 ± 0.0011	5.0
3	5.96,5.90,6.02, 6.13,6.05,5.94	7600	0.1402 ± 0.0012	1.5
4	5.95,5.93,5.97, 6.07,6.07,6.00	8180	0.1410 ± 0.0011	1.0
5	6.22,5.97,5.67, 6.39,6.15,5.60	5340	0.1354 ± 0.0013	4.9
(True optimum)	6.00,6.00,6.00, 6.00,6.00,6.00	---	0.1424 ± 0.0010	0.0

Table 2: Results for single run optimization of balanced system

Case: N = 12, B = 2

Replicate number	Estimated θ^*	Run length	Thruput at estimated θ^*	Thruput % deviation from optimum
(Starting value)	3.00,6.00,4.00, 8.00,10.00,5.00	---	0.0968 ± 0.0005	34.2
1	6.35,6.23,5.06, 6.29,6.68,5.40	3380	0.1377 ± 0.0007	6.5
2	6.24,6.09,5.74, 5.97,6.25,5.71	5160	0.1438 ± 0.0016	2.3
3	6.11,6.19,5.83, 6.53,4.94,6.40	5060	0.1395 ± 0.0009	5.2
4	5.66,6.22,5.67, 5.89,6.30,6.26	8160	0.1425 ± 0.0007	3.2
5	6.06,5.96,6.25, 6.05,5.58,6.10	6860	0.1441 ± 0.0008	2.1
(True optimum)	6.00,6.00,6.00, 6.00,6.00,6.00	---	0.1472 ± 0.0013	0.0

other four replicates do even better. In the case of N=12, the worst case has a deviation of about 6% from the true optimum and gives an improvement of 42% over the starting point. On the other hand, the lengths of all optimization runs made in this example are less than 8500 completed parts. Considering that the run lengths of the usual simulations to get reasonable throughput estimates are over 25000 completed parts (5 replicates of about 5400 parts each), the single run optimization algorithm is seen to converge very quickly here.

Table 3: Results for single run optimization of unbalanced system

Case: N = 6, B = 1

Replicate number	Estimated θ^*	Run length	Thruput at estimated θ^*	Thruput % improvement over starting point
(Starting value)	3.00,6.00,4.00, 8.00,10.00,5.00	---	0.0944 ± 0.0007	0.0
1	5.98,6.17,6.18, 5.61,6.06,5.99	5120	0.1132 ± 0.0011	19.9
2	6.24,5.78,5.59, 6.15,6.32,5,93	4500	0.1135 ± 0.0013	20.2
3	6.45,5.77,6.03, 6.16,5.55,6.04	4520	0.1115 ± 0.0006	18.1
4	5.13,6.18,6.17, 6.09,6.28,6.15	5120	0.1127 ± 0.0015	19.4
5	6.01,5.97,5.94, 5.96,6.13,5.99	4760	0.1137 ± 0.0015	20.4

Table 4: Results for single run optimization of unbalanced system

Case: N = 12, B = 2

Replicate number	Estimated θ^*	Run length	Thruput at estimated θ^*	Thruput % improvement over starting point
(Starting value)	3.00,6.00,4.00, 8.00,10.00,5.00	---	0.0925 ± 0.0005	0.0
1	5.51,6.72,6.26, 5.88,6.68,4.95	2220	0.1194 ± 0.0016	29.1
2	6.29,5.92,6.10, 6.52,5.36,5.80	4360	0.1205 ± 0.0017	30.3
3	2.63,6.79,6.95, 5.96,6.84,6.82	5440	0.1126 ± 0.0019	21.7
4	5.90,5.18,5.95, 6.59,6.52,5.85	6020	0.1203 ± 0.0019	30.1
5	6.04,6.53,3.14, 6.72,7.05,6.53	4800	0.1174 ± 0.0011	26.9

4.2 Optimization of an Unbalanced System

In this example we explore the unbalanced case where not all stations have identical characteristics. Here the true optimum is not known analytically. All configuration data are identical to those in section 4.1 except that $\alpha_3=\alpha_6=0.03$. The starting point of the algorithm is also the same and results are given in Tables 3 and 4.

Similar to the previous example, 95% confidence intervals of the mean throughputs of all estimated optima are constructed. Since each replication of the optimization algorithm gives slightly different estimates of θ^*, by running simulations at each estimate we actually explore the neighborhood region of the estimates. For N=6, the difference between the largest (i.e. the best) and the smallest improvements in throughput obtained with the five optimization replications is about 3% of the starting value. Thus we see that there is no

large difference in throughput in the neighborhood region of the estimated optima. The smallest throughput of the 5 replicates is greater than the starting one by 18% (of the starting value). Similar remarks apply to the case N=12 (Table 4). Run lengths are all under 5500 completed parts. Again, the algorithm converges very quickly. Although we do not know precise errors of the estimated optima (without running a large number of simulations over a wide region of θ), at least it is certain that the algorithm converges to a fairly 'flat' region of throughput and the throughputs do improve significantly over the starting point.

An interesting observation from Tables 3 and 4 is that, for this system, even though the jam rates are unbalanced, the optimal cycle times do not seem to differ much from the balanced case.

5. CONCLUSIONS

We have extended the one parameter single run optimization algorithm in Suri and Leung (1987) to a multiple parameter optimization algorithm. It is applied to a cycle time optimization problem of a closed loop flexible assembly system. The optimization algorithm, including gradient estimation via perturbation analysis, is implemented in the SIMAN language. Numerical results show that the algorithm is promising, in the sense that run lengths are very short and deviations of the estimated optima from the true optimum seem to be small. Obviously, more work has to be done before the algorithm can be applied to real life problems. Some research issues include the stopping criterion, the iteration length L, convergence proof of the algorithm, and applications to more realistic CLFAS optimization problems. Nevertheless, this study represents one step forward in the area of single run optimization of discrete event stochastic systems.

REFERENCES

Blum, J.R. (1954). 'Multidimensional Stochastic Approximation Methods', *Annals of Mathematical Statistics 25*, 737-744.

Boothroyd, G., Poli, C., and Murch, L.E. (1982). *Automatic Assembly Systems*, Marcel and Dekker Inc., New York and Basel.

Buzen, J.P. (1973). 'Computational Algorithms for Closed Queueing Networks with Exponential Servers', *Communications of the Association for Computing Machinery 16*, 527-531.

Ho, Y.C. and Cao, X. (1983). 'Perturbation Analysis and Optimization of Queueing Networks', *Journal of Optimization Theory and Applications 40*, 4, 559-582.

Kamath, M., Suri, R., and Sanders, J.L. (1986). 'Analytical Performance Models for Closed Loop Flexible Assembly Systems', submitted to *International Journal of Flexible Manufacturing Systems*.

Kiefer, J. and Wolfowitz, J. (1952). 'Stochastic Estimation of the Maximum of a Regression Function', *Annals of Mathematical Statistics 23*, 462-466.

Meketon, M.S. (1983). 'A Tutorial on Optimization in Simulations', presented at the 1983 Winter Simulation Conference.

Owen, A.E. (1984). *Flexible Assembly Systems*, Plenum Press, New York and London.

Robbins, H. and Monro, S. (1951). 'A Stochastic Approximation Method', *Annals of Mathematical Statistics 22*, 400-407.

Suri, R. (1987). 'Infinitesimal Perturbation Analysis for General Discrete Event Systems', *Journal of the Association for Computing Machinery 34*, 3.

Suri, R. and Leung, Y.T. (1987). 'Single Run Optimization of Discrete Event Simulations: An Empirical Study Using the M/M/1 Queue',

Technical Report No. 87-3, Department of Industrial Engineering, University of Wisconsin - Madison.

Suri, R. and Zazanis, M.A. (1985). 'Perturbation Analysis Gives Strongly Consistent Sensitivity Estimates for the M/G/1 Queue', to appear in *Management Science*.

APPENDIX A: SIMAN SOURCE CODE FOR SINGLE RUN OPTIMIZATION OF A 6-STATION CLFAS

Model File:

```
BEGIN;
          CREATE,1:0,X(2);
          QUEUE,19;
          SEIZE:BUFFER(1);
          ROUTE:0,1;
;
          STATION,1-6;
          QUEUE,M+6;
          SEIZE:MACHINE(M);
          RELEASE:BUFFER(M);
          ASSIGN:A(2)=M;
          DELAY:P(1,A(2));
          ASSIGN:A(4)=(M-1)*X(1)+M;
          ASSIGN:D(A(4))=D(A(4))+1;
          BRANCH,1:
            WITH,P(2,A(2)),JAM:
            ELSE,NOJAM;
;
JAM       QUEUE,M+12;
          SEIZE:JAMCLEAR(M);
          DELAY:ED(1);
          RELEASE:JAMCLEAR(M);
          COUNT:2,1;
NOJAM     ASSIGN:M=M+1;
          BRANCH,1:
            IF,M.GT.X(1),RESET:
            ELSE,NORESET;
RESET     ASSIGN:M=1;
NORESET   BRANCH,1:
            IF,NR(M).EQ.0,IDLE1:
            IF,NR(M+6).EQ.MR(M+6),BLOCKED1:
            ELSE,NOMINAL;
;
IDLE1     ASSIGN:A(3)=0:NEXT(CARRYON1);
BLOCKED1  ASSIGN:A(3)=1:NEXT(CARRYON1);
NOMINAL   ASSIGN:A(3)=2;
CARRYON1  QUEUE,M;
          SEIZE:BUFFER(M);
          RELEASE:MACHINE(A(2));
          DELAY:P(3,1);
          BRANCH,1:
            IF,A(3).EQ.2,CARRYON2:
            IF,A(3).EQ.0,IDLE2:
            ELSE,BLOCKED2;
;
IDLE2     ASSIGN:A(5)=1;
LOOPID    ASSIGN:A(4)=(A(2)-1)*X(1)+A(5);
          ASSIGN:A(6)=D(A(4));
          ASSIGN:A(4)=(M-1)*X(1)+A(5);
          ASSIGN:D(A(4))=A(6);
          ASSIGN:A(5)=A(5)+1;
          BRANCH,1:
            IF,A(5).LE.X(1),LOOPID:
            ELSE,CARRYON2;
BLOCKED2  ASSIGN:A(5)=1;
LOOPBL    ASSIGN:A(4)=(M-1)*X(1)+A(5);
          ASSIGN:A(6)=D(A(4));
          ASSIGN:A(4)=(A(2)-1)*X(1)+A(5);
          ASSIGN:D(A(4))=A(6);
          ASSIGN:A(5)=A(5)+1;
          BRANCH,1:
            IF,A(5).LE.X(1),LOOPBL:
            ELSE,CARRYON2;
;
CARRYON2  BRANCH,1:
            IF,M.EQ.1,COLLECT:
            ELSE,NCOLLECT;
COLLECT   COUNT:1,1;
          ASSIGN:X(8)=X(8)+1;
          BRANCH,1:
            IF,X(8).LT.P(5,1),NCOLLECT:
            ELSE,UPDATE;
;
UPDATE    ASSIGN:X(8)=0;
          ASSIGN:X(6)=0;
          ASSIGN:X(5)=NC(1)/TNOW;
          ASSIGN:A(5)=1;
          ASSIGN:J=X(1)*X(1)-X(1)+1;
LOOPUP    ASSIGN:P(8,A(5))=-1*X(5)*D(J)/TNOW;
          ASSIGN:X(6)=X(6)+P(8,A(5));
          ASSIGN:A(5)=A(5)+1;
          ASSIGN:J=J+1;
          BRANCH,1:
            IF,A(5).LE.X(1),LOOPUP:
            ELSE,RM;
;
RM        ASSIGN:X(7)=X(7)+1;
          ASSIGN:X(6)=X(6)/X(1);
          ASSIGN:X(9)=0;
          ASSIGN:X(10)=0;
          ASSIGN:A(5)=1;
LOOPRM    ASSIGN:P(9,A(5))=P(8,A(5))-X(6);
          ASSIGN:P(10,A(5))=P(9,A(5))*P(7,1)/X(7);
          ASSIGN:P(1,A(5))=P(1,A(5))+P(10,A(5));
          BRANCH,1:
            IF,P(1,A(5)).GT.X(10),RMCONT1:
            ELSE,NEGATIVE;
NEGATIVE  ASSIGN:X(10)=P(1,A(5));
          ASSIGN:A(7)=A(5);
;
RMCONT1   BRANCH,1:
            IF,NC(1).GT.P(5,1),SMOOTH:
            ELSE,INIT;
INIT      ASSIGN:P(11,A(5))=ABS(P(9,A(5))):
            NEXT(RMCONT2);
SMOOTH    ASSIGN:P(11,A(5))=0.8*P(11,A(5))+
            0.2*ABS(P(9,A(5)));
RMCONT2   ASSIGN:X(9)=MX(X(9),P(11,A(5)));
          ASSIGN:A(5)=A(5)+1;
          BRANCH,1:
            IF,A(5).LE.X(1),LOOPRM:
            ELSE,RMCONT3;
;
RMCONT3   BRANCH,1:
            IF,X(10).EQ.0,CHECK:
            ELSE,RESCALE;
RESCALE   ASSIGN:X(11)=RA(2)*ABS((P(1,A(7))-
            P(10,A(7)))/P(10,A(7)));
          ASSIGN:A(5)=1;
LOOPRE    ASSIGN:P(1,A(5))=P(1,A(5))-
            P(10,A(5))+X(11)*P(10,A(5));
          ASSIGN:A(5)=A(5)+1;
          BRANCH,1:
            IF,A(5).LE.X(1),LOOPRE:
            ELSE,CHECK;
CHECK     BRANCH,1:
            IF,(X(9)*P(7,1)/X(7)).GT.P(6,1),
              NCOLLECT:
            ELSE,FINAL;
NCOLLECT  ROUTE:0,M;
```

```
FINAL     ASSIGN:A(5)=1;
LOOPFI    ASSIGN:S(A(5))=P(1,A(5))-
            P(7,1)*P(9,A(5))/X(7);
          ASSIGN:A(5)=A(5)+1;
          BRANCH,1:
            IF,A(5).LE.X(1),LOOPFI:
            ELSE,VERYEND;
VERYEND   DELAY:1;
          COUNT:3,1;
END;
```

Experiment File:

```
BEGIN;
PROJECT,CLFAS,SURI & LEUNG,5/26/87;
DISCRETE,50,7,19,6;
RESOURCES:1-6,MACHINE:
          7-12,BUFFER,1:
          13-18,JAMCLEAR;
DISTRIBUTIONS:1,UN(4,1);
PARAMETERS:1,3,6,4,8,10,5:
           2,0.005,0.005,0.03,0.005,0.005,
             0.03:
           3,0.0:
           4,6,66:
           5,20:
           6,0.005:
           7,800:
           8,0,0,0,0,0,0:
           9,0,0,0,0,0,0:
           10,0,0,0,0,0,0:
           11,0,0,0,0,0,0;
INITIALIZE,X(1)=6,
           X(2)=6;
CSTAT:1,S(1),MC 1 OPT SERV:
      2,S(2),MC 2 OPT SERV:
      3,S(3),MC 3 OPT SERV:
      4,S(4),MC 4 OPT SERV:
      5,S(5),MC 5 OPT SERV:
      6,S(6),MC 6 OPT SERV;
COUNTERS:1,TOTAL # OF PARTS:
         2,TOTAL # OF JAMS:
         3,CONTROL COUNTER,1;
SEEDS:1,17367,NO:2,5891,NO;
REPLICATE,1;
END;
```

AUTHORS' BIOGRAPHIES

Rajan Suri is an associate professor of Industrial Engineering at the University of Wisconsin - Madison, and also one of the faculty involved in the Manufacturing Systems Program. He specializes in modeling and decision support for manufacturing systems. He is the author of many journal publications and several books, and has chaired international conferences on this subject. He has worked with major industrial corporations and is a principal of Network Dynamics Inc., Cambridge MA, a firm specializing in software for manufacturing systems.

Rajan Suri
University of Wisconsin - Madison
Department of Industrial Engineering
1513 University Avenue
Madison, WI 53706, U.S.A.

Ying Tat Leung is a Ph.D. student in Industrial Engineering at the University of Wisconsin - Madison. He received a B.Sc.(Eng.) from University of Hong Kong (Hong Kong), and an M.S. from University of Wisconsin - Madison, both in Industrial Engineering. His current research interests include simulation optimization and modeling of manufacturing systems.

Ying Tat Leung
University of Wisconsin - Madison
Department of Industrial Engineering
1513 University Avenue
Madison, WI 53706, U.S.A.