Efficient Algorithms for Buffer Space Allocation

Stanley B. Gershwin
Massachusetts Institute of Technology
77 Massachusetts Avenue
Cambridge, Massachusetts 02139-4307
gershwin@mit.edu
http://web.mit.edu/manuf-sys/www
Telephone 617-253-2149
Fax 617-258-5972

James E. Schor
Analytics, Inc.
101 Rogers Street
Cambridge, Massachusetts 02142
analytx@ibm.net
Telephone 617-577-7302
Fax 617-577-7304

1 Introduction

1.1 Problem description

Production systems are often organized with machines or work centers connected in series and separated by buffers. This arrangement is often called a *flow line*, or *transfer line*, or *production line*. A five-machine line is represented in Figure 1, in which the squares represent machines and the circles represent buffers. Material moves in the direction of the arrows, from upstream inventory to the first machine for an operation, to the first buffer where it waits for the second machine, to the second machine, etc.

Material flow may be disrupted by machine failures or variable processing times. Buffers are inserted between machines to limit the propagation of disruptions, and this increases the average production rate of the line. Inclusion of buffers requires additional capital investment and floor space, which may be expensive. Buffering also increases in-process inventory. If the buffers are too large, the work-in-process inventory and capital costs incurred will outweigh the benefit of increased productivity. If the buffers are too small, the machines will be underutilized or demand will not be met.

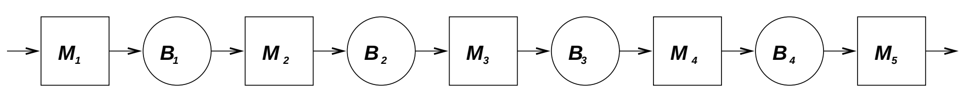

Figure 1: Flow line

Often, the limitation on the amount of in-process inventory is not due to physical constraints. Instead this limitation is a control policy, called the *kanban* policy. It is essential to determine buffer sizes to achieve the desired performance.

Much literature analyzes the effect of machine unreliability and buffer sizes on system performance. Less effort has been expended to develop methods for optimizing or improving the allocation of buffer space. The purpose of this paper is to describe efficient algorithms for determining how buffer space should be allocated. This paper is a summary of Schor (1995), in which can be found all details of algorithms and numerical results. The relevant literature is surveyed in Schor (1995) and Gershwin and Goldis (1995).

1.2 Approach and Outline

We consider systems that may be represented by a deterministic processing time model (Buzacott, 1967; Gershwin, 1994) or a continuous material flow model (Gershwin and Schick, 1980; Burman, 1995). For a discrete material, deterministic processing time system, we use an approximate decomposition method, developed by Gershwin (1987) to determine production rate and average buffer levels. The DDX algorithm, developed by Dallery, David, and Xie (1988), solves the decomposition equations efficiently. The ADDX algorithm, which we use to evaluate the performance measures for the continuous material flow model, was developed by Burman (1995).

We describe algorithms which choose buffers sizes for a flow line. They are based on the qualitative properties of the production rate described in Section 2. The number of machines and the reliability and speed parameters of each machine are assumed to be specified. Two problem formulations are presented in Section 3. The goal of the *primal* is to minimize the total buffer space required for the line to meet or exceed a given average production rate. This formulation is appropriate if either floor space or the buffering mechanism is expensive, if work in process inventory is inexpensive, and if an average production rate is mandated.

The goal of the *dual* is to maximize the production rate achievable with a given total buffer space. This is appropriate in cases where the total floor space is fixed, where the number of buffer locations is fixed, and where the problem is how to space the machines out to get the maximum benefit from the buffers. Here the dual is used in as a subproblem in solving the primal. We introduce another subproblem for the primal in Section 4.

The solution methods are summarized in Sections 5 and 6. The behavior of the algorithms is discussed in Section 7. We reformulate the problems to include inventory costs in Section 8. We conclude in Section 9.

2 Properties

In the following, N_i, the size of Buffer B_i, is the decision variable, and P is the production rate of the line. The line has k machines and $k-1$ buffers. In the discrete material formulation,

Machine	r_i	p_i
1	0.35	0.037
2	0.15	0.015
3	0.4	0.02

Table 1: Three-machine line parameters

all machines are assumed to have the same operation time, which we use as the time unit. The parameters of Machine M_i are p_i, the probability of a failure during a time unit while the machine is operating; and r_i, the probability of a repair during a time unit while the machine is down.

In the continuous material formulation, workpieces are treated like a continuous fluid. This is one way of modeling machines that have different speeds. The rate at which Machine M_i processes material, when it is operational and neither starved nor blocked is μ_i. That is, it processes $\mu_i \delta t$ material in a time interval of length δt. The reliability parameters of Machine M_i are p_i and r_i. They are probability rates: $p_i \delta t$ is the probability of a failure during a time period of length δt during which the machine is operating and neither starved nor blocked; and $r_i \delta t$ is the probability of a repair during a δt time period during which the machine is down. Details of both models can be found in Gershwin (1994).

Although the production rate is a function of machine speeds and reliabilities (and, in general, other attributes not treated here), we vary only buffer sizes, so we write $P = P(N_1, \ldots, N_{k-1})$. In the discrete material formulation, N_i is a non-negative integer (although in all our examples, it was convenient to restrict N_i to be greater than or equal to 4); in the continuous material formulation, N_i is a non-negative real number. We write $N_i \geq N_{min}$, where $N_{min} = 4$ in the discrete material case and $N_{min} = 0$ in the continuous material case.

We assume two qualitative properties:

- **Monotonicity** The production rate increases monotonically in each N_i.
- **Concavity** The production rate appears to be a concave function of the vector of N_i.

These properties are evident in Figure 2, which shows how production rate varies with buffer sizes for the three-machine line whose parameters are given in Table 1. Figure 3 shows the lines of constant P for this case. (The *optimal curve* is described in Section 4.) The literature provides proofs of these properties for similar systems.

3 Primal and Dual Problem

3.1 Primal Problem

In the primal problem, we seek the buffer sizes $(N_1, ..., N_{k-1})$ that minimize total buffer space N^{total} such that the production rate P is greater or equal to a specified value P^*. Or,

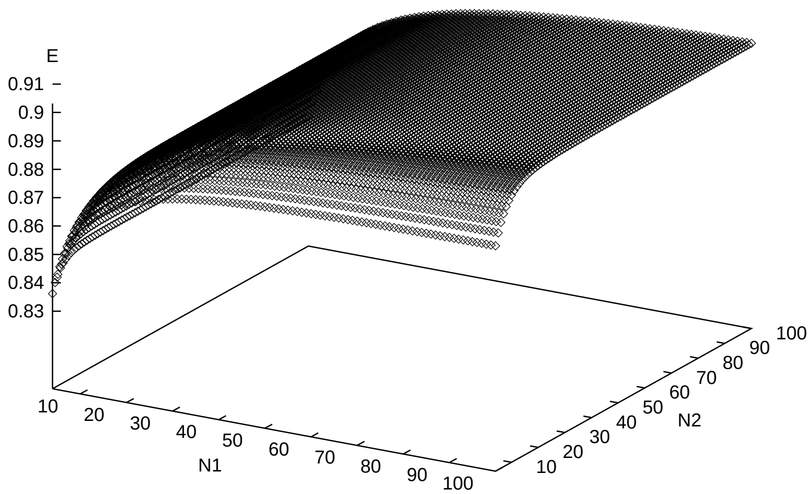

Figure 2: P vs. N_1 and N_2

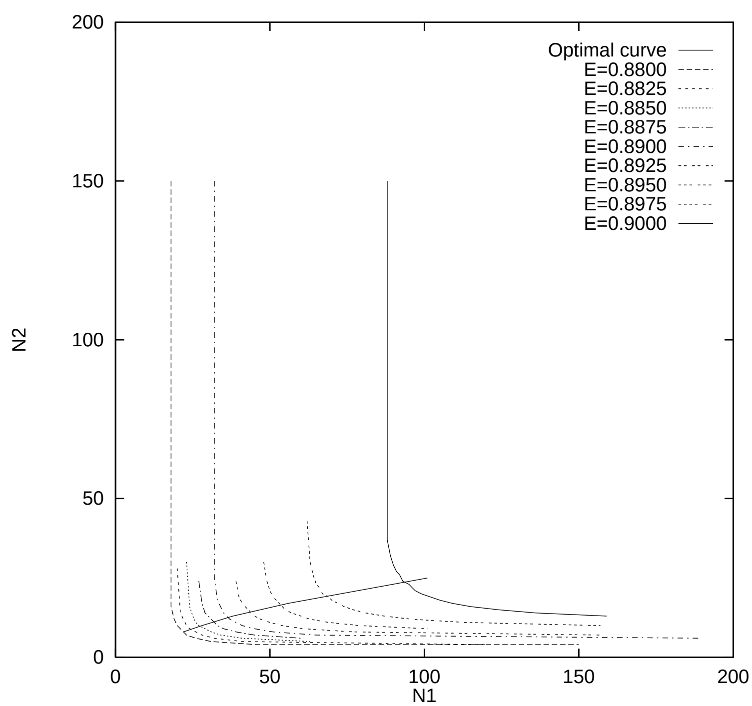

Figure 3: Iso-P lines

$$\text{Minimize } N^{total} = \sum_{i=1}^{k-1} N_i \tag{1}$$

subject to

$$P(N_1, \ldots, N_{k-1}) \geq P^*; \; P^* \text{ specified} \tag{2}$$

$$N_i \geq N_{min}$$

This problem is difficult because the constraint $P \geq P^*$ cannot be expressed in closed form. Consequently, even if we know a value of $(N_1, \ldots, N_{k-1})$ satisfying $P(N_1, \ldots, N_{k-1}) = P^*$, it is difficult to construct another $(N_1, \ldots, N_{k-1})$ that will also be on that surface.

The qualitative properties ensure that a solution will be found with constraint (2) satisfied with equality. Since the properties appear to hold strictly, there will be a unique solution.

3.2 Dual problem

In the dual problem, we seek the buffer sizes $(N_1, ..., N_{k-1})$ that maximize the production rate P such that the total buffer space is equal to a specified value N^{total}. That is,

$$\text{Maximize } P(N_1, \ldots, N_{k-1}) \tag{3}$$

subject to

$$N^{total} = \sum_{i=1}^{k-1} N_i; \; N^{total} \text{ specified} \tag{4}$$

$$N_i \geq N_{min}$$

The input to this problem is N^{total} and the outputs are the optimal P_{max} and $(N_1, \ldots, N_{k-1})$. The resulting maximum value of P is written $P_{max}(N^{total})$.

4 One-Dimensional Primal Problem (ODPP) Algorithm

Figure 4 is the graph of $P_{max}(N^{total})$ for the three-machine, two-buffer system determined in Table 1. The locus of buffer size vectors $(N_1, \ldots, N_{k-1})$ that satisfy the dual problem, for all possible values of N^{total} is the *optimal curve*. It is indicated in Figure 3 for the same production line. It is useful that this curve is nearly a straight line. We exploit this to greatly enhance the speed of the primal algorithm (Section 6).

Consider the following auxiliary problem:

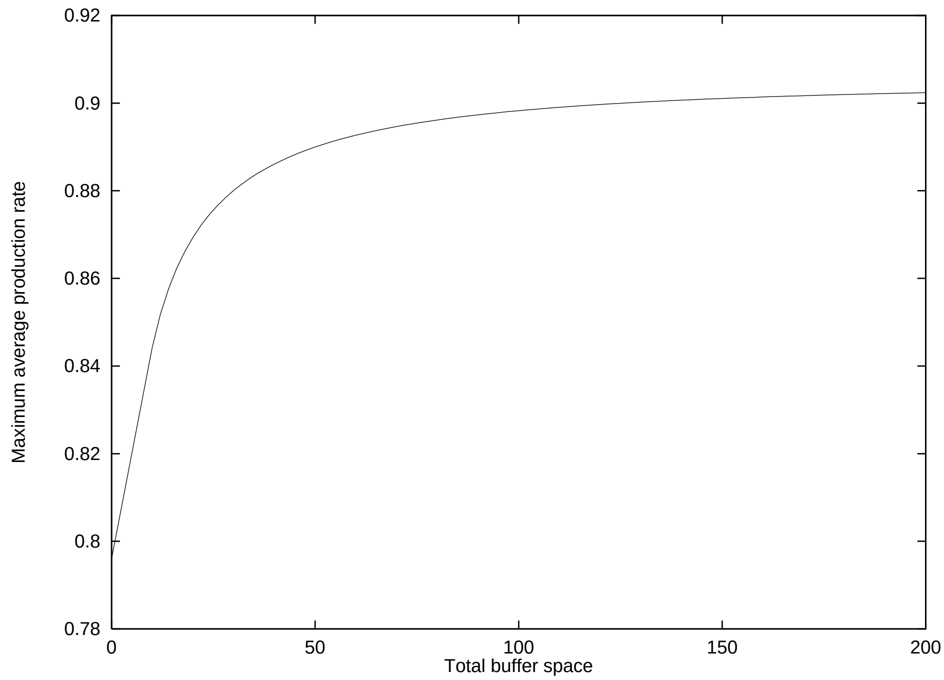

Figure 4: P_{max} vs. N^{total}

$$\text{Minimize } N^{total} = \sum_{i=1}^{k-1} N_i \tag{5}$$

subject to

$$P_{max}(N^{total}) \geq P^*; \; P^* \text{ specified}$$

$$N^{total} \geq (k-1)N_{min}$$

This problem is essentially the same as the primal problem because the same N^{total} satisfies both problem. Schor (1995) develops a one-dimensional search which is based on the property that $P_{max}(N^{total})$ is monotonically increasing in N^{total}. The algorithm requires an initial guess, N^1. We evaluate $P_{max}(N^1)$ using the dual algorithm. We also need $P_{max}(0)$, the production rate of the line when all buffers are of size zero. The evaluation of $P_{max}(0)$ is described in Section 6.2 under "Necessity." We construct a linear approximation of $P_{max}(N^{total})$, and we use this to estimate N^2, the second guess.

For $j \geq 2$, $P_{max}(N^j)$ is evaluated using the dual algorithm, j is incremented and then we iterate. Each iteration uses N^j and N^{j-1} to construct the next linear approximation of $P_{max}(N^{total})$. The process continues until $P_{max}(N^{total})$ is sufficiently close to P^*.

5 Dual Algorithm

The dual algorithm is a gradient algorithm. An initial guess is found and a direction to move in $(N_1, ...N_{k-1})$ space is determined. A linear search is then conducted in that direction until a maximum is encountered. A new direction is chosen and the process continues until no improvement is realized.

The dual problem is relatively easy to solve, and the dual algorithm is relatively easy to implement, because the constraint set (4) is a plane. As a result, large steps may be taken. As indicated above, the primal problem is *not* easy to solve in this way because its constraint is not a plane. The primal algorithm described in Section 6 uses the dual algorithm to circumvent this difficulty.

To determine the search direction, we find the gradient $\boldsymbol{g}$ according to a forward or backward difference formula. The forward difference is

$$g_i = \frac{P(N_1, \ldots, N_i + \Delta N, \ldots, N_{k-1}) - P(N_1, \ldots, N_i, \ldots, N_{k-1})}{\Delta N} \tag{6}$$

Monotonicity guarantees that $g_i \geq 0$ for all i.

We must select $\delta N_i = \hat{N}_i - N_i$ (where N_i is the most recent guess of the dual solution and $\hat{N}_i$ is the next guess) to achieve the greatest increase in P and to continue to satisfy constraint (4). We construct the search direction $\boldsymbol{p}$ by projecting $\boldsymbol{g}$ onto the hyperplane $\sum_{i=1}^{k-1} \delta N_i = 0$ as follows:

$$\bar{g} = \frac{1}{N-1} \sum_{i=1}^{k-1} g_i; \quad p_i = g_i - \bar{g} \tag{7}$$

An exception occurs when $N_q = 0$ and $p_q \leq 0$ for some q. We calculate the new direction by deleting the q_{th} component of $\boldsymbol{g}$ and setting $p_q = 0$. Then we recalculate the other components of $\boldsymbol{p}$ using (7), and the new $\boldsymbol{g}$. This process is continued until a feasible step may be taken or all components of $\boldsymbol{g}$ are deleted.

We find the scalar A such that $\hat{N}_i = N_i + \delta N_i = N_i + Ap_i$ maximizes P. Then we calculate the next gradient $\boldsymbol{p}$ and repeat. This process ends when all components of $\boldsymbol{p}$ are sufficiently small.

6 Primal Algorithm

6.1 Algorithm Structure

The reason for the algorithm structure described here is that it is difficult to construct $(N_1, ..., N_{k-1})$ that satisfies constraint (2) with equality. The goal is to avoid having to search surfaces like those of Figure 2. By using the dual algorithm as part of the solution of the primal problem, we avoid it.

We solve the primal problem by dividing it into two subproblems: the dual and the one-dimensional primal problem (ODPP). The primal problem is solved by searching the curve $P_{max}(N^{total})$ for $P_{max} = P^*$. The ODPP algorithm organizes the search. The dual algorithm is invoked to perform the evaluation step of this search.

The nearly straight optimal curve is helpful in making the algorithm efficient. Linearizing it leads to a very good initial guess of $(N_1, ..., N_{k-1})$ for the dual algorithm.

6.2 Implementation issues

There are some important issues to consider when implementing these algorithms.

Feasibility We check feasibility by requiring that the isolated production rate of each machine is greater than the production rate target. That is, in the discrete material case, a solution exists only if

$$P^* < \min_i \frac{r_i}{r_i + p_i}$$

and in the continuous material case, a solution exists only if

$$P^* < \min_i \frac{r_i \mu_i}{r_i + p_i}$$

Necessity We also want to determine if there is a need for buffers. If the production rate of the line with no buffers is greater than P^* there is no reason to run the algorithm. When the deterministic processing time model is used (Buzacott 1967),

$$P(0) = \frac{1}{1 + \sum_{i=1}^{k} \frac{p_i}{r_i}} \tag{8}$$

If the processing rates of the machines are different, we invoke the ADDX algorithm with $N_i = 0$ for all i to evaluate $P(0)$. In either case, if $P(0) \geq P^*$, then all the optimal buffer sizes are 0.

ADDX and DDX algorithm initialization These are iterative algorithms that must be initialized whenever they are invoked. Here, instead of a standard initialization, we use the final value of $(N_1, ..., N_{k-1})$ from the last evaluation. This decreases the number of iterations required for the algorithms to converge. (See Gershwin and Goldis (1995).)

Dual initialization Just as the next guess for N^{total} is obtained from locally linearizing the $P_{max}(N^{total})$ graph, a good guess for $(N_1, ..., N_{k-1})$ can be obtained from linearizing the optimal curve.

Rounding In the final step of the algorithms for continuous material lines, we round N^i so that the solution is integer. This can sometimes reduce P to below the target P^*, so some adjustment is made.

7 Algorithm Behavior

7.1 Dual

Schor (1995) observes that the solutions are either optimal or very close to optimal in all of the cases studied. The accuracy of the solution is affected by the rounding methodology. Results were compared with those of Ho, Eyler, and Chien (1979), which performs evaluation by simulation. The distribution of buffer space seemed to be quite different, but the total buffer space and the production rate were very close.

7.2 Primal

Again, Schor (1995) observes that the solutions are either optimal or very close to optimal. He also compares the algorithm with those of Park (1993) and Gershwin and Goldis (1995) and finds that the new algorithm is roughly equally accurate or more accurate and roughly equally fast or faster on all cases tried.

The algorithm takes more time for longer lines. It also takes more time as the target production rate, P^*, approaches the maximal production rate of the line, the speed of the slowest machine. See "Feasibility" in Section 6.2.

8 Profit Maximization

Here, we generalize the objective function to include inventory holding costs. Let $\bar{n}_i$ be the average amount of inventory in Buffer B_i. This quantity is an output of the DDX and ADDX algorithms. Let Φ, a_i, and b_i be non-negative cost coefficients. The profit is

$$\Pi = \Phi P(N_1, \ldots, N_{k-1}) - \sum_{i=1}^{k-1} a_i N_i - \sum_{i=1}^{k-1} b_i \bar{n}_i$$

in which Φ is the the profit per item sold, when all costs other than inventory costs at this system are considered. The cost of inventory space is linear in the size of the space, and the cost per unit space in Buffer B_i is a_i. The cost of inventory held is linear in the average amount of inventory, and the cost per item stored in Buffer B_i is b_i.

8.1 Profit Maximization With Total Buffer Space Constraint

$$\text{Maximize } \Pi \tag{9}$$

subject to

$$N^{total} = \sum_{i=1}^{k-1} N_i; \quad N_i \geq N_{min}$$

8.2 Profit Maximization Without Space Constraints

$$\text{Maximize } \Pi \tag{10}$$

subject to

$$N_i \geq N_{min}$$

8.3 Properties of Π

We can no longer rely on the objective function being monotonic or concave. However, numerical experience suggests that it has a single maximum, and this appears to be enough for the algorithms to work effectively.

Figure 5 is a graph of $\Pi(N_1, N_2)$ vs. N_1 and N_2 for the balanced three-machine continuous material line with parameters $r_i = 0.1$, $p_i = 0.01$, and $\mu_i = 1$ for $i = 1, 2, 3$. The cost coefficients are $\Phi = 1000$, $a_i = 0$, and $b_i = 1$ for $i = 1, 2$. Therefore $\Pi(N_1, N_2) = 1000P(N_1, N_2) - \bar{n}_1 - \bar{n}_2$. The contour lines indicate that the function $\Pi(N_1, N_2)$ has a global maximum and no other local maxima. The maximum profit for this line is realized when $(N_1, N_2) = (21, 49)$. (These have been rounded to integers.) Note that Figure 5 is not symmetric and the buffer sizes are not equal, even though the line is balanced.

8.4 Profit Maximization Algorithms

The algorithm for the constrained profit maximization problem (9) is essentially the same as the algorithm for the dual problem. The algorithm for the unconstrained profit maximization problem (10) follows the primal algorithm very closely. It also consists of the iterated solution of two problems: the constrained profit maximization problem (in place of the dual) and a one-dimensional profit maximization problem (in place of the one-dimensional primal).

9 Conclusions

The algorithms described here are fast and accurate. They can be used in the design of production systems. These methods can easily be extended to assembly/disassembly systems (Gershwin, 1994). It would be valuable to extend them to systems that have loops, such

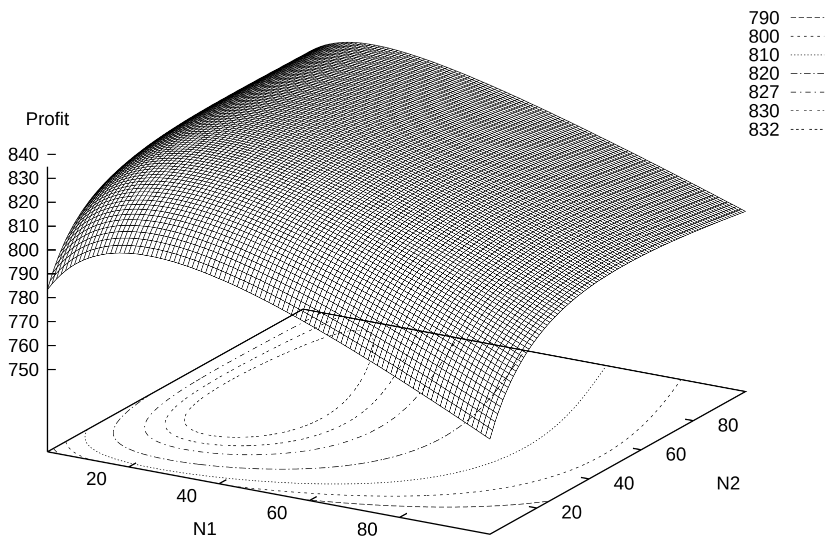

Figure 5: Profit vs. N_1 and N_2

as systems with captive pallets or fixtures. Other useful extensions include systems with multiple part types, with set-up changes, etc. The main challenges in such extensions are more in the modeling and analysis than in the optimization algorithm.

These extensions are valuable for designing more general systems. As shown in Gershwin (1997) they are also valuable in designing real-time scheduling policies for manufacturing systems.

References

M. Burman (1995), "New Results in Flow Line Analysis", Ph. D. Thesis, MIT, Cambridge MA, 1995.

J. A. Buzacott (1967), "Automatic Transfer Lines with Buffer Stocks," *International Journal of Production Research*, Vol. 5, No. 3, pp. 182-200.

Y. Dallery, R. David, and X.L. Xie (1988), "An Efficient Algorithm for Analysis of Transfer Lines with Unreliable Machines and Finite Buffers," *IIE Transactions,* Vol. 20, No. 3, pp. 280-283, September, 1988.

S. B. Gershwin (1987), "An Efficient Decomposition Algorithm for The Approximate Evaluation of Tandem Queues with Finite Storage Space and Blocking," *Operations Research*, Vol. 35, pp. 291-305.

S. B. Gershwin (1994), *Manufacturing Systems Engineering,* Prentice-Hall, 1994.

S. B. Gershwin (1997), "Design and Operation of Manufacturing Systems — Control- and System-Theoretical Models and Issue," *1997 American Control Conference*, Albuquerque, New Mexico, June 4 – 6, 1997.

S. B. Gershwin and Y. Goldis (1995), "Efficient Algorithms for Transfer Line Design", MIT Laboratory for Manufacturing and Productivity Report LMP-95-005, 1995.

S. B. Gershwin and I. C. Schick (1980), "Continuous Model of an Unreliable Material Flow Systems with a Finite Interstage Buffer," MIT LIDS Report LIDS-R-1039, September, 1980.

Y. C. Ho, M. A. Eyler, and T. T. Chien (1979), "A Gradient Technique for General Buffer Storage Design in a Production Line," *International Journal of Production Research*, Vol. 17, No. 6, pp. 557-580.

T. Park (1993), "A Two-Phase Heuristic Algorithm for Determining Buffer Sizes of Production Lines," *International Journal of Production Research*, Vol. 31, No. 3, pp. 613-631, 1993.

J. E. Schor (1995) "Efficient Algorithms for Buffer Allocation", M. S. Thesis, MIT, Cambridge MA, 1995.