Optimal Buffer Allocation in Production Lines Using an Automata Search

Tolga Tezcan

Abhijit Gosavi
gosavi@uscolo.edu
2200 Bonforte Blvd, 261 Tech Building
University of Southern Colorado, Pueblo, CO 81001

Abstract

In this paper, we use a learning automata search technique (LAST), which is based on game theory, to solve the optimal buffer allocation problem in production lines. We have incorporated the search algorithm within a simulator in order to optimize the performance of a production line. We have conducted experiments with this algorithm on some problems for which optimal solutions are available in the literature. We find encouraging preliminary results; the algorithm finds near optimal solutions in a relatively small number of iterations.

Keywords: Production Lines, Learning Automata, and Simulation Based Optimization

1. Introduction

The theory of learning automata (LA) can be used for solving combinatorial optimization problems. In this paper, we use an LA based search technique for finding the optimal buffer sizes in a stochastic production line. We have some encouraging preliminary results. To the best of our knowledge, this is the first use of an automata search on the buffer allocation problem.

A production line is a series of work centers (made up of one or more machines), which are connected linearly and are separated with buffers. Each work center consists of one or more identical machines in parallel. An example of a production line is shown in Figure 1, in which circles represent buffers and squares represent work centers. In flow lines, every item visits each workstation in a fixed order, enters the system from the first machine, and leaves the system from the last machine. Production lines are generally used in high volume manufacturing, e.g. automobile manufacturing. (Gershwin [6])

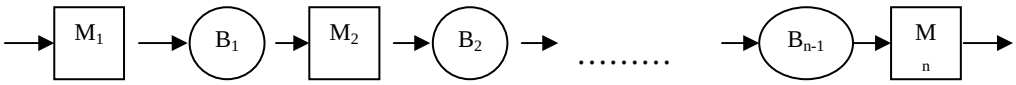

Figure 1: *n* machine flow line

Stochastic production lines are subject to disturbances arising from variations in processing times and failures of the workstations involved. This can cause the machines to be idle and can lower the throughput of the line. In order to mitigate the effect of this disturbance, in between machines, buffers are used. The throughput of a stochastic production line can be increased to a certain extent by allocating more buffer space between the machines. However there is a limit to the amount of spaces allocated to buffers, because holding inventory costs money. Also space and

material handling equipment are required to maintain work-in-process (WIP) (Gershwin [6]). Thus the problem of buffer space allocation becomes a stochastic optimization problem with several variables involved. The basic idea of Kanban control is rooted in this optimization problem.

In the 1980s and before, most of the research related to production lines was concentrated on predicting the performance of these systems with a given configuration rather than on designing them. After the 1980s, a number of papers on the design of these systems, i.e., allocating the buffers between machines with some objective, have appeared. An extensive literature review can be found in Altiok [1], Gershwin [6], Gershwin and Schor [7] among other papers and books. We, next, review simulation-based approaches for the design of flow lines.

Conway *et al.* [4] did research on determining where in the production line WIP is most effective and on measuring the benefit of WIP as a function of quantity. They analyzed balanced lines in general, balanced lines with unequal variability, unbalanced lines, unreliable stations, and came up with design rules. Martin [11] compared some of the previous simulation-based methods for predicting system delay, found the estimate of Anderson and Moodie [3] to be a reasonable one, and refined the equation of Anderson and Moodie. Then Martin [12] used it to find the optimal buffer capacities. Liu and Lin [10] used simulation to come up with predictive functions for the throughput and the coefficient of variation of a two-machine line without breakdowns. Then they applied an aggregation method to estimate the throughput of longer lines, and used dynamic programming to determine the minimal buffer spaces for a desired throughput. Powell [14] used simulation to find the optimal allocation of buffers in order to optimize the throughput. He did not include breakdowns in his system, focused on three-station lines, and developed some rules of thumb for allocating buffers. Ho *et al.* [8][9] combined simulation with a gradient-based approach. They assumed identical deterministic processing times and geometrically distributed repair and failure times in their experiments.

Learning Automata (LA) is a stochastic search technique, originally proposed to model learning behavior in biological systems. Narendra and Thatachar [13] present an introduction to LA. Thatachar and Sastry [17] introduced a method for learning optimal discriminant functions through the model of a cooperative game of learning automata. This method has been used for solving machine vision problems. (Sarkar and Chavali [15])

2. Problem Description

The production line in our model consists of n machines in series and (n-1) buffers located between consecutive machine pairs. The parts begin their processing in the first machine, visit each machine in the line in a fixed order and leave the system from the nth machine.

Each machine is characterized by three parameters: the processing time, the time between failures and the time to repair. In a large number of papers, processing times are assumed to be deterministic or random variables distributed exponentially. Deterministic processing times are typical of highly automated systems (Gershwin [6]). Exponential processing times are assumed for ease of modeling especially in theoretical models for the stochastic case. Most of the literature assumes the time between failures and the repair times to be geometrically or exponentially distributed. In some of our experiments, we have assumed the exponential distribution for processing times, for time between failures and for repair times to compare our results with existing results. But since our approach is simulation based, we can easily relax this assumption. Next, we state some assumptions made in a general description of buffer allocation problems.

A machine processes jobs when it is operational and it can find a job in its buffer. Whenever a production is completed, the part just produced is transferred to the buffer of the next machine unless the next buffer is full. If the next buffer is full and the part cannot be transferred, the machine is said to be *blocked* and the part stays on the machine till the next buffer generates space. If a machine is available and operational, but cannot find a job in its buffer, it is said to be *starved*. A machine is said to have *failed*, if it is under repair. A machine cannot process jobs while it is under repair.

There are some assumptions associated with the simulation model that we have used for our experiments. These assumptions can be easily justified from realistic considerations. Firstly, we assume that first machine never starves; in other words, there is an infinite supply of raw material for the first machine. Secondly, we assume that the last machine is never blocked, that is, there is an infinite buffer after the last machine. Besides we assume that repair starts as soon as a machine fails and the job being processed remains on that machine till the machine is fixed. The

process resumes after repair. In the literature, two types of production systems have been discussed: continuous and discrete. We focus on discrete production systems in this paper.

There are several different objectives studied in the design of flow lines in the literature. Some of these are: long run average profit per unit time (Altiok and Stidham [2]), total buffer space to achieve a given production rate (Yamashita and Altiok [18]) and the throughput under a linear resource constraint (Seong et al [16]). In this paper, we focus on maximizing the long-run average profit per unit time. A formal problem statement follows:

$$\text{Maximize } f\left(B_1, B_2, ..., B_{n-1}\right) = \pi T\left(B_1, B_2, ..., B_{n-1}\right) - h\bar{B} \quad (1)$$

where:

$T\left(B_1, B_2, ..., B_{n-1}\right)$ is the long - run throughput of the system when the buffer limit of the ith buffer is B_i

π is the profit per part produced,

h is the WIP holding cost per part per unit time, and

$\bar{B}$ is the long - run average number of units in the system.

3. Solution Approach

We have used a relatively new stochastic search technique, called the learning automata search technique (LAST), which is based on the theory of learning automata and the theory of games. LAST uses a stochastic search to explore the solution space of a combinatorial optimization problem, such as the one considered in this paper. The automata theory literature has some of its own notation, which can be explained in terms of combinatorial optimization. Next, we shall explain this notation.

With each parameter (of a combinatorial optimization problem) or player, we associate a so-called *learning automaton* and with each value that a parameter can assume we associate an *action.* So any given solution to the problem can be thought of as a collection of actions selected by the team of learning automata. This collection of actions is called a *policy*. The value of the objective function associated with a solution is also called "*feedback*" from the "*environment*". The environment is usually the objective function evaluator, which in our case, is a simulator.

We shall begin with a description of the algorithm for *one parameter*. In a nutshell, the behavior of the algorithm for this case can be described as follows. The algorithm starts with a prior probability distribution over its action set (i.e. the set of values that the parameter can assume). For example, if there are 3 values (1, 5, 6) for the parameter *x*, we begin with a distribution defined by $P(x = 1) = P(x = 5) = P(x = 6) = \frac{1}{3}$. An action is chosen with these probabilities and the function is evaluated at that point. The feedback for this selected action, that is, the value of the function is used to update the probability of selecting that action. If the feedback is favorable, the probability is increased else it is reduced. See Figure 2 for a schematic showing the learning mechanism.

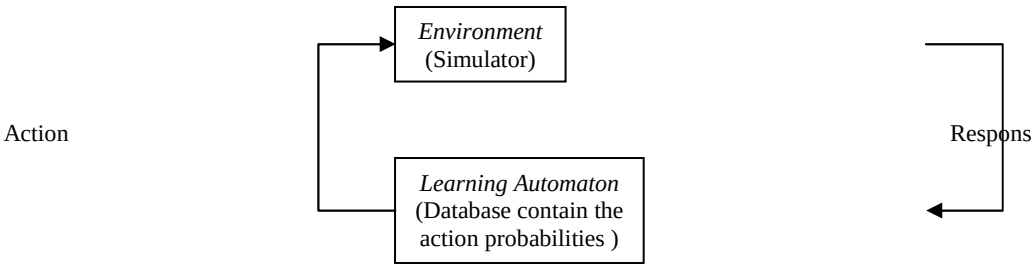

Figure 2: Learning Mechanism of LAST

In the *multiple parameter* case, each automaton maintains its own probability distribution. In every iteration of the algorithm, each automaton selects an action and this collection of actions (policy) is fed into the environment. The resulting feedback is used to update the probability distribution of each automaton. In the next section, we give a detailed description of the algorithm.

3.1. Algorithm Description

In the buffer allocation problem we consider each buffer to be a player and the reward (feedback) to be the average profit in the long-run. Each buffer selects an action and determines its allocated buffer space in order to maximize the profit. The reward, which is the response of the production line, is computed via simulation with a selected buffer configuration (policy).

Let $p(i,a)$ denote the probability of selecting action a for the ith parameter. Clearly $\sum_{a\in \mathrm{A}(i)} p(i,a) = 1$, where $\mathrm{A}(i)$ denotes the finite set of actions associated with the ith parameter. At the beginning of the search, we assign an equal probability to all available actions of a parameter, that is, we set

$$p(i,a) = \frac{1}{|A(i)|}, \qquad (2)$$

where $|A(i)|$ is the cardinality of the set $A(i)$.

Now, armed with these probabilities, each automaton selects an action. The result of each policy is stored in a so-called result matrix (G). The policy is fed into the simulator and the feedback is used to update the action probabilities. The updating mechanism utilizes the results (payoffs) of previous polices in some sense. The result matrix is converted into a so-called feedback matrix (F), the terms of which are essentially normalized versions of terms in the result matrix (G). To normalize, we use estimates for the best payoff (G_{max}) and the worst payoff (G_{min}), which are found by some initial experimentation. We normalize the kth element of G using:

$$F(k) = \frac{(G(k) - G_{\min})}{(G_{\max} - G_{\min})} \qquad \text{for all } k. \qquad (3)$$

We now explain the notation that will be used in the algorithm statement. The term $p^n(i,a)$ will denote the probability of selecting action a for parameter i in the nth iteration. And μ is a learning-rate or step size used to update the probabilities. $B(i,a)$ will denote the maximum feedback associated with action a in parameter i based on the information gathered in the learning process. The step-by-step details of the algorithm are presented below.

LAST

1. Set iteration n to 0. Set $p^n(i,a) = \frac{1}{|A(i)|}$ for $i = 1,2,...,N$ and $a \in A(i)$. Initialize each element of matrix $B(i,a)$ to 0. Assign appropriate values to $\mu, G_{\max}, G_{\min}, n_{\max}$, and P_{Best}. P_{Best} denotes the best payoff so far and should equal a small quantity. The maximum number of iterations for which the algorithm will be repeated is denoted by n_{max}.

2. For each player (i), select an action $u(i)$ from $A(i)$ with probability $p^n(i,u(i))$.

3. Select the policy u. Let the payoff be P. Normalize the payoff using $D = \frac{(P - G_{\min})}{(G_{\max} - G_{\min})}$. If $P > P_{BEST}$, set $u_{BEST}(i) = u(i)$ for all i.

4. Set $i = 1$.

5. For each value of j from 1 to $|A(i)|$ do:

If $B(i, j) < B(i, u(i))$,

$$\text{Set } p^{n+1}(i, j) \leftarrow p^{n}(i, j) - \frac{\mu[B(i, u(i)) - B(i, j)]p^{n}(i, j)p^{n}(i, u(i))}{[|A(i)| - 1]}$$

If $B(i, j) \geq B(i, u(i))$,

$$\text{Set } p^{n+1}(i, j) \leftarrow p^{n}(i, j) + \frac{\mu[B(i, u(i)) - B(i, j)](1 - p^{n}(i, j))p^{n}(i, u(i))}{[|A(i)| - 1]}$$

6. Set $c = i$ and $p^{n+1}(i, u(i)) \leftarrow 1 - \sum_{j \neq u(i)} p^{n+1}(i, j)$.
   If $c < N$, set $i = c + 1$ and go to step 5. Else increment n by 1 and got to step 7.
7. If $n < n_{\max}$ go to step 8 else terminate learning with u_{best} as solution.
8. Update the B matrix. Set $i = 0$.
   8a. Increment i by 1.
   8b. If $D \leq B(i, u(i))$, go to step 8c. Else set $B(i, u(i)) = D$ and then go to step 8c.
   8c. If $i < N$ go to step 8a, else set $i = 0$ and go to step 2.

4. Preliminary Results

In this paper, some experiments are carried out on problem caes drawn from the literature in order to compare our results with optimal solutions. We use the production line studied by Altiok and Stidham [2]. They modeled the system using Markov chain theory to come up with an analytically tractable form for the objective function (see equation (1)) and used the search procedure of Hookes and Jeeves [4] to find the optimal solution. Our objective function is same as theirs, which is maximizing the long-run average profit per unit time but we use simulation to evaluate it. They gave the optimal solutions to a 3-machine 2-buffer production line with exponential processing times, repair times and time between failures. The parameters of the machines are given in Table 1.

Table 1: Parameters for the experimented production line

	Prod. Rate (Product/unit time)	Repair Rate (Repairs/unit time)	Failure Rate (Failures/unit time)
Station #	μ	λ	ς
1	0.25	0.1	0.01
2	0.2	0.3	0.02
3	0.3	0.5	0.04

In our experiments, we simulate the system for 100,000 time units, for 5 replications. More than 10,000 parts are produced in each simulation replication. In these experiments, we let $A(i) = \{n : n \in J \text{ and } n \leq 15\}$ (where J is the set of natural numbers) for $i = 1$ and 2 (i is the index for the buffer) and we run the algorithm with 50 iterations.

We perform experiments on three problem cases in which the parameters of the machines are same but the profit per part and the holding cost are altered. The results of Altiok and Stidham [2] and LAST can be found in Table 2.

Table 2: Results of Proposed Algorithm and A&S

		Altiok & Stidham [2]		LAST	
Profit per part	h	Buffer Allocation	Profit/Unit Time	Buffer Allocation	Profit/Unit Time
30	0.5	(1,5)	1.92	(1,4)	1.9079

30	0.2	(3,8)	2.5164	(3,6)	2.5144
14	0.2	(2,11)	1.6212	(2,10)	1.6113

The results from the proposed algorithm are very close to the optimal solutions. It finds the optimal allocation for the first buffer in all the three cases and the solutions for the second buffer diverge from the optimal by at most 2.

5. Conclusions

The results presented here are a part of on-going research (Tezcan [17]). We used a stochastic search technique that is based on the theory of learning automata and the theory of games in order to solve the optimal buffer allocation problem in production lines. The algorithm consists of two basic components: 1) a team of automata (buffers) which tries to find the optimal policy (buffer allocations) to maximize the long-run average profit, and 2) the environment (production line simulator) which associates an objective function value with each policy.

We compare the results from our algorithm to some optimal solutions given for a small case problem. We found encouraging results; our algorithm found near optimal solutions in a relatively small number of iterations.

The main contribution of this paper is: the use of a new algorithm on the buffer allocation problem. We use a simulation-based approach. This allows us to study any line configuration, which is hard (or sometimes even impossible) to analyze using theoretical approaches (such as Markov chains) and allows us to study cases in which the random variables that govern the behavior of the system are characterized by any general distribution.

Currently we are extending our experiments to longer production lines with any given distribution. By doing so, we will be able to analyze the convergence and convergence rate of this proposed algorithm numerically. Analytical convergence analysis of the algorithm will also be studied.

References:

1. Altiok, T., 1996, *Performance Analysis of Manufacturing Systems*, Springer, New York.
2. Altiok, T. and. S. Stidham, Jr., 1984, "The Allocation of Interstage Buffer Capacities in Production Lines," *IIE Transactions*, 15(4), 292-299.
3. Anderson, D.R. and C.L Moodie, 1969, "Optimal Buffer Storage Capacity in Production Line Systems," *International Journal of Production Research*, 7(3), 233-240.
4. Bazaraa, M. S. and C.M. Shetty, 1977, *Nonlinear Programming*, John Wiley and Sons, New York.
5. Conway, R., W. Maxwell, J.O. McClain, and L.J. Thomas, "The role of work-in-process inventory in serial production lines," *Operations Research*, 36(2) (1988), 229-241.
6. Gershwin, S.B., 1994, *Manufacturing Systems engineering*, Prentice-Hall, Englewood Cliffs, New Jersey.
7. Gershwin, S.B., N.E. Schor, 2000, "Efficient Algorithms for Buffer Space Allocation," *Annals of Operations Research*, 93, 117-144.
8. Ho, Y.C., M.A. Eyler, and T.T. Chien, 1994, "A Gradient Technique for General Buffer Storage Design in a Production Line," *International Journal of Production Research*, 32(5), 989-1000.
9. Ho, Y.C., M.A. Eyler, and T.T. Chien, 1983, "A New Approach to Determine Parameter Sensitivities of Transfer Lines," *Management Science*, 29 (6), 700-714.
10. Liu, C.M. and C.L., Lin, 1994, "Performance Evaluation of Unbalanced Serial Production Lines," *International Journal of Production Research*, 32 (12), 2897-2914.
11. Martin, G.E., 1993, "Predictive Formulae for Unpaced Line Efficiency," *International Journal of Production Research*, 31 (8), 1981-1990.
12. Martin, G.E., 1994, "Optimal Design of Production Lines," *International Journal of Production Research*, 32 (5), 989-1000.
13. Narendra, K. and M.A.L. Thathachar, 1989, *Learning Automata: An Introduction*, Prentice-Hall, Englewood Cliffs, New Jersey
14. Powell, S.G., 1994, "Buffer Allocation in Unbalanced Three-station Serial Lines," *International Journal of Production Research*, 32 (9), 2201-2217.

15. Sarkar, S. and S. Chavali, 2000, “Modeling Parameter Space Behavior of Vision Systems Using Bayesian Networks,” *Computer Vision and Image Understanding* 79, 185-223
16. Seong, D., S.Y. Chang, and Y. Hong, 1995, “Heuristic Algorithms for Buffer Allocation in a Production Line with Unreliable Machines,” *International Journal of Production Research*, 33 (7), 1989-2005.
17. Thathachar, M.A.L. and P.S. Sastry, 1987, “Learning Optimal Discriminant Functions Through a Cooperative Game of Automata,” *IEEE Transactions on Systems, Man, and Cybernetics*, 17 (1), 73-85.
18. Yamashita, H.,and T. Altiok, , 1998, “Buffer Capacity Allocation for a Desired Throughput in Production Lines,” *IIE Transactions*, 30, 883-891.