

An Efficient Buffer Design Algorithm for Production Line Profit Maximization

Chuan Shi and Stanley B. Gershwin

Department of Mechanical Engineering
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139-4307 USA

August 5, 2008

Abstract

In this paper, we present an accurate and fast algorithm for maximizing profits through buffer size optimization for production lines. We consider both buffer space cost and average inventory cost with distinct cost coefficients for different buffers, and we include a nonlinear production rate constraint. To solve the problem, a corresponding unconstrained problem is introduced and a nonlinear programming approach is adopted. Numerical results are provided to show the efficiency and accuracy of our algorithm for both short and long lines.

1 Introduction

1.1 Problem

A manufacturing system is a set of machines, transportation elements, computers, storage buffers, and other items that are used together for manufacturing (Gershwin 1994). A production line, or flow line, or transfer line, is organized with machines connected in series and separated by buffers. Figure 1, for instance, is a six-machine line. Squares represent machines or sequences of machines without buffers while circles represent buffers. (In the following, we treat a sequence of machines without buffers as a single machine.) As it is shown in Figure 1, material flows 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. It is desirable to find ways to optimize buffer space allocation to make factories most efficient and most profitable. Tools for rapid design of production lines are especially important for products with short life cycles.

Generally, designers of such production lines want to either optimize the production rate, or the profit, of the line. However, material flow may be disrupted by machine failures. The inclusion of buffers increases the average production rate of the line by limiting the propagation of disruptions,

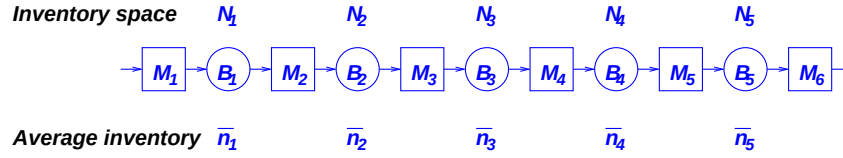


Figure 1: A production line example

but at the cost of additional capital investment, floor space of the line, and inventory. In this paper, we assume that the manufacturing process and machines have already been chosen. Therefore, the decision variables are sizes of buffer spaces. Production line cost comes from buffer space cost and average inventory cost.

In this paper, aiming at maximizing profits under a production rate constraint for production lines, we present an accurate, fast, and reliable algorithm for optimizing buffer space through a nonlinear programming approach. In our objective function, we consider both buffer space cost and average inventory cost.

1.2 Literature review

Substantial research has been conducted in production line evaluation and optimization (Dallery and Gershwin 1992). However, there are many studies focusing on maximizing the production rate but fewer studies concentrating on maximizing the profit. Basically, in production line optimization, there are two distinct approaches: the simulation based approach and the numerical evaluation approach. It is more desirable to propose methods of numerical evaluation approach in production line design since they are much faster than simulation. For a brief introduction on simulation methods, see Gershwin and Schor (2000). Here we describe some literature on non-simulation methods.

As early as 1967, Buzacott derived the analytic formula for the production rate for two-machine, one-buffer lines in deterministic processing time models (Buzacott 1967). The invention of decomposition methods with unreliable machines and finite buffers (Gershwin 1987) and its corresponding DDX algorithm (Dallery, David, and Xie 1988) enabled the numerical evaluation of the production rate of lines having more than two machines. The decomposition method permits the development of long line optimization because we cannot practically analytically evaluate lines that have more than two machines¹. Park (1993) developed a two-phase heuristic algorithm to solve the total buffer space minimization problem. But his method could only guarantee to find near-optimal solutions and did not always converge. Schor (1995), and Gershwin and Schor (2000) presented an efficient buffer allocation algorithm that applied a primal-dual approach to minimize the total buffer space under a production rate constraint, as their major problem. They also studied on the profit maximization of a line, as their minor problem, through a nonlinear programming method that was fast and accurate, but they did not consider the production rate constraint in the minor problem.

More recently, Huang, Chang, and Chou (2002) considered a flow-shop-type production system

¹Gershwin and Schick (1983) derived an analytical solution for a three-machine line with unreliable machines and small buffers. There have numerical methods for exact analysis of lines that are slightly longer with small buffers.

and used a dynamic programming approach to maximize its production rate or minimize work-in-process under a certain buffer allocation strategy. Diamantidis and Papadopoulos (2004) also presented a dynamic programming algorithm for optimizing buffer allocation based on the aggregation method given by Lim, Meerkov, and Top (1990). Although their dynamic programming methodology brought new approaches to production line design, they did not attempt to maximize the profits for lines. Chan and Ng (2002) compared four buffer allocation strategies and presented a modified one for production rate maximization. Shi and Men (2003) introduced a hybrid algorithm based on hybrid nested partitions and a Tabu search method for production line optimization. However, they also focused on maximizing the production rate of the line under a total buffer space constraint, rather than the profit of the line. Smith and Cruz (2005) solved the buffer allocation problem for general finite buffer queueing networks in which they minimize buffer space cost under the production rate constraint, but they did not consider the average inventory cost. One paper that considers both buffer space cost and average inventory cost is Dolgui, Ereemeev, Kolokolov, and Sigaev (2002). Their buffer allocation problem consists in determining buffer capacities considering the production rate of the line, the buffer acquisition and installation cost, and the inventory cost. For that problem, they proposed a genetic algorithm where tentative solutions were evaluated with an approximate method based on the Markov-model aggregation approach. They did not have the production rate constraint in their problem. Some practical considerations in optimization of flow production systems were reported in Tempelmeier (2003).

It becomes much harder if the production rate constraint is considered in production line design because the production rate is a nonlinear function of buffer sizes. Our problem includes the production rate constraint and aims at maximizing the profit for the line. As introduced in Section 1.1, we consider both buffer space cost and average inventory cost in our objective function, and assign distinct cost coefficients to different buffers. The average inventory of the line, and consequently the line's cost, are also nonlinear functions of buffer sizes. Hence, we have nonlinear elements in both our objective function and constraints.

One possible way of solving it is to conduct a P^* surface search². However, since the P^* surface is boundless, it is only possible to search on a portion of the surface. However, this method tells nothing about the effect each buffer acts on the production line behavior. It is therefore difficult to decide which portion of the P^* surface to search. Consequently, this method, though it could offer an accurate optimal solution after an extensive search, is very time-consuming.

1.3 Outline of paper

The paper is organized as follows. The model we study in our research and important features of its behavior are described in Section 2. The proposed algorithm is derived in Section 3, following by numerical results showing its accuracy and efficiency in Section 4. We summarize the paper and propose future research in Section 5. In addition, we provide the continuous variable version of the solution of two-machine line evaluation in Appendix A, the proofs about our algorithm for a special case in Appendix B, and a brief introduction to the P^* surface search in Appendix C.

²We touch this concept again in Section 4 and provide a brief introduction to this in Appendix C.

2 Problem statement, assumptions, and notation

2.1 Model of the Line

The model described here is the deterministic processing time model of Gershwin (1987), (1994). We make all the assumptions and approximations of that model, follow all his conventions, and use his notation. We outline the key features of the model below.

In our model, we denote machine i by M_i and buffer i by B_i . The line consisting of k machines and $k - 1$ buffers is called a k -machine, $k - 1$ -buffer line, or k -machine line for short. Processing times of all machines are equal, deterministic, and constant. Time is scaled so that operations take one time unit. We further assume that all the machines start their operations at the same instant. Transportation time is negligible compared to the operation time.

In addition, N_i , the size of buffer B_i , $\forall i = 1, \dots, k - 1$, are decision variables. Therefore, there are $k - 1$ decision variables for a k -machine, $k - 1$ -buffer line. Machines are unreliable and are parameterized by probabilities of failure and repair. Specifically, 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. As a consequence, the times to failure and to repair are geometrically distributed. By convention, repairs and failures occur at the beginnings of time units and changes in the buffer levels take place at the ends of time units. Machine parameters are fixed. We let P be the production rate of a line. Although the production rate P is a function of machines and their reliability, we vary only buffer sizes, so we write $P = P(N_1, \dots, N_{k-1})$, or $P(\mathbf{N})$ for short, where $\mathbf{N}$ is the vector $(N_1, \dots, N_{k-1})$. $P(\mathbf{N})$ is a nonlinear function of buffer sizes $\mathbf{N}$, and is calculated numerically by decomposition for lines having more than two machines. The numerical method is described in Gershwin (1994).

The profit of a k -machine, $k - 1$ -buffer line can be formulated as

$$\text{Profit} = AP(N_1, \dots, N_{k-1}) - \sum_{i=1}^{k-1} b_i N_i - \sum_{i=1}^{k-1} c_i \bar{n}_i - K,$$

where $A > 0$ is the profit coefficient associated with the production rate $P(\mathbf{N})$, b_i and c_i are cost coefficients associated with the buffer space and average inventory for the i th buffer, respectively, and K stands for all costs other than those due to buffer sizes, average inventory, and raw material. Since K is independent of $\mathbf{N}$, we simplify the formulation above and write our objective function as

$$J(N_1, \dots, N_{k-1}) = AP(N_1, \dots, N_{k-1}) - \sum_{i=1}^{k-1} b_i N_i - \sum_{i=1}^{k-1} c_i \bar{n}_i. \quad (1)$$

In the following, we refer to J as the profit of the line. To simplify terminology, the first item on the right side of equation (1) can be seen as the total revenue of the line; while the other two items together can be interpreted as the total cost of the line. Allowing different buffers to have different cost coefficients is realistic as we know that, for example, the cost of buffer space in a clean room is much expensive than elsewhere in a factory.

2.2 Monotonicity and Concavity of $P(\mathbf{N})$

A common intuition in the line design field is the concavity of $P(\mathbf{N})$, though there is no analytical result that conclusively shows that the lines we study in this paper exhibit the concavity property. However, some research in similar systems indicates that this is a reasonable assumption. Shanthikumar and Yao (1989a) pointed out monotonicity and concavity properties in cyclic queueing networks with finite buffers. In Shanthikumar and Yao (1989b), they established the concavity of the throughput in a multicell system. Anantharam and Tsoucas (1990) proved stochastic concavity of throughput in a series of queues with finite buffers. Meester and Shanthikumar (1990) showed the concavity of throughput in tandem queueing systems with finite buffers and exponential servers. Dallery, Liu, and Towsley (1994) generalized their work and present the concavity properties in Fork/Join queueing networks. Glasserman and Yao (1996) found that the throughput was a concave function of buffer parameters in serial lines with general blocking and synchronized service.

In addition, some work based on the concavity of the production rate for the same or similar systems has been published. Park (1993) assumed the concavity of the production rate over both a buffer and a vector of buffers in his study of buffer size optimization. Gershwin and Schor (2000) established a primal-dual algorithm for buffer space allocation in production lines basing on the assumption of concave $P(\mathbf{N})$. Levantesi, Matta, and Tolio (2001) presented an algorithm for buffer allocation in production lines with the same assumption. Jeong and Kim (2000) applied that property to assembly systems. So (1997) also mentioned the concavity of the production rate in his study on optimal buffer allocation strategy for unpaced production lines.

It is also necessary to assume that $P(\mathbf{N})$ is monotonically increasing in $\mathbf{N}$. Schor 1995 provided a relatively detailed survey on this property. Monotonicity and concavity of $P(\mathbf{N})$ are key assumption behind our algorithm.

2.3 Problem Formulations

In this section, we introduce mathematical models. Our prime goal is a constrained problem, in which we aim at maximizing profits of production lines subject to a production rate constraint. In order to solve the constrained problem, we present a corresponding unconstrained problem, in which we drop the production rate constraint. We introduce the two problems here and leave the reason for introducing the unconstrained problem to the next section.

2.3.1 The constrained problem

The constrained problem is formulated as follows:

$$\begin{aligned}
\max \quad & J(N_1, \dots, N_{k-1}) = AP(N_1, \dots, N_{k-1}) - \sum_{i=1}^{k-1} b_i N_i - \sum_{i=1}^{k-1} c_i \bar{n}_i \\
\text{subject to} \quad & P(N_1, \dots, N_{k-1}) \geq P^*, \\
& N_i \geq 0, \forall i = 1, \dots, k-1,
\end{aligned} \tag{2}$$

where P^* is the demand, or the required production rate. The first constraint is the production rate constraint. Note that it is nonlinear. The second constraint is called the buffer size constraint. It comes from the natural property of buffer sizes since it makes no sense to have negative buffer sizes.

However, it is necessary for us to further limit the buffer sizes. This is because we use the decomposition to evaluate the production rate of the line, and the decomposition is based on an analytical solution of the two-machine line (Gershwin 1994). This analytical two-machine-line evaluation requires buffer sizes to be greater than 2. For a line having buffer sizes equal to or less than 2, there are different ways to measure its performance and we do not discuss them in this paper. Therefore, we only focus on lines whose buffer sizes are all greater than 2. In the following, let N_{min} denote the minimum of the buffer size and we re-write the constrained problem as:

$$\begin{aligned} \max \quad & J(N_1, \dots, N_{k-1}) = AP(N_1, \dots, N_{k-1}) - \sum_{i=1}^{k-1} b_i N_i - \sum_{i=1}^{k-1} c_i \bar{n}_i \\ \text{subject to} \quad & P(N_1, \dots, N_{k-1}) \geq P^*, \\ & N_i \geq N_{min}, \forall i = 1, \dots, k-1. \end{aligned} \tag{3}$$

2.3.2 The unconstrained problem

In the unconstrained problem, we ignore the production rate constraint. Thus, the unconstrained problem is

$$\begin{aligned} \max \quad & J(N_1, \dots, N_{k-1}) = AP(N_1, \dots, N_{k-1}) - \sum_{i=1}^{k-1} b_i N_i - \sum_{i=1}^{k-1} c_i \bar{n}_i \\ \text{subject to} \quad & N_i \geq N_{min}, \forall i = 1, \dots, k-1. \end{aligned} \tag{4}$$

This is a convenient, although not quite accurate, name since we still have the buffer size constraint. As we show in Section 3, the unconstrained problem can be solved easily by a gradient method. We will further illustrate the relationship between the two problems and reveal how we take advantage of the unconstrained problem to solve the constrained one.

3 Solution technique

In this section, we present our algorithm for solving the constrained problem. Instead of solving it directly, we adopt a two-step strategy in which we introduce a new variable A' . We replace A by A' in the unconstrained problem (4) and solve it iteratively for different values of A' .

To solve the unconstrained problem, we take advantage of the analytical form of the two-machine-line evaluation, which enables us to treat N_i as continuous variables. This is because the formulas for the evaluation of the two-machine line do not require N_i to be integers³. Consequently,

³We include the continuous variable version of the solution of the two-machine line in Appendix A.

though our model is a deterministic processing time discrete state model, we can still treat N_i as continuous variables. In addition, since we evaluate $P(\mathbf{N})$ by the decomposition and the analytic two-machine-line evaluation, and treat N_i as continuous variables, we are able to treat $P(\mathbf{N})$ and $J(\mathbf{N})$ as continuously differentiable functions. Therefore, we adopt a gradient method to solve the unconstrained problem. However, we need to indicate that due to the lack of an analytical expression of the profit of a line having more than two machines, we compute gradients according to a forward difference formula.

It is important to point out that gradient methods are appropriate when the space being searched has a single maximum. This requires our objective function of the unconstrained problem, $J(N_1, \dots, N_{k-1})$, to have only one maximum point for the proposed optimization method to work correctly. Schor (1995) encountered the same issue and showed that $J(N_1, \dots, N_{k-1})$ has single maximum through numerical evidence and an intuitive argument. He introduced a two-step gradient method to solve our unconstrained problem. His results were demonstrated to be correct when compared with both simulation and another optimization method (Seong, Chang, and Hong 1994). Gershwin and Schor (2000) confirmed this.

3.1 Algorithm derivation

Solving the unconstrained problem yields two cases:

1. The unconstrained problem has a solution $(N_1^*, \dots, N_{k-1}^*)$ in which $P(N_1^*, \dots, N_{k-1}^*) \geq P^*$. In this case, the solution of the constrained problem is $(N_1^*, \dots, N_{k-1}^*)$.
2. The solution of the unconstrained problem satisfies $P < P^*$. This is not the solution of the constrained problem. In this case, we consider the following unconstrained problem:

$$\max J(N_1, \dots, N_{k-1}) = A'P(N_1, \dots, N_{k-1}) - \sum_{i=1}^{k-1} b_i N_i - \sum_{i=1}^{k-1} c_i \bar{n}_i \quad (5)$$

$$\text{subject to } N_i \geq N_{min}, \forall i = 1, \dots, k-1,$$

in which A of (4) is replaced by A' . Let $(N'_1, \dots, N'_{k-1})$ be the solution to this problem and $P' = P(N'_1, \dots, N'_{k-1})$. Then, we claim the following.

Assertion The constrained problem

$$\begin{aligned} \max J(N_1, \dots, N_{k-1}) &= A'P(N_1, \dots, N_{k-1}) - \sum_{i=1}^{k-1} b_i N_i - \sum_{i=1}^{k-1} c_i \bar{n}_i \\ \text{subject to } P(N_1, \dots, N_{k-1}) &\geq P^* \\ N_i &\geq N_{min}, \forall i = 1, \dots, k-1 \end{aligned} \quad (6)$$

has the same solution for all A' in which the solution of the unconstrained problem (5) has $P' < P^*$.

This is because the solution of problem (6) will satisfy $P(N_1, \dots, N_{k-1}) = P^*$ so the objective function is equivalent to $A'P^* - \sum_{i=1}^{k-1} b_i N_i - \sum_{i=1}^{k-1} c_i \bar{n}_i$. Since the first term is independent of all of the N_i , it has no effect on the solution of the problem.

We claim in the assertion that if the optimal solution of the unconstrained problem (4) is not the solution of the constrained problem (3), then the solution of the constrained problem (3), $(N_1^*, \dots, N_{k-1}^*)$, satisfies $P(N_1^*, \dots, N_{k-1}^*) = P^*$. Therefore, to solve the constrained problem (3), we replace A by A' in both problems (3) and (4) and solve problem (5) for different A' 's. We need to find the value of A' such that the solution to problem (5) satisfies $P(N_1, \dots, N_{k-1}) = P^*$. Then, this solution is the same as that of the original constrained problem (3).

We prove this assertion by the Karush-Kuhn-Tucker (KKT) conditions of nonlinear programming. We first convert the constrained problem (3) into the minimization form:

$$\begin{aligned} \min \quad & -J(N_1, \dots, N_{k-1}) = -AP(N_1, \dots, N_{k-1}) + \sum_{i=1}^{k-1} b_i N_i + \sum_{i=1}^{k-1} c_i \bar{n}_i \\ \text{subject to} \quad & P^* - P(N_1, \dots, N_{k-1}) \leq 0, \\ & N_{\min} - N_i \leq 0, \forall i = 1, \dots, k-1. \end{aligned} \tag{7}$$

We have argued that we treat N_i as continuous variables, and $P(\mathbf{N})$ and $J(\mathbf{N})$ as continuously differentiable functions. Let us consider the KKT conditions. A statement of the KKT conditions (Bertsekas 1999) is that: Let x^* be a local minimum of the problem

$$\begin{aligned} \min \quad & f(x) \\ \text{subject to} \quad & h_1(x) = 0, \dots, h_m(x) = 0, \\ & g_1(x) \leq 0, \dots, g_r(x) \leq 0, \end{aligned} \tag{8}$$

where f , h_i , and g_j are continuously differentiable functions from $\mathbb{R}^n$ to $\mathbb{R}$. Assume that x^* is regular⁴. Then there exist unique Lagrange multipliers $\lambda_1^*, \dots, \lambda_m^*$ and $\mu_1^*, \dots, \mu_r^*$, satisfying the following conditions:

$$\begin{aligned} \nabla_x L(x^*, \lambda^*, \mu^*) &= 0, \\ \mu_j^* &\geq 0, j = 1, \dots, r, \\ \mu_j^* g_j(x^*) &= 0, j = 1, \dots, r. \end{aligned} \tag{9}$$

where $L(x, \lambda, \mu) = f(x) + \sum_{i=1}^m \lambda_i h_i(x) + \sum_{j=1}^r \mu_j g_j(x)$ is called the Lagrangian function.

⁴A feasible vector x is regular if the equality constraint gradients $\nabla h_i(x), i = 1, \dots, m$, and the active inequality constraint gradients $\nabla g_j(x), j \in M(x)$, are linearly independent. Also x is regular in the exceptional case where there are no equality constraints and all the inequality constraints are inactive at x . $M(x)$ is the set of active inequality constraints, i.e., $M(x) = \{j | g_j(x) = 0\}$ (Bertsekas 1999).

Before we apply the KKT conditions to our problem, we need to point out a necessary condition that guarantees the existence of Lagrange multipliers. One appropriate for our problem is the Slater constraint qualification for convex inequalities (Bertsekas 1999), which is: Let x^* be a local minimum of the problem (8), where f and g_j are continuously differentiable functions from $\mathbb{R}^n$ to $\mathbb{R}$, and the functions h_i are linear. Assume that the functions g_j are convex and that there exists a feasible vector $\bar{x}$ satisfying $g_j(\bar{x}) < 0, \forall j \in M(x^*)$. Then x^* satisfies the KKT conditions.

Let us now consider our constrained problem. There are no equality constraints in the problem, but there are k inequality constraints:

$$\begin{aligned} g_0(\mathbf{N}) &= P^* - P(N_1, \dots, N_{k-1}) \leq 0, \\ g_i(\mathbf{N}) &= N_{min} - N_i \leq 0, \forall i = 1, \dots, k-1. \end{aligned} \quad (10)$$

Due to the concavity of $P(\mathbf{N})$, $g_0(\mathbf{N})$ is a convex function. All other $g_i(\mathbf{N})$ are linear so that they are also convex. It is not hard to find a feasible vector to make our problem satisfy the Slater constraint qualification. Since the required production rate, P^* , has to be feasible for the line, there exists sufficiently large $\hat{\mathbf{N}}$ such that $P(\hat{N}_1, \dots, \hat{N}_{k-1}) > P^*$ so $g_0(\hat{N}_1, \dots, \hat{N}_{k-1}) < 0$. In addition, $g_i(\hat{N}_1, \dots, \hat{N}_{k-1}) < 0, \forall i = 1, \dots, k-1$ because $N_{min} - \hat{N}_i < 0, \forall i = 1, \dots, k-1$. Hence, our constrained problem satisfies the Slater constraint qualification, and there exist unique Lagrange multipliers $\mu_i^*, i = 0, \dots, k-1$ for our problem to satisfy the KKT conditions:

$$-\nabla J(\mathbf{N}^*) + \mu_0^* \nabla(P^* - P(\mathbf{N}^*)) + \sum_{i=1}^{k-1} \mu_i^* \nabla(N_{min} - N_i) = 0 \quad (11)$$

or

$$-\begin{pmatrix} \frac{\partial J(\mathbf{N}^*)}{\partial N_1} \\ \frac{\partial J(\mathbf{N}^*)}{\partial N_2} \\ \vdots \\ \frac{\partial J(\mathbf{N}^*)}{\partial N_{k-1}} \end{pmatrix} - \mu_0^* \begin{pmatrix} \frac{\partial P(\mathbf{N}^*)}{\partial N_1} \\ \frac{\partial P(\mathbf{N}^*)}{\partial N_2} \\ \vdots \\ \frac{\partial P(\mathbf{N}^*)}{\partial N_{k-1}} \end{pmatrix} - \mu_1^* \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} - \mu_2^* \begin{pmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{pmatrix} - \dots - \mu_{k-1}^* \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \quad (12)$$

and

$$\mu_i^* \geq 0, \forall i = 0, \dots, k-1, \quad (13)$$

$$\mu_0^*(P^* - P(\mathbf{N}^*)) = 0, \quad (14)$$

$$\mu_i^*(N_{min} - N_i^*) = 0, \forall i = 1, \dots, k-1, \quad (15)$$

where $\mathbf{N}^*$ is the optimal solution of our constrained problem.

Next, we show that finding the Lagrange multipliers $\mu_i^*, i = 0, \dots, k-1$ and the optimal solution $\mathbf{N}^*$ to satisfy the KKT conditions (12) to (15) is equivalent to solving the constrained problem (3) by our algorithm. Suppose that $\mathbf{N}^*$ is an interior solution⁵. In all our experiments, the optimal

⁵An interior solution means that all N_i^* in $\mathbf{N}^*$ are greater than N_{min} . We discuss the case in which some N_i^* are on the boundary, i.e., $N_i^* = N_{min}$, in Appendix B.

solutions have this feature. In this case, by condition (15), we know that $\mu_i^* = 0, \forall i = 1, \dots, k-1$. Hence, we can simplify the KKT conditions (12) to (15) to

$$-\begin{pmatrix} \frac{\partial J(\mathbf{N}^*)}{\partial N_1} \\ \frac{\partial J(\mathbf{N}^*)}{\partial N_2} \\ \vdots \\ \frac{\partial J(\mathbf{N}^*)}{\partial N_{k-1}} \end{pmatrix} - \mu_0^* \begin{pmatrix} \frac{\partial P(\mathbf{N}^*)}{\partial N_1} \\ \frac{\partial P(\mathbf{N}^*)}{\partial N_2} \\ \vdots \\ \frac{\partial P(\mathbf{N}^*)}{\partial N_{k-1}} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \quad (16)$$

$$\mu_0^*(P^* - P(\mathbf{N}^*)) = 0, \quad (17)$$

where $\mu_0^* \geq 0$. We know, since $\mathbf{N}^*$ is not the optimal solution of the unconstrained problem, that $\nabla J(\mathbf{N}^*) \neq 0$. $\nabla J(\mathbf{N}^*) \neq 0$ means that not all $\partial J(\mathbf{N}^*)/\partial N_i$ are equal to 0. Thus, $\mu_0^* \neq 0$ since otherwise condition (16) would be violated. By condition (17), the optimal solution $\mathbf{N}^*$ satisfies $P(\mathbf{N}^*) = P^*$. Since g_0 is the only active inequality constraint, $\mathbf{N}^*$ is regular. In addition, conditions (16) and (17) reveal how we could find μ_0^* and $\mathbf{N}^*$. For every μ_0^* , condition (16) determines $\mathbf{N}^*$ since there are $k-1$ equations and $k-1$ unknowns. Therefore, we can think of $\mathbf{N}^* = \mathbf{N}^*(\mu_0^*)$. We search for a value of μ_0^* such that $P(\mathbf{N}^*(\mu_0^*)) = P^*$. As we indicate in the following, this is exactly what our algorithm does.

Replacing μ_0^* by $\mu_0 > 0$ in constraint (16) gives

$$-\begin{pmatrix} \frac{\partial J(\bar{\mathbf{N}})}{\partial N_1} \\ \frac{\partial J(\bar{\mathbf{N}})}{\partial N_2} \\ \vdots \\ \frac{\partial J(\bar{\mathbf{N}})}{\partial N_{k-1}} \end{pmatrix} - \mu_0 \begin{pmatrix} \frac{\partial P(\bar{\mathbf{N}})}{\partial N_1} \\ \frac{\partial P(\bar{\mathbf{N}})}{\partial N_2} \\ \vdots \\ \frac{\partial P(\bar{\mathbf{N}})}{\partial N_{k-1}} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \quad (18)$$

where $\bar{\mathbf{N}}$ is the unique solution of (18). Note that $\bar{\mathbf{N}}$ is the solution of the following optimization problem

$$\min \quad -\bar{J}(\mathbf{N}) = -J(\mathbf{N}) + \mu_0(P^* - P(\mathbf{N})) \quad (19)$$

$$\text{subject to} \quad N_{\min} - N_i \leq 0, \forall i = 1, \dots, k-1.$$

which is equivalent to

$$\max \quad \bar{J}(\mathbf{N}) = J(\mathbf{N}) - \mu_0(P^* - P(\mathbf{N})) \quad (20)$$

$$\text{subject to} \quad N_{\min} - N_i \leq 0, \forall i = 1, \dots, k-1.$$

or

$$\max \quad \bar{J}(\mathbf{N}) = AP(\mathbf{N}) - \sum_{i=1}^{k-1} b_i N_i - \sum_{i=1}^{k-1} c_i \bar{n}_i - \mu_0(P^* - P(\mathbf{N})) \quad (21)$$

$$\text{subject to } N_{min} - N_i \leq 0, \forall i = 1, \dots, k-1.$$

or

$$\max \quad \bar{J}(\mathbf{N}) = (A + \mu_0)P(\mathbf{N}) - \sum_{i=1}^{k-1} b_i N_i - \sum_{i=1}^{k-1} c_i \bar{n}_i \quad (22)$$

$$\text{subject to } N_i \geq N_{min}, \forall i = 1, \dots, k-1.$$

or, finally,

$$\max \quad \bar{J}(\mathbf{N}) = A'P(\mathbf{N}) - \sum_{i=1}^{k-1} b_i N_i - \sum_{i=1}^{k-1} c_i \bar{n}_i \quad (23)$$

$$\text{subject to } N_i \geq N_{min}, \forall i = 1, \dots, k-1.$$

where $A' = A + \mu_0$. This is exactly the unconstrained problem (5), and $\bar{\mathbf{N}}$ is its optimal solution. Note that $\mu_0 > 0$ indicates that $A' > A$. In addition, the KKT condition (17) indicates that the optimal solution of the constrained problem $\mathbf{N}^*$ satisfies $P(\mathbf{N}^*) = P^*$. This means that, for every $A' > A$ (or $\mu_0 > 0$), we can find the corresponding optimal solution $\bar{\mathbf{N}}$ satisfying condition (18) by solving problem (5), and, we need to find the A' such that the solution to problem (5), denoted as $\mathbf{N}^{A'}$, satisfies $P(\mathbf{N}^{A'}) = P^*$. Then, $\mu_0 = A' - A$ and $\mathbf{N}^{A'}$ satisfy conditions (16) and (17). Hence, $\mu_0 = A' - A$ is exactly the Lagrange multiplier satisfying the KKT conditions of our constrained problem, and $\mathbf{N}^* = \mathbf{N}^{A'}$ is the optimal solution of our constrained problem. Consequently, solving the constrained problem (3) through our algorithm is essentially finding the unique Lagrange multipliers and optimal solution of the problem. We have proven our assertion.

Therefore, in our algorithm, we conduct an one-dimensional search on $A' > A$ and stop after we find the A' such that the solution to the unconstrained problem satisfies $P(N_1^*, \dots, N_{k-1}^*) = P^*$. We conclude that it is also the desired solution of the constrained problem. We summarize our algorithm in Section 3.2.

3.2 Algorithm statement

1. Check the feasibility of the problem. We describe this in Section 3.5.
2. Solve the unconstrained problem (4). If the solution $(N_1^*, \dots, N_{k-1}^*)$ satisfies $P(N_1^*, \dots, N_{k-1}^*) \geq P^*$, stop. The solution of the constrained problem is $(N_1^*, \dots, N_{k-1}^*)$. This step is also the necessity check of the algorithm as we point out in Section 3.5.
3. Do a one-dimensional search on $A' > A$ to find A' such that the solution of the unconstrained problem satisfies $P(N_1^*, \dots, N_{k-1}^*) = P^*$. Stop. The desired solution is $(N_1^*, \dots, N_{k-1}^*)$.

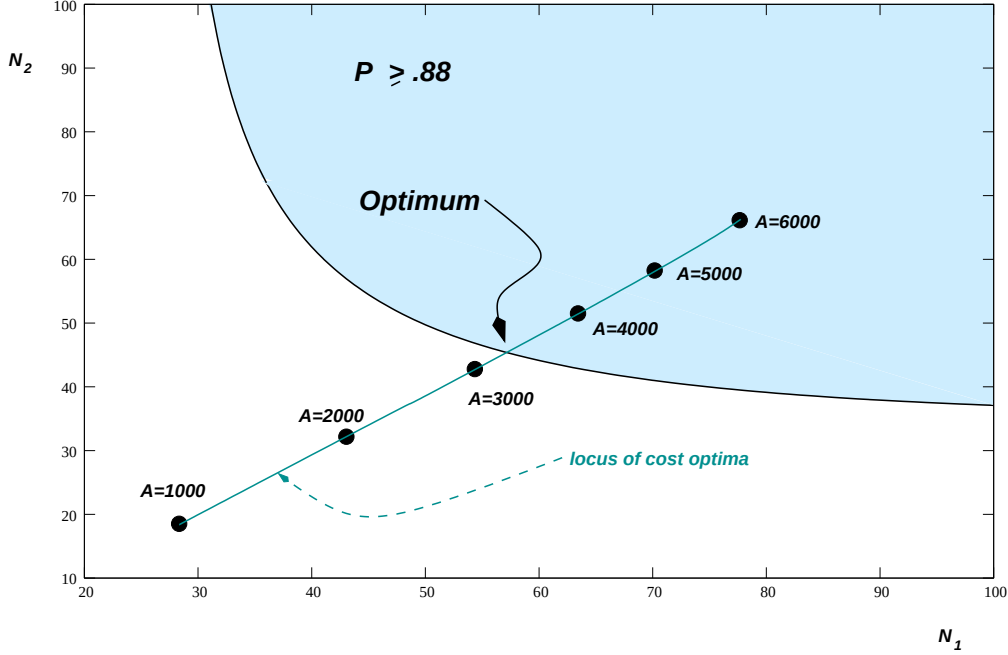


Figure 2: An example of our algorithm

3.3 An example of our algorithm

We provide an example to show our algorithm. It is a three-machine, two-buffer line with parameters $r_1 = .12$, $p_1 = .01$, $r_2 = .09$, $p_2 = .01$, $r_3 = .11$, $p_3 = .01$, and the profit function $J(N_1, N_2) = 1000P(N_1, N_2) - .5N_1 - N_2$. Suppose first that the required production rate, P^* , is .85. Solving the unconstrained problem and letting $(\hat{N}_1, \hat{N}_2)$ be the optimal solution yield $P(\hat{N}_1, \hat{N}_2) = .8576$. Since $P(\hat{N}_1, \hat{N}_2) > P^* = .85$, the solution $(\hat{N}_1, \hat{N}_2)$ is equivalent to the solution of the constrained problem and no further search on $A' > A$ is needed.

Next suppose that the required production rate, P^* , is .88. In this case, the optimal solution of the unconstrained problem does not satisfy $P(\hat{N}_1, \hat{N}_2) \geq P^*$. Thus, we need to conduct the one-dimensional search on $A' > A$ to find the optimal solution of the constrained problem. So, we solve the unconstrained problem for $A' = 1000, 2000, \dots, 6000$, and the optimal solutions for distinct A' are shown in Figure 2. Figure 2 also tells that the required production rate $P^* = .88$. The locus of unconstrained optima of the cost function is sketched. Let $(N_1^{A'}, N_2^{A'})$ be the optimal solution of the unconstrained problem for a certain A' . Note that for $A' = 1000, 2000$, and 3000 , the unconstrained optima have $P(N_1^{A'}, N_2^{A'}) < .88$ while for $A' = 4000, 5000$, and 6000 , the unconstrained optima have $P(N_1^{A'}, N_2^{A'}) > .88$. Therefore, if the problem to be solved is to maximize $1000P - .5N_1 - N_2$ subject to $P \geq .88$, then the solution is the intersection of $P = .88$ and the locus of unconstrained optima.

3.4 Detailed description of the algorithm

We describe in detail the algorithm we propose to solve the constrained problem here. The algorithm includes solving the unconstrained problem and the one-dimensional search on $A' > A$, if necessary. We have explained that we can treat the decision variables, N_i , as continuous variables, and $J(N_1, \dots, N_{k-1})$ as a continuously differentiable function. Therefore, we adopt a gradient method that is based on the decomposition (Gershwin 1987) and the DDX algorithm (Dallery, David, and Xie 1988) to solve the unconstrained problem.

3.4.1 Gradient method for the unconstrained problem

We solve the unconstrained problem with a gradient method. An initial guess $(N_1^0, \dots, N_{k-1}^0)$ is selected first. For example, N_i^0 can be chosen as the minimal value to satisfy $P(\infty, \dots, N_i^0, \dots, \infty) = P^*$. This is how we choose the initial guess in our implementation of the algorithm. However, it is helpful to indicate that our algorithm does not require that $P(N_1^0, \dots, N_{k-1}^0) \leq P^*$. We have verified this with experiments and the initial guesses $(N_1^0, \dots, N_{k-1}^0)$ such that $P(N_1^0, \dots, N_{k-1}^0) > P^*$ lead to the same solution.

Then, we calculate the gradient direction to move in $(N_1, \dots, N_{k-1})$ space. A line search is then conducted in that direction until a maximum is encountered. This becomes the next guess. A new direction is chosen and the process continues until no improvement can be achieved. There is no analytical expression to compute profits of lines having more than two machines. Consequently, to determine the search direction, we compute the gradient, $\mathbf{g}$, according to a forward difference formula, which is

$$g_i = \frac{J(N_1, \dots, N_i + \delta N_i, \dots, N_{k-1}) - J(N_1, \dots, N_i, \dots, N_{k-1})}{\delta N_i} \quad (24)$$

where g_i is the gradient component of buffer B_i , J is the profit of the line and can be obtained by equation (1), and δN_i is the increment of buffer B_i . Since we treat N_i as continuous variables, in the gradient calculation above, we choose $\delta N_i = .01$, which has proved to be a good choice in all experiments we have conducted.

Apart from acquiring the gradient direction, we still need to determine the step size, a . We conduct a bisection search to find a such that $J(N_1 + ag_1, \dots, N_{k-1} + ag_{k-1})$, or $J(\mathbf{N} + a\mathbf{g})$, is maximized. Then, we calculate the next gradient and repeat. This process ends when there is no improvement in profit or when all components of the gradient are sufficiently small. A block diagram of this gradient method appears in Figure 3.

3.4.2 One dimensional search on $A' > A$

To find A' such that the solution of the unconstrained problem satisfies $P(N_1, \dots, N_{k-1}) = P^*$, we use the Newton Chord method, which is an efficient way to find t' such that $f(t') = 0$ for a given function $f(\cdot)$. Thus, in our algorithm, for any particular value of A' , we define $f(A')$ as

$$f(A') = P(\mathbf{N}^{opt_{A'}}) - P^* \quad (25)$$

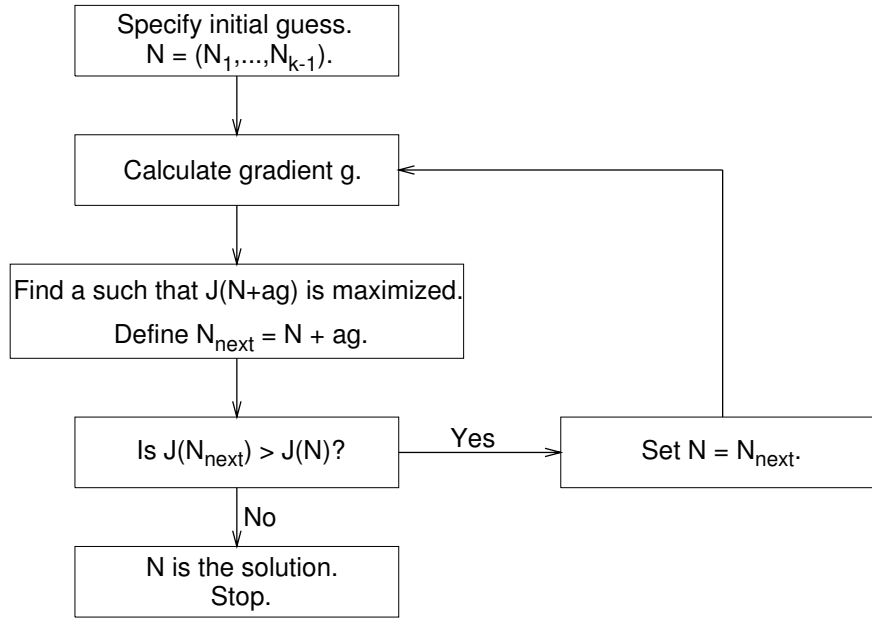


Figure 3: Block diagram of the gradient method

where $\mathbf{N}^{opt_{A'}}$ is the optimal buffer allocation associated with that A' and P^* is the required production rate. The Newton Chord method in our algorithm consists of the following three steps:

1. Guess A'_0 and A'_1 . Calculate the approximate slope

$$s = \frac{f(A'_1) - f(A'_0)}{A'_1 - A'_0}. \quad (26)$$

2. Choose A'_2 so that

$$f(A'_0) + (A'_2 - A'_0)s = 0, \quad (27)$$

or

$$A'_2 = -\frac{f(A'_0)}{s} + A'_0. \quad (28)$$

3. Repeat with $A'_0 = A'_1$ and $A'_1 = A'_2$ until $|f(A'_0)|$ is small enough.

In our implementation, the termination criterion is $|f(A'_0)| \leq 10^{-4}$. The first two values of A' are $A'_0 = A$ and $A'_1 = A + 1000$.

3.5 Implementation issues

Implementation issues about the algorithm include the feasibility and necessity of the algorithm, the initialization of the DDX algorithm of the decomposition, and the conversion from continuous solutions to integers.

Before running the algorithm, we should ensure that the required production rate, P^* , is feasible for the line to be optimized. This means that P^* should satisfy

$$P^* < \min_i \frac{r_i}{r_i + p_i} \quad (29)$$

where $r_i/(r_i + p_i)$ is the isolated production rate of machine M_i . If this fails, no set of buffers can satisfy the production rate constraint.

We also have to make sure that there is a need to conduct the one-dimensional search on $A' > A$ of our algorithm. This actually can be decided after we first solve the unconstrained problem. We have indicated this in Section 3.1 and restate it as the second step of the algorithm in Section 3.2. If the unconstrained problem (4) has a solution $(N_1, \dots, N_{k-1})$ in which $P(N_1, \dots, N_{k-1}) \geq P^*$, then we do not need to implement the one-dimensional search on $A' > A$ and the solution is equivalent to that of the constrained problem.

Furthermore, since we apply the DDX algorithm for the decomposition and it is an iterative algorithm, we must initialize it whenever it is called in our algorithm. We use the most recent value of $(N_1, \dots, N_{k-1})$ from the last evaluation, instead of a standard initialization, to reduce the numbers of iterations and the two-machine-line evaluations.

Finally, it is necessary to point out that, to use the result of the algorithm in practical production line design for factories, we have to convert it back to integers.

4 Numerical results and analysis

Numerical results are provided to show not only the efficiency of our algorithm but also its implementation process. Hence, taking a four-machine, three-buffer line as an example, we first explain how $P(\mathbf{N}^*)$ changes with A' . Then, we apply the algorithm to both short and long lines to illuminate its efficiency. Computation issues are discussed at the end of this section. In our implementation of the algorithm, we let N_{min} be $2 + \epsilon$, where $\epsilon = 10^{-6}$. In addition, the algorithm is written with Matlab and run for all experiments on a computer with a 2.4 GHz Intel Core 2 Duo CPU.

4.1 System behavior

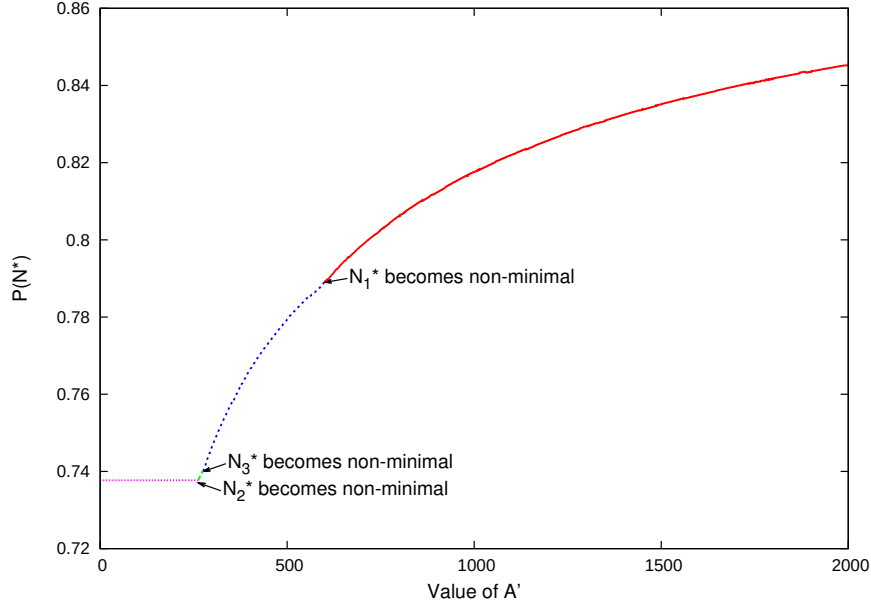
Let us consider a four-machine, three-buffer line, whose parameters are summarized in Table 1. All cost coefficients are set to be 1 so that profit is calculated as

$$J(N_1, \dots, N_3) = A'P(N_1, \dots, N_3) - \sum_{i=1}^3 N_i - \sum_{i=1}^3 \bar{n}_i. \quad (30)$$

To study the system behavior of the algorithm, we run it for this line to generate the curve of $P(\mathbf{N}^*)$ versus A' . We vary the A' from 0 to 2000 with a step size of 1 and the desired curve are shown in Figure 4. There are four segments in the curve. The flat segment stands for the case in which the optimal sizes of all buffers are N_{min} . For A' ranging from 0 to 263, the optimal buffer sizes for the line stay on the boundary of the feasible region so they are $(N_{min}, N_{min}, N_{min})$. The

Table 1: Data for the system behavior experiment

Machine	M_1	M_2	M_3	M_4
r	.11	.12	.10	.09
p	.008	.01	.01	.01

Figure 4: $P(\mathbf{N}^*)$ Vs. A'

production rates associated with those A' are identical, forming the horizontal segment in the curve. After A' passes 263, one of the optimal buffer sizes (N_2^*) becomes greater than N_{min} , i.e., it turns to a non-minimal value from N_{min} . From that time, the production rate begins to monotonically increase with A' . Each time a new buffer becomes non-minimal, the derivative of the production rate curve changes, but the curve remains continuous. Figure 4 suggests that $P(\mathbf{N}^*)$ for which not all components in $\mathbf{N}^*$ are $2 + \epsilon$ increases with A' monotonically. This system behavior further verifies our algorithm; as A' increases, $P(\mathbf{N}^*)$ increases monotonically so we can eventually find the A' such that $P(\mathbf{N}^*) \geq P^*$ (for P^* feasible).

4.2 Experiments on short lines

Our algorithm optimizes short lines very quickly, which can be shown by the following two experiments. We optimize a five-machine line and a six-machine line, and compare the optimal solutions of our algorithm with solutions gained from searching the $P = P^*(N_1, \dots, N_{k-1})$ surface

Table 2: Machine parameters of the five-machine line experiment

machine	M_1	M_2	M_3	M_4	M_5
r	.11	.12	.10	.09	.10
p	.008	.01	.01	.01	.01

in $(N_1, \dots, N_{k-1})$ space⁶.

4.2.1 Experiment on a five-machine, four-buffer line

The machine parameters are listed in Table 2. The required production rate P^* is .88. It is easy to check that the isolated production rate of the bottleneck of the line, machine M_4 , is greater than P^* , so the problem is feasible. The profit is calculated as:

$$J = 2500P(\mathbf{N}) - \sum_{i=1}^4 N_i - \sum_{i=1}^4 \bar{n}_i.$$

To verify the optimal solution, we conduct a search on the P^* surface in $(N_1, \dots, N_{k-1})$ space. We search on the surface around the optimal solution of our algorithm and the step size for all buffers in our search is 0.01. Experimental results including optimal solutions both from our algorithm and P^* surface search, and errors, are generated in Table 3. We see that the maximal error is 0.84% and appears in $\bar{n}_2$. Computer time for this experiment is 2.71 seconds. The number of the two-machine-line evaluations is 89148. (The reason we provide this number is that it is not affected by the capability and performance of the computer that runs the algorithm.) Hence, our algorithm offers accurate results very quickly.

The optimal solution reveals that, since machine M_4 is the bottleneck, the optimal size of buffer B_3 is greater than the optimal sizes of other three buffers. So, next we change b_3 , the cost coefficient associated with buffer size of B_3 , to 2. This means that the line pays more for the buffer size of B_3 . Thus, we expect the optimal solution for the new line to have a smaller size for B_3 while greater sizes for the other three buffers so as to satisfy the production rate constraint. Experimental results confirm our expectation (See Table 4). We see that the optimal buffer size of B_3 is reduced from 92.6403 to 78.8163, and optimal buffer sizes of other three buffers increase. The maximal error is 0.93% and appears in $\bar{n}_1$. Computer time for the revised experiment is 3.02 seconds. The number of the two-machine-line evaluations is 95328. To further study this phenomenon, we conduct more experiments for this line by varying the cost coefficient of B_3 from 0 to 14 with a step size of 0.2, and report results in Figure 5. Figure 5 indicates that as the cost coefficient of B_3 becomes larger and larger, the optimal value of B_3 becomes smaller to limit the cost spent on it. Meanwhile the optimal values of the other three buffers get larger so the line maintains the required production rate.

⁶For a brief introduction about P^* surface search, see Appendix C.

Table 3: Results of the five-machine line experiment

	P^* Surface Search	our algorithm	error
Production rate	.8800	.8800	
N_1^{opt}	28.70	28.6973	0.01%
N_2^{opt}	57.94	58.3192	0.65%
N_3^{opt}	92.92	92.6403	0.30%
N_4^{opt}	87.57	87.1665	0.46%
Inventory of B_1	18.9269	18.9257	0.007%
Inventory of B_2	33.8823	34.1677	0.84%
Inventory of B_3	48.4858	48.4529	0.07%
Inventory of B_4	31.8914	31.8059	0.27%
Profit (\$)	1799.7	1799.8	0.006%

Table 4: Results of the revised five-machine line experiment

	P^* Surface Search	our algorithm	error
Production rate	.8800	.8800	
N_1^{opt}	30.76	30.9987	0.78%
N_2^{opt}	64.77	64.4949	0.42%
N_3^{opt}	78.88	78.8163	0.08%
N_4^{opt}	96.19	95.9437	0.26%
Inventory of B_1	20.5419	20.7330	0.93%
Inventory of B_2	39.7438	39.6175	0.32%
Inventory of B_3	41.6177	41.5989	0.05%
Inventory of B_4	33.5221	33.4626	0.18%
Profit (\$)	1715.1	1715.4	0.02%

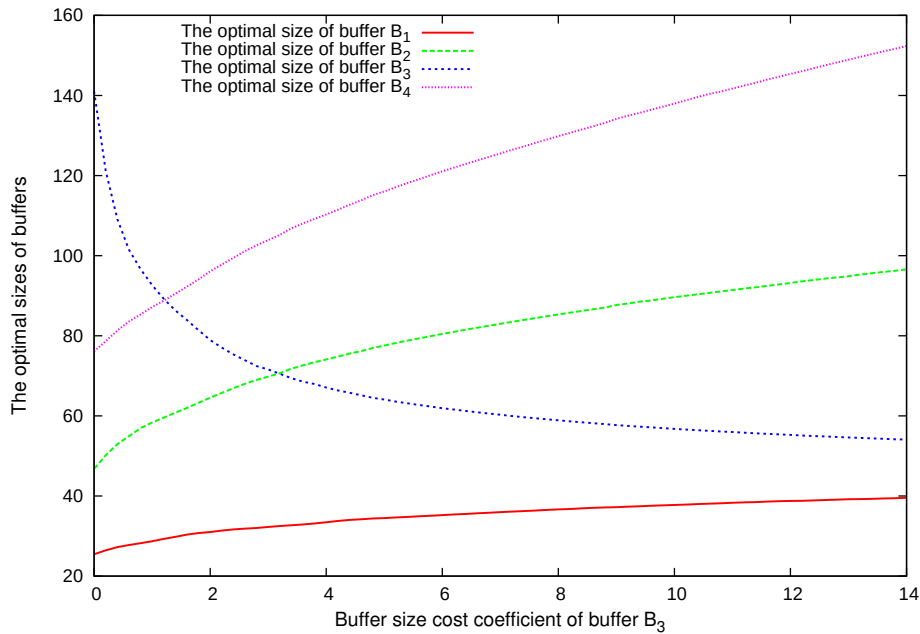

 Figure 5: Optimal buffer spaces Vs. cost coefficient of B_3

Table 5: Machine parameters of the six-machine line experiment

machine	M_1	M_2	M_3	M_4	M_5	M_6
r	.11	.12	.10	.09	.10	.11
p	.008	.01	.01	.01	.01	.009

buffer	B_1	B_2	B_3	B_4	B_5
b_i	1	2	.5	.8	1
c_i	1	1	2	1	1.5

4.2.2 Experiment on a six-machine, five-buffer line

The machine parameters and cost coefficients are provided in Table 5. The required production rate P^* is still .88, and it is easy to check that it is feasible for the line. The profit is calculated as

$$J = 3000P(\mathbf{N}) - \sum_{i=1}^4 b_i N_i - \sum_{i=1}^4 c_i \bar{n}_i.$$

Experimental results are presented in Table 6. The maximal error between the solution of our algorithm and that of surface search is 0.72% and appears in $\bar{n}_1$. Computer time for this experiment is 7.72 seconds. The number of the two-machine-line evaluations is 286368.

Table 6: Results of the six-machine line experiment

	P^* Surface Search	our algorithm	error
Production rate	.8800	.8800	
N_1^{opt}	33.20	33.4002	0.60%
N_2^{opt}	46.25	46.3292	0.17%
N_3^{opt}	103.01	102.6612	0.34%
N_4^{opt}	111.51	110.9841	0.47%
N_5^{opt}	56.11	56.0119	0.17%
Inventory of B_1	22.4904	22.6530	0.72%
Inventory of B_2	26.4580	26.5590	0.38%
Inventory of B_3	51.4389	51.3941	0.09%
Inventory of B_4	42.7788	42.6756	0.24%
Inventory of B_5	17.4193	17.3998	0.11%
Profit (\$)	2096.7	2097.0	0.01%

Table 7: Machine parameters of the twelve-machine line experiment

machine	M_1	M_2	M_3	M_4	M_5	M_6
r	.11	.12	.10	.09	.10	.11
p	.008	.01	.01	.01	.01	.01

machine	M_7	M_8	M_9	M_{10}	M_{11}	M_{12}
r	.10	.11	.12	.10	.12	.09
p	.009	.01	.009	.008	.01	.009

4.3 Experiments on long lines

Next we apply our algorithm to lines with more than ten machines. In particular, we run an experiment for a twelve-machine, eleven-buffer line. Again, to verify our optimal solution, we conduct a search on the P^* surface. We search on the surface around the optimal solution gained by our algorithm but this time we set the step size in our search to 0.1 for all buffers to reduce search time. Machine parameters are shown in Table 7. We still use .88 as a feasible required production rate for this line. In addition, we set all cost coefficients to 1 and the profit is calculated as

$$J = 6000P(\mathbf{N}) - \sum_{i=1}^{11} N_i - \sum_{i=1}^{11} \bar{n}_i.$$

Experimental results are generated in Table 8. We see that the maximal error is 0.33% and appears in B_8 . It would be further reduced if we decreased the step size of the P^* surface search. Therefore, our algorithm provides accurate optimal solutions for long lines as well. Computer time for this experiment is 91.47 seconds. The number of the two-machine-line evaluations is 3236500.

Table 8: Results of the twelve-machine line experiment

	P^* Surface Search	our algorithm	error
Production rate	.8800	.8800	
N_1^{opt}	29.10	29.1769	0.26%
N_2^{opt}	59.20	59.2830	0.14%
N_3^{opt}	97.80	97.7980	0.002%
N_4^{opt}	107.50	107.4176	0.08%
N_5^{opt}	84.50	84.4804	0.02%
N_6^{opt}	70.80	70.6892	0.17%
N_7^{opt}	63.10	63.1893	0.14%
N_8^{opt}	53.10	52.9274	0.33%
N_9^{opt}	47.20	47.2232	0.05%
N_{10}^{opt}	47.90	47.7967	0.22%
N_{11}^{opt}	48.80	48.7716	0.06%
Inventory of B_1	19.2388	19.2986	0.31%
Inventory of B_2	34.9561	35.0423	0.25%
Inventory of B_3	52.5423	52.6032	0.12%
Inventory of B_4	45.1528	45.1840	0.07%
Inventory of B_5	34.4289	34.4770	0.14%
Inventory of B_6	30.7073	30.7048	0.01%
Inventory of B_7	28.0446	28.1299	0.30%
Inventory of B_8	21.5666	21.5438	0.11%
Inventory of B_9	21.5059	21.5442	0.18%
Inventory of B_{10}	22.6756	22.6496	0.11%
Inventory of B_{11}	20.8692	20.8615	0.04%
Profit (\$)	4239.3	4239.2	0.002%

4.4 Computation issues

Finally, we discuss some computation issues about our algorithm. Although we have shown that our algorithm offers the optimal solution for a twelve-machine, eleven-buffer line within two minutes, it is important to observe the algorithm behavior for longer lines. Thus, we run our algorithm for a series of experiments for lines having identical machines. We vary the length of the line from 4 machines up to 30 machines. Machine parameters are $p = .01$ and $r = .1$. In all cases, we choose a feasible production rate $P^* = .88$, and the objective function is

$$J(\mathbf{N}) = AP(\mathbf{N}) - \sum_{i=1}^{k-1} N_i - \sum_{i=1}^{k-1} \bar{n}_i. \quad (31)$$

where $A = 500k$ for the line of length k . Furthermore, we initialize $A'_0 = A$ and $A'_1 = A + 1000$. The numbers of the two-machine-line evaluations for lines with different lengths are generated in Table 9 and Figure 6, respectively. We fit a curve to those points in order to reveal the relation between the length of the production line and the computation effort of our algorithm. From those points in Figure 6, we guess that we can find an exponential curve to fit them. Using the Curve Fitting Toolbox of Matlab⁷, we find

$$y = 8.156 \times 10^5 e^{0.1933k}, \quad (32)$$

where y denotes the number of the two-machine-line evaluations and k denotes the length of the line. The curve is shown in Figure 6 as well. It can be seen, from Table 9, that it costs our algorithm less than 10 minutes to find the optimal buffer allocation for the fifteen-machine line, and about 20 minutes to find the optimal buffer allocation for the twenty-machine line. However, the exponential curve implies that our algorithm needs much more time for lines with more than thirty machines. As we have pointed out in Section 1, in practice, it is possible that several machines are located adjacently in series and followed by a buffer. So while it is not rare for a transfer line to have more than thirty machines, the number of buffers is often much smaller. However, the exponential curve spurs us to further improve our algorithm to make it more efficient for longer lines. We finally present the specific optimal solution for the thirty-machine line. The optimal sizes of buffers are shown in Figure 7(a). The curve of $\bar{n}_i/N_i$ is illustrated in Figure 7(b).

5 Conclusion

5.1 Summary

In this paper, we present an accurate, fast, and reliable algorithm for maximizing profits through buffer space optimization for production lines. In the cost function, we consider both buffer space cost and average inventory cost and assign different cost coefficients to different buffers. In addition, we include a production rate constraint in our problem. A nonlinear programming approach is adopted to solve the problem. Experimental results are also provided to show the accuracy and

⁷Please refer to <http://www.mathworks.com/products/curvefitting/>.

Table 9: Numbers of the two-machine-line evaluations for lines with different lengths

Line length	Production rate	# of 2-mach eval	Computer time (Sec)
4	.8799	20460	0.692
5	.8800	60330	1.658
6	.8799	116752	3.133
7	.8800	235380	6.014
8	.8799	461184	11.554
9	.8800	921998	22.955
10	.8800	1590080	41.169
11	.8799	2492856	61.393
12	.8800	3819240	98.121
13	.8800	5712564	153.719
14	.8800	8762496	232.136
15	.8800	13117832	347.881
20	.8799	44007696	1166.694
25	.8799	107669164	2789.636
30	.8799	267271054	6208.451

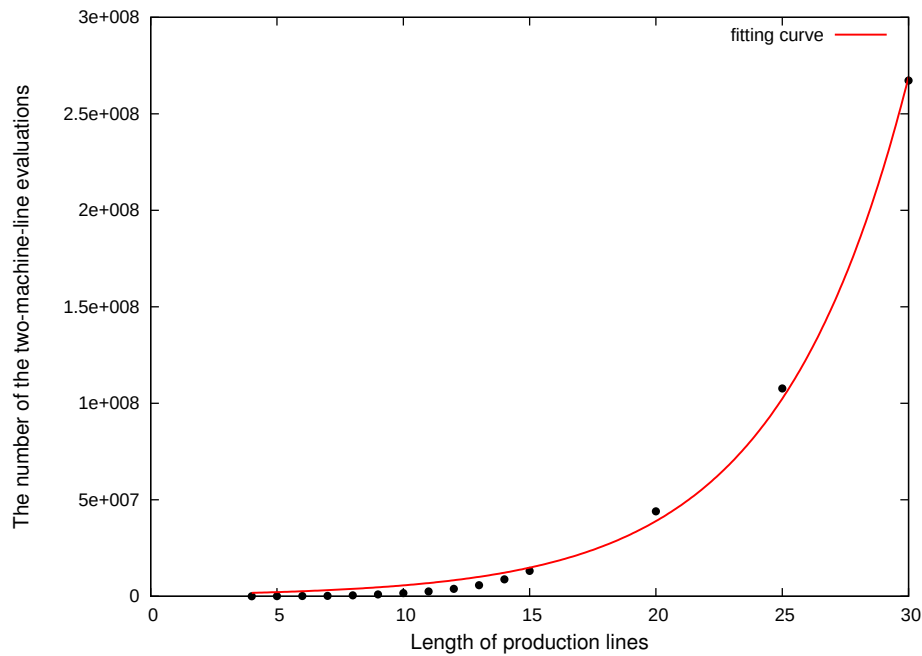


Figure 6: The Number of the two-machine-line evaluations Vs. The length of production lines and its fitting curve

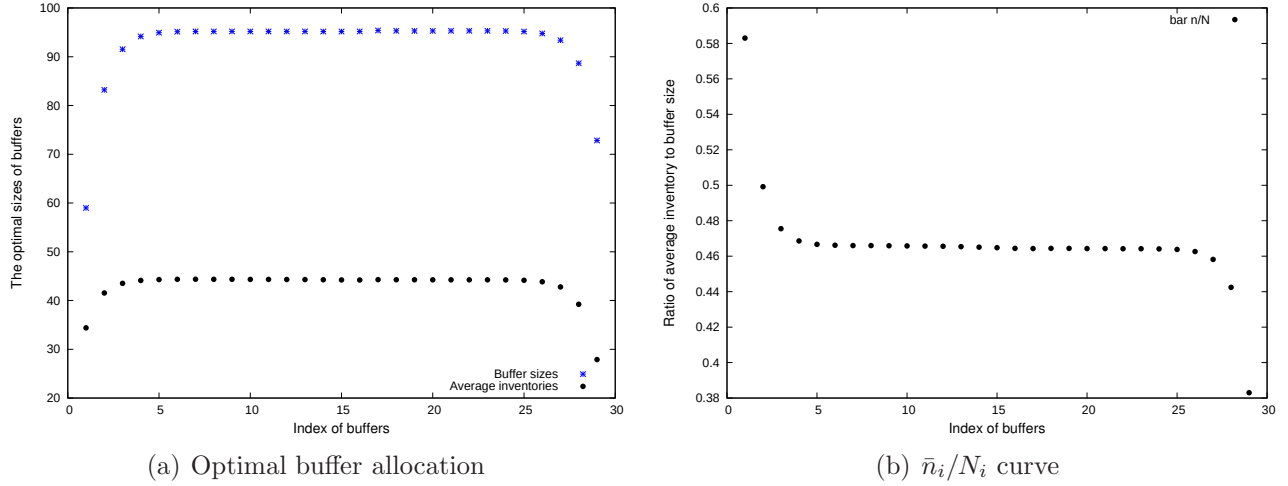


Figure 7: Results of the thirty-machine line

efficiency of our algorithm. Our algorithm is based on the assumption of the concavity of $P(\mathbf{N})$. Thus, one future research focus is to prove this assumption rigorously.

5.2 Future work

There are several research directions to which we can extend our algorithm in the future.

1. Assembly/Disassembly (A/D) systems.

A/D systems are extensions of lines in which assembly and disassembly take place. The first extension of our algorithm is to acyclic A/D systems.

2. Systems with loops.

Many production lines contain pallets or fixtures that travel in closed loops. Many production lines are controlled by constant work-in-progress (CONWIP) to restrict inventory. Such systems have tokens that travel in closed loops. Thus, there is a need for machine line designers to invite algorithms to enhance the performance of such lines. Werner (2001) and Gershwin and Werner (2007) presented a decomposition algorithm that enabled us to evaluate the production rate for lines with single loops. Zhang (2006) extended it to lines with multiple loops. So a good extension is that we take advantage of the decomposition algorithms to maximize profits for lines with loops.

3. Systems with machine selection.

In the design of a production line, each operation has various machine choices. Each machine has its own distinct parameters and cost. Different machine combinations lead to diverse production line performance, and we want to decide which machine to choose for each operation as we select buffers to maximize the profit for the line.

4. Systems with set-up cost for buffers.

This means whenever we decide to establish a buffer between two machines (or machine sets), we introduce a fixed buffer set-up cost. After the buffer is established, the buffer space cost will be proportional to its size. So, in this case, buffer space cost will be 0 if $N_i = 0$ or $a_i + b_i N_i$ if $N_i \geq N_{min}$ for buffer B_i .

5. Systems with quality control.

By taking account of quality control, we assume that machines generate both good parts and bad parts. Unfortunately, buffers delay the inspection of bad parts. Consider a two-machine line. Suppose parts are inspected only after the second machine. So, when the first machine begins to generate bad parts, we will not know it immediately if there are still some good parts in the buffer. We will only know that the first machine is generating bad parts after all good parts in the buffer are processed by the second machine and it begins to process the first bad part. During that delay, the first machine could have generated more bad parts. Thus, in this case, buffers bring potential delay to inspection, which reduces the production rate and the profit of the line. Kim and Gershwin (2005) pointed out that in the case of our example above, an increase of buffer size could either increase or decrease the production rate for different lines.

6. Systems with additional buffer space constraints.

In addition to the constraints in our constrained problem, we may encounter additional buffer space constraints of the form $\sum_i g_{ij} N_i \leq h_j$. For instance, consider a production line, in which two buffers, say buffers B_5 and B_6 , are in the clean room. Due to the high cost of buffer space in the clean room, there is a maximum total space constraint for B_5 and B_6 as $N_5 + N_6 \leq C$, where C is a constant. This brings additional constraints to our problem, so we need to make necessary modification to our algorithm so that it can apply to the new problem.

These extensions are key complements to make our model more practical for actual manufacturing systems in factories. A good understanding of our algorithm behavior on those extensions would be valuable in future production line design.

Acknowledgments

The authors are grateful for support from the MIT-Portugal Program and from the Singapore-MIT Alliance.

References

- Anantharam, V. and P. Tsoucas (1990). Stochastic concavity of throughput in series of queues with finite buffers. *Advances in Applied Probability* 22(3), 761–763.
- Bertsekas, D. P. (1999). *Nonlinear Programming: 2nd Edition*. Athena Scientific.

- Buzacott, J. A. (1967). Automatic transfer lines with buffer stocks. *International Journal of Production Research* 5(3), 183–200.
- Chan, F. T. S. and E. Y. H. Ng (2002). Comparative evaluations of buffer allocation strategies in a serial production line. *The International Journal of Advanced Manufacturing Technology* 19(11), 789–800.
- Dallery, Y., R. David, and X.-L. Xie (1988). An efficient algorithm for analysis of transfer lines with unreliable machines and finite buffers. *IIE Transactions* 20(3), 280–283.
- Dallery, Y. and S. B. Gershwin (1992). Manufacturing flow line systems: A review of models and analytical results. *Queueing Systems Theory and Applications, Special Issue on Queueing Models of Manufacturing Systems* 12, 3–94.
- Dallery, Y., Z. Liu, and D. Towsley (1994). Equivalence, reversibility, symmetry and concavity properties in fork-join queueing networks with blocking. *Journal of the ACM (JACM)* 41(5), 903–942.
- Diamantidis, A. C. and C. T. Papadopoulos (2004). A dynamic programming algorithm for the buffer allocation problem in homogeneous asymptotically reliable serial production lines. *Mathematical Problems in Engineering* 2004(3), 209–223.
- Dolgui, A., A. Ereemeev, A. Kolokolov, and V. Sigaev (2002). A genetic algorithm for the allocation of buffer storage capacities in a production line with unreliable machines. *Journal of Mathematical Modelling and Algorithms* 1(2), 89–104.
- Gershwin, S. B. (1987). An efficient decomposition method for the approximate evaluation of tandem queues with finite storage space and blocking. *Operations Research* 35(2), 291–305.
- Gershwin, S. B. (1994). *Manufacturing Systems Engineering*. Prentice-Hall. Currently available at <http://home.comcast.net/~hierarchy/MSE/mse.html>. For corrections, see <http://web.mit.edu/manuf-sys/www/gershwin.errata.html>.
- Gershwin, S. B. and I. C. Schick (1983). Modeling and analysis of three stage transfer lines with unreliable machines and finite buffers. *Operations Research* 31(2), 354–377.
- Gershwin, S. B. and J. E. Schor (2000). Efficient algorithms for buffer space allocation. *Annals of Operations Research* 93, 117–144.
- Gershwin, S. B. and L. M. Werner (2007). An approximate analytical method for evaluating the performance of closed-loop flow systems with unreliable machines and finite buffers. *International Journal of Production Research* 45(14), 3085–3111.
- Glasserman, P. and D. D. Yao (1996). Structured buffer-allocation problems. *Discrete Event Dynamic Systems* 6(1), 9–41.
- Huang, M.-G., P.-L. Chang, and Y.-C. Chou (2002, February). Buffer allocation in flow-shop-type production systems with general arrival and service patterns. *Computer & Operations Research* 29(2), 103–121.
- Jeong, K.-C. and Y.-D. Kim (2000). Heuristics for selecting machines and determining buffer capacities in assembly systems. *Computers & Industrial Engineering* 38(3), 341–360.

- Kim, J. and S. B. Gershwin (2005). Integrated quality and quantity modeling of a production line. *OR Spectrum* 27(2–3), 287–314. Reprinted in *Stochastic Modeling of Manufacturing Systems — Advances in Design, Performance Evaluation, and Control Issues*, edited by George Liberopoulos, Chrissoleon T. Papadopoulos, Barış Tan, J. MacGregor Smith, and Stanley B. Gershwin, Springer, 2006.
- Levantesi, R., A. Matta, and T. Tolio (2001). A new algorithm for buffer allocation in production lines. In *The Third Aegean International Conference on Design and Analysis of Manufacturing Systems*, Tinos Island, Greece.
- Lim, J.-T., S. M. Meerkov, and F. Top (1990). Homogeneous, asymptotically reliable serial production line: theory and a case study. *IEEE Transaction on Automatic Control* 35(5), 524–534.
- Meester, L. E. and J. G. Shanthikumar (1990). Concavity of the throughput of tandem queueing systems with finite buffer storage space. *Advances in Applied Probability* 22(3), 764–767.
- Park, T. (1993). A two-phase heuristic algorithm for determining buffer sizes of production lines. *International Journal of Production Research* 31(3), 613–631.
- Schor, J. E. (1995, May). Efficient algorithms for buffer allocation. Mit eecs ms thesis, Massachusetts Institute of Technology. Also available as MIT Laboratory for Manufacturing and Productivity report LMP-95-006.
- Seong, D., S. Y. Chang, and Y. Hong (1994). An algorithm for buffer allocation with linear resource constraints in a continuous flow production line. Technical report, Department of Industrial Engineering, POSTECH, Korea. Technical Report IE-TR-94-05.
- Shanthikumar, J. G. and D. D. Yao (1989a). Monotonicity and concavity properties in cyclic queueing networks with finite buffers. In H. G. Perros and T. Altıok (Eds.), *Queueing networks with blocking*, pp. 325–344. Elsevier Science.
- Shanthikumar, J. G. and D. D. Yao (1989b). Optimal buffer allocation in a multicell system. *International Journal of Flexible Manufacturing Systems* 1(4), 347–356.
- Shi, L. and S. Men (2003). Optimal buffer allocation in production lines. *IIE Transactions* 35(1), 1–10.
- Smith, J. M. and F. R. B. Cruz (2005). The buffer allocation problem for general finite buffer queueing networks. *IIE Transactions* 37(4), 343–365.
- So, K. C. (1997). Optimal buffer allocation strategy for minimizing work-in-process inventory in unpaced production lines. *IIE Transactions* 29(1), 81–88.
- Tempelmeier, H. (2003). Practical considerations in the optimization of flow production systems. *International Journal of Production Research* 41(1), 149–170.
- Werner, L. (2001). Analysis and design of closed loop manufacturing systems. Master’s thesis, Massachusetts Institute of Technology Operations Research Center.

Zhang, J. Z. (2006). *Analysis and Design of Manufacturing Systems with Multiple-Loop Structures*. Ph. D. thesis, Massachusetts Institute of Technology Department of Mechanical Engineering.

Appendix A

The continuous variable version of the solution of the deterministic two-machine line is presented in this section. Suppose we have a two-machine line with parameters r_1 , p_1 , r_2 , and p_2 , and the buffer size is N . In the two-machine case, the state of the system is $s = (n, \alpha_1, \alpha_2)$ when n is the buffer level ($0 \leq n \leq N$) and α_i is the repair state of Machine i ($i = 1, 2, \alpha_i = 0, 1$). $p(n, \alpha_1, \alpha_2)$ stands for the steady-state probability of that state. Gershwin (1994) showed that the steady-state probability distribution is:

$$p(0, 0, 0) = 0 \quad (\text{A.1})$$

$$p(0, 0, 1) = CX \frac{r_1 + r_2 - r_1 r_2 - r_1 p_2}{r_1 p_2} \quad (\text{A.2})$$

$$p(0, 1, 0) = 0 \quad (\text{A.3})$$

$$p(0, 1, 1) = 0 \quad (\text{A.4})$$

$$p(1, 0, 0) = CX \quad (\text{A.5})$$

$$p(1, 0, 1) = CXY_2 \quad (\text{A.6})$$

$$p(1, 1, 0) = 0 \quad (\text{A.7})$$

$$p(1, 1, 1) = \frac{CX}{p_2} \frac{r_1 + r_2 - r_1 r_2 - r_1 p_2}{p_1 + p_2 - p_1 p_2 - r_1 p_2} \quad (\text{A.8})$$

$$p(N-1, 0, 0) = CX^{N-1} \quad (\text{A.9})$$

$$p(N-1, 0, 1) = 0 \quad (\text{A.10})$$

$$p(N-1, 1, 0) = CX^{N-1}Y_1 \quad (\text{A.11})$$

$$p(N-1, 1, 1) = \frac{CX^{N-1}}{p_1} \frac{r_1 + r_2 - r_1 r_2 - p_1 r_2}{p_1 + p_2 - p_1 p_2 - p_1 r_2} \quad (\text{A.12})$$

$$p(N, 0, 0) = 0 \quad (\text{A.13})$$

$$p(N, 0, 1) = 0 \quad (\text{A.14})$$

$$p(N, 1, 0) = CX^{N-1} \frac{r_1 + r_2 - r_1 r_2 - p_1 r_2}{p_1 r_2} \quad (\text{A.15})$$

$$p(N, 1, 1) = 0 \quad (\text{A.16})$$

$$p(n, \alpha_1, \alpha_2) = CX^n Y_1^{\alpha_1} Y_2^{\alpha_2}, 2 \leq n \leq N-2; \alpha_1 = 0, 1; \alpha_2 = 0, 1 \quad (\text{A.17})$$

where C is a normalizing constant and

$$Y_1 = \frac{r_1 + r_2 - r_1 r_2 - r_1 p_2}{p_1 + p_2 - p_1 p_2 - p_1 r_2} \quad (\text{A.18})$$

$$Y_2 = \frac{r_1 + r_2 - r_1 r_2 - p_1 r_2}{p_1 + p_2 - p_1 p_2 - r_1 p_2} \quad (\text{A.19})$$

$$X = \frac{Y_2}{Y_1} \quad (\text{A.20})$$

For convenience, we define internal states as those states in which $2 \leq n \leq N-2$; while boundary states as those states in which $n = 0, 1, N-1$, or N . We further let p_B be the summation of the steady-state probabilities of all boundary states and p_I be the summation of the steady-state probabilities of all internal states. Then,

$$\begin{aligned} p_B &= \sum_{\text{all boundary states}} p(n, \alpha_1, \alpha_2) \\ &= CX \frac{r_1 + r_2 - r_1 r_2 - r_1 p_2}{r_1 p_2} + CX + CXY_2 + \frac{CX}{p_2} \frac{r_1 + r_2 - r_1 r_2 - r_1 p_2}{p_1 + p_2 - p_1 p_2 - r_1 p_2} \\ &\quad + CX^{N-1} + CX^{N-1}Y_1 + \frac{CX^{N-1}}{p_1} \frac{r_1 + r_2 - r_1 r_2 - p_1 r_2}{p_1 + p_2 - p_1 p_2 - p_1 r_2} \\ &\quad + CX^{N-1} \frac{r_1 + r_2 - r_1 r_2 - p_1 r_2}{p_1 r_2} \end{aligned} \quad (\text{A.21})$$

In addition, we calculate p_I in the continuous variable version by

$$p_I = \sum_{\text{all internal states}} p(n, \alpha_1, \alpha_2) = \begin{cases} C \frac{X^{N-1} - X^2}{X - 1} (1 + Y_1)(1 + Y_2) & \text{if } |X - 1| > \delta \\ C(N - 3)(1 + Y_1)(1 + Y_2) & \text{if } |X - 1| \leq \delta \end{cases} \quad (\text{A.22})$$

where δ is a very small non-zero positive value. Let p_T be the summation of all steady-state probabilities of both boundary states and internal states, then

$$p_T = p_B + p_I = 1 \quad (\text{A.23})$$

To find C , let L_B be p_B/C and L_I be p_I/C . L_B and L_I can be calculated directly by the machine parameters of the line and the buffer size N . Since p_B and p_I must sum to 1, then

$$C = \frac{1}{L_B + L_I} \quad (\text{A.24})$$

The production rate of the line can be calculated by

$$\begin{aligned} P &= \frac{r_1}{r_1 + p_1} (1 - p_b) \\ &= \frac{r_1}{r_1 + p_1} (1 - p(N, 1, 0)) \\ &= \frac{r_1}{r_1 + p_1} \left(1 - CX^{N-1} \frac{r_1 + r_2 - r_1 r_2 - p_1 r_2}{p_1 r_2} \right) \end{aligned} \quad (\text{A.25})$$

where $p_b = p(N, 1, 0)$ is the probability of blocking.

Finally, let us consider how to calculate the average inventory for the line. We compute the average inventory for the internal states, denoted by $\bar{n}_I$, as

$$\begin{aligned}\bar{n}_I &= \sum_{\text{all internal states}} np(n, \alpha_1, \alpha_2) \\ &= \begin{cases} C \frac{X(-2X + (N-1)X^{N-2} - \frac{X^{N-1}-X^2}{X-1})}{X-1} (1+Y_1)(1+Y_2) & \text{if } |X-1| > \delta \\ \frac{1}{2}CN(N-3)(1+Y_1)(1+Y_2) & \text{if } |X-1| \leq \delta \end{cases} \quad (\text{A.26})\end{aligned}$$

where δ is a very small non-zero positive value. Then, to calculate the average inventory for the line, we need to consider both internal states and boundary states whose steady-state probability is non-zero and n is non-zero. Then, the average inventory of the line is calculated as

$$\begin{aligned}\bar{n} &= \sum_{\text{all states}} np(n, \alpha_1, \alpha_2) \\ &= p(1, 0, 0) + p(1, 0, 1) + p(1, 1, 1) + \bar{n}_I \\ &\quad + (N-1)(p(N-1, 0, 0) + p(N-1, 1, 0) + p(N-1, 1, 1)) + Np(N, 1, 0)\end{aligned} \quad (\text{A.27})$$

Appendix B

Here, we provide the proofs of our assertion for the case in which some $N_i^* = N_{min}$. Recall that the assertion is:

Assertion The constrained problem

$$\begin{aligned}\max \quad & J(N_1, \dots, N_{k-1}) = A'P(N_1, \dots, N_{k-1}) - \sum_{i=1}^{k-1} b_i N_i - \sum_{i=1}^{k-1} c_i \bar{n}_i \\ \text{subject to} \quad & P(N_1, \dots, N_{k-1}) \geq P^* \\ & N_i \geq N_{min}, \forall i = 1, \dots, k-1\end{aligned}$$

has the same solution for all A' in which the solution of the unconstrained problem (5) has $P' < P^*$.

Let B be $\{i | N_i^* = N_{min}\}$. Hence, $N_i^* = N_{min}, \forall i \in B$, while $N_i^* > N_{min}, \forall i \notin B$. In this case, those N_i^* equal to N_{min} are on the boundary of the feasible region of the optimal solution. By condition (15), we know that $\mu_i^* = 0, \forall i \notin B$. Hence, we simplify the KKT conditions (12) to (15)

to

$$-\begin{pmatrix} \frac{\partial J(\mathbf{N}^*)}{\partial N_1} \\ \frac{\partial J(\mathbf{N}^*)}{\partial N_2} \\ \vdots \\ \frac{\partial J(\mathbf{N}^*)}{\partial N_{k-1}} \end{pmatrix} - \mu_0^* \begin{pmatrix} \frac{\partial P(\mathbf{N}^*)}{\partial N_1} \\ \frac{\partial P(\mathbf{N}^*)}{\partial N_2} \\ \vdots \\ \frac{\partial P(\mathbf{N}^*)}{\partial N_{k-1}} \end{pmatrix} - \sum_{i \in B} \mu_i^* e_i = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \quad (\text{B.1})$$

where

$$e_i = \begin{pmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{pmatrix}$$

in which the 1 is in the i th position.

$$\mu_0^*(P^* - P(\mathbf{N}^*)) = 0, \quad (\text{B.2})$$

$$\mu_i^*(N_{min} - N_i^*) = 0, \forall i = 1, \dots, k-1 \quad (\text{B.3})$$

where $\mu_i^* \geq 0, \forall i = 0, \dots, k-1$. We first argue that not all $N_i^*, i \notin B$ satisfy $\partial J(\mathbf{N}^*)/\partial N_i = 0$. This is because if $\partial J(\mathbf{N}^*)/\partial N_i = 0$ for all $N_i^*, i \notin B$ and all other $N_i^* = N_{min}, i \in B$, then the solution of the constrained problem $(N_1^*, \dots, N_{k-1}^*)$ is equivalent to the solution of the unconstrained problem, since it is just constrained by the buffer size constraint. This is a conflict. Therefore, it is not possible for all $N_i^*, i \notin B$ to satisfy $\partial J(\mathbf{N}^*)/\partial N_i = 0$. So, $\exists \hat{i}$ for which $N_{\hat{i}}^* > N_{min}$ and $\partial J(\mathbf{N}^*)/\partial N_{\hat{i}} \neq 0$. Since $N_{\hat{i}}^* > N_{min}$, by condition (B.3), we know $\mu_{\hat{i}}^* = 0$. Thus, $\mu_0^* \neq 0$ since otherwise condition (B.1) would be violated. Hence, by condition (B.2), we know that the optimal solution $\mathbf{N}^*$ satisfies $P(\mathbf{N}^*) = P^*$. In this case, there are more than one active inequality constraints: the production rate constraint $g_0(\mathbf{N})$ and buffer size constraints $g_i(\mathbf{N}), \forall i \in B$. It is not hard to see they are linearly independent at $\mathbf{N}^*$ so the optimal solution is regular.

As before, replacing μ_0^* by $\mu_0 > 0$ and $\mu_i^*, \forall i \in B$ by $\mu_i > 0, \forall i \in B$ in constraint (B.1) gives

$$-\nabla J(\bar{\mathbf{N}}) + \mu_0 \nabla(P^* - P(\bar{\mathbf{N}})) + \sum_{i \in B(i)} \mu_i \nabla(N_{min} - \bar{N}_i) = 0, \quad (\text{B.4})$$

where $\bar{\mathbf{N}}$ is the unique solution. Note that $\bar{\mathbf{N}}$ is exactly the optimal solution of the following optimization problem

$$\begin{aligned} \min \quad & -\bar{J}(\mathbf{N}) = -J(\mathbf{N}) + \mu_0(P^* - P(\mathbf{N})) + \sum_{i \in B} \mu_i(N_{min} - N_i) \\ \text{subject to} \quad & N_{min} - N_i \leq 0, \forall i = 1, \dots, k-1, \end{aligned} \quad (\text{B.5})$$

$$N_i = N_{min}, \forall i \in B.$$

which is equivalent to

$$\begin{aligned}
& \max \quad \bar{J}(\mathbf{N}) = J(\mathbf{N}) - \mu_0(P^* - P(\mathbf{N})) \\
& \text{subject to} \quad N_{\min} - N_i \leq 0, \forall i = 1, \dots, k-1, \\
& \quad \quad \quad N_i = N_{\min}, \forall i \in B.
\end{aligned} \tag{B.6}$$

or

$$\begin{aligned}
& \max \quad \bar{J}(\mathbf{N}) = (A + \mu_0)P(\mathbf{N}) - \sum_{i=1}^{k-1} b_i N_i - \sum_{i=1}^{k-1} c_i \bar{n}_i \\
& \text{subject to} \quad N_i \geq N_{\min}, \forall i = 1, \dots, k-1, \\
& \quad \quad \quad N_i = N_{\min}, \forall i \in B.
\end{aligned} \tag{B.7}$$

or, finally,

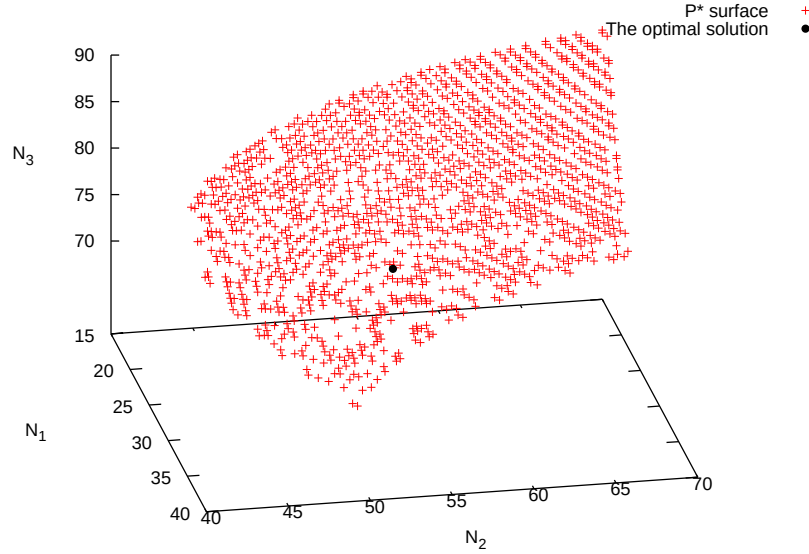
$$\begin{aligned}
& \max \quad \bar{J}(\mathbf{N}) = A'P(\mathbf{N}) - \sum_{i=1}^{k-1} b_i N_i - \sum_{i=1}^{k-1} c_i \bar{n}_i \\
& \text{subject to} \quad N_i \geq N_{\min}, \forall i = 1, \dots, k-1, \\
& \quad \quad \quad N_i = N_{\min}, \forall i \in B,
\end{aligned} \tag{B.8}$$

where $A' = A + \mu_0$. Again, this is exactly our unconstrained problem in which A is replaced by A' , and $\bar{\mathbf{N}}$ is its optimal solution, except that in the optimal solution, $\bar{N}_i = N_{\min}, \forall i \in B$. A similar argument as in Section 3.1 could prove our assertion for this case.

Appendix C

We provide a brief discussion about the P^* surface search here. All buffer size allocations, $\mathbf{N}$, such that $P(\mathbf{N}) = P^*$ compose the P^* surface. For instance, consider a four-machine line with parameters $r_1 = .11, p_1 = .008, r_2 = .12, p_2 = .01, r_3 = .1, p_3 = .01, r_4 = .09$, and $p_4 = .01$. $P^* = .88$. A portion of the P^* surface of this example is shown in Figure C.1. It means that all points on this surface satisfy $P(\mathbf{N}) = P^*$. We further let the profit is $J(\mathbf{N}) = 2000P(\mathbf{N}) - \sum_{i=1}^3 N_i - \sum_{i=1}^3 \bar{n}_i$. Then, the optimal solution achieved by our algorithm is also shown in this figure. To verify the optimal solution, we conduct a search on this P^* surface. It is obvious that the P^* surface is boundless and it is impractical to search the whole surface, even for the simplest three-machine line. Therefore, in our implementation, we only search around the optimal solution achieved by our algorithm. Since there is only one maximum, searching around the optimal solution is accurate enough.

In the P^* surface search, we first need to find all feasible points on the surface around the optimal solution gained by our algorithm. To do this, we set ranges for all buffers and then at each time we vary one buffer size by a small step size (0.01 or 0.1) while keeping other buffers unchanged. Therefore, we can calculate the production rates and the profits for all combinations of

Figure C.1: P^* surface of the example

those buffers. All those points, whose production rate P satisfy $|P - P^*| \leq \delta$ where δ is a very small non-zero positive value, compose the P^* surface. Then we compare the profits of those points. The point having the maximal profit in the P^* search becomes the optimal solution by this method. Then, we compare our optimal solution with this one and calculate the error.