Alternatives to the Gradient in Optimal Transfer Line Buffer Allocation

by

Ketty Tanizar

B.S., Industrial Engineering and Operations Research
University of California at Berkeley, CA, 2002

Submitted to the Sloan School of Management
in partial fulfillment of the requirements for the degree of

Master of Science in Operations Research

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2004

Author . . . .
Sloan School of Management
August 13, 2004

Certified by . . . .
Stanley B. Gershwin
Senior Research Scientist, Department of Mechanical Engineering
Thesis Supervisor

Accepted by . . . .
James B. Orlin
Co-Directors, Operations Research Center

Alternatives to the Gradient in Optimal Transfer Line Buffer Allocation

by

Ketty Tanizar

Submitted to the Sloan School of Management
on August 13, 2004, in partial fulfillment of the
requirements for the degree of
Master of Science in Operations Research

Abstract

This thesis describes several directions to replace the gradient in Schor's gradient algorithm to solve the dual problem. The alternative directions are: the variance and standard deviation of buffer levels, the difference between the average buffer level from half-full, the difference between probability of starvation and blocking, and the difference between the production rate upstream and downstream of a buffer. The objective of the new algorithms is to achieve the final production rate of the dual problem faster than the gradient. The decomposition method is used to evaluate the production rates and average buffer levels. We find that the new algorithms work better in lines with no bottlenecks. The variance and standard deviation algorithms work very well in most cases.

Thesis Supervisor: Stanley B. Gershwin
Title: Senior Research Scientist, Department of Mechanical Engineering

Acknowledgments

First I would like to thank Dr. Stanley Gershwin for being a terrific advisor and a great friend. I thank him for his trust in me and for constantly challenging me.

I would like to thank Chiwon Kim, Youngjae Jang, Jongyoon Kim, and James for being great friends and colleagues. I will always be grateful for all the discussions we had and most of all for our beautiful friendship.

I would also thank my fellow ORC S.M. students: Kendell Timmers, Christopher Jeffreys, Corbin Koepke, and Sepehr Sarmadi for being the best study mates I could ever ask for. Our study sessions and stimulating discussions were certainly one of the most enjoyable parts of my M.I.T. experience.

I want to give special thanks to Hendrik, Leonard, Doris, and Liwen for always taking the time to listen and for accepting me the way I am. Also I would like to thank Rudy and Andrea for making me laugh. I could not ask for better friends.

In closing, I would thank my mother, my father, and my sister Nelly for their unfaltering love and support. They make this all worth while.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 General Problem Description

In this paper, we focus on buffer optimization of a production line. A *production line*, or *flow line*, or *transfer line* is a manufacturing system where machines or work centers are connected in series and separated by buffers. Materials flow from the upstream portion of the line to the first machine, from the first machine to the first buffer, from the first buffer to the second machine, and continue on to the downstream portion of the line.

There are two types of buffer optimization problems. The first one, or the *primal* problem, is to minimize the total buffer space in the production line while trying to achieve a production rate target. This problem is usually encountered when the available factory space is limited. The other problem, or the *dual* problem, is to maximize production rate subject to a specified total buffer space. The second problem is encountered when floor space is not a problem and the major goal is to achieve as high production rate as possible. In both problems, the size of each buffer becomes the decision variable.

The dual problem is solved using a gradient method and the primal problem is solved using the solution of the dual problem. Schor uses a gradient method to solve the dual problem and solve the primal problem using the dual solution [Schor00]. Although the gradient approach proves to be very efficient, it can be time-consuming.

We propose using different approaches that are based on variability of buffer levels to replace the gradient approach.

1.2 Approach

In this paper, we focus on solving the dual problem. Therefore, the question we are asking is: given the machines of a transfer line and a total buffer space to be allocated, how should we allocate buffer space so that the production rate is maximized? We begin our paper by describing some work in the area of buffer optimization (Chapter 1).

In Chapter 2, we define some parameters and notation used throughout the paper. We also mention some qualitative properties assumed about transfer lines. Finally, we describe the primal and the dual problem quantitatively and review the Schor's gradient algorithm for solving the dual problem.

The purpose of the paper is to examine some alternative directions, other than the gradient, to use in solving the dual problem. In Chapter 3, we mention an intuitive justification and motivation for using the alternative directions. We also include some prior work, which supports the use of the alternative directions to replace the gradient. The new algorithms are also derived in Chapter 3.

We review the performance of the new algorithms, in terms of accuracy, reliability, and speed, in Chapter 4. In Chapter 5, we perform a sensitivity analysis of the new algorithms if the machine parameters are varied by a small amount. Chapter 6 discusses the conclusion and future research.

We also include two appendices. Appendix A describes an algorithm to generate realistic line parameters. This algorithm generates $r, p,$ and μ based on realistic constraints set on them. It avoids any filtering and is, therefore, fast and reliable. Appendix B describes the linear search method used in the new algorithms.

1.3 Literature Review

1.3.1 Qualitative Properties of the Line Performance

In order to develop algorithms relevant to transfer lines, we need to understand the behavior of the lines. One aspect of the behavior is how the production rate of the line is affected by changes in buffer sizes. Adan and Van Der Wal [Adan89] showed that the production rate increases as each buffer size is increased. However, the increase in production rate becomes less significant as the buffer size increases. This is shown by Meester and Shanthikumar [Meester90], who proved that the production rate is an increasing, concave function of the buffer sizes.

1.3.2 Characteristics of the Solutions

The purpose of this paper is to propose alternative directions that can be used to replace the gradient in Schor's Dual Algorithm [Schor00]. One paper that motivates two alternative directions used in this paper is the paper by Jacobs, Kuo, Lim and Meerkov [Jacobs96]. Jacobs *et al.* did not suggest a method to design a manufacturing system, but instead proposed a method of improving an existing system using data that is available as the system runs. They used the concept of "improvability" (similar to "optimality" but used in the context when optimality may be impossible due to the lack of precise information on the factory floor) to determine how buffer space should be allocated. Jacobs *et al.* showed that a transfer line is unimprovable with respect to work force if, each buffer is, on the average, half-full and if the probability that Machine M_i is blocked equals the probability that Machine M_{i+1} is starved.

1.3.3 Some Solutions Approaches

One approach to buffer size optimization is done by means of *perturbation analysis.* Ho, Eyler, and Chien [Ho79] were pioneers of this simulation-based technique. In this paper, Ho *et al.* estimated the gradient of the production rate of a discrete-event dynamic system (one with discrete parts, identical constant processing times, and geometrically distributed repair and failure times) with respect to all buffer sizes, using a single simulation run.

Another approach is by means of *dynamic programming.* Chow [Chow87] developed a dynamic programming approach to maximize production rates subject to a total buffer space constraint.

A branch and bound approach has also been used to solve buffer allocation problems. Park [Park93] proposed a two-phase heuristic method using a dimension reduction strategy and a beam search method. He developed a heuristic method to minimize total buffer storage while satisfying a desired throughput rate.

Another approach to buffer allocation problem is by Spinellis and Papadopoulos [Spinellis00]. They compared two stochastic approaches for solving buffer allocation problem in large reliable production lines. One approach is based on simulated annealing. The other one is based on genetic algorithms. They concluded that both methods can be used for optimizing long lines, with simulated annealing producing more optimal production rate and the genetic algorithm leading in speed.

Finally, an approach closest to our algorithm is Gershwin and Schor's Gradient Algorithm [Schor00]. They used a gradient method to solve the problem of maximizing production rate subject to a total buffer space constraint (the dual problem) and used the solution of the dual problem to solve the primal problem.

1.3.4 Preliminary Work

The intuition behind using the alternative directions to substitute the gradient is supported by the following preliminary numerical and simulation studies:

1. A study about the relationship between optimum buffer allocation and buffer level variance in a finite buffer line

   In this study [Kinsey02], several machine lines with different machine parameters and efficiencies are generated. The different machines are generated by starting with some center values for r, p, and μ. Later, a random number generator was used to produce some variations for r, p, and μ around their center values. In some of the lines, a bottleneck was introduced by imposing a particularly low efficiency on a selected machine in the line. Finally, the correlation between the optimal buffer allocation and standard deviation of buffer level is calculated.

The following graphs show the optimal buffer sizes and the standard deviation of the buffer levels for three different lines.

Figure 1-1 shows the result of the study for a twenty-identical-machine line with a 60% efficiency level. The R^2 value of the correlation between the optimal buffer sizes and the standard deviation of the buffer levels is 0.99.

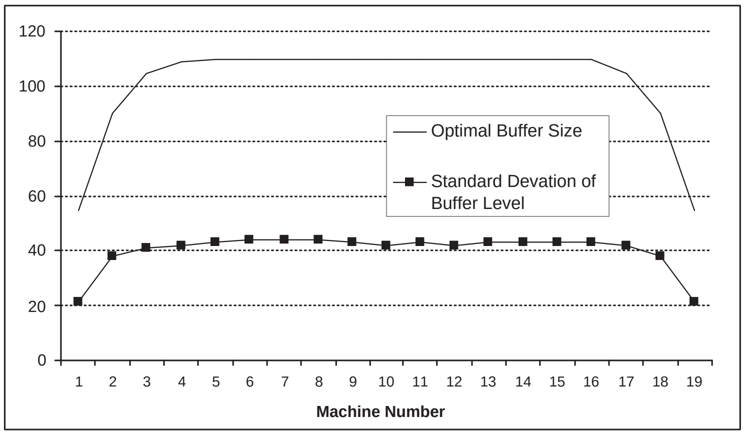


Figure 1-1: Test Result: 60% Efficiency with 0% Variations in r and p Values

Figure 1-2 shows the result for a twenty-machine line with a 96% efficiency level and with 15% variations in r and p values. The R^2 value is 0.75.

Figure 1-3 shows the result for a twenty-machine line with a 96% efficiency level and with 15% variations in r and p values. For this line, Machine M_{11} is the bottleneck. The R^2 value of the correlation is 0.98.

This study concludes that the standard deviation of the buffer levels are well-correlated with the sizes of the buffers in optimized lines. In addition, bottlenecks in the machine line do not affect the correlation. This confirms the hypothesis that the best place to put space is where variation is the greatest.

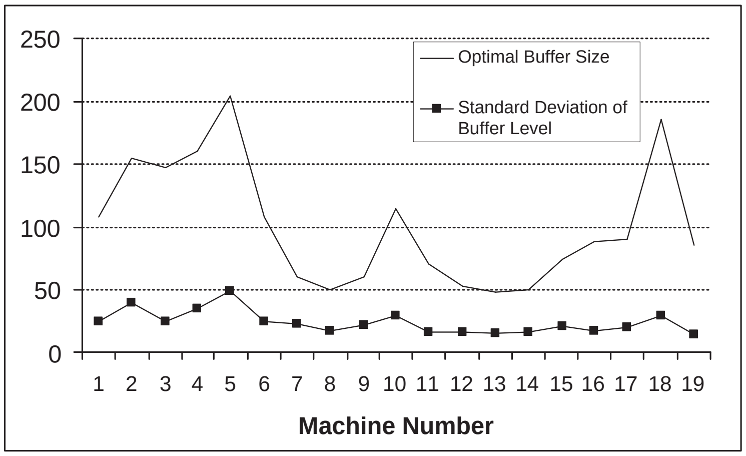


Figure 1-2: Test Result: 96% Efficiency with 15% Variations in r and p Values

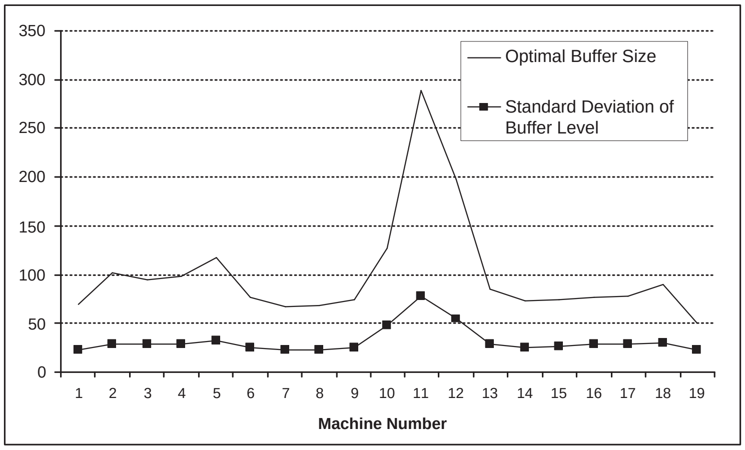


Figure 1-3: Test Result: 91% Efficiency with Bottleneck and with 15% Variations in r and p Values

2. A study about the values of all directions at the optimal buffer allocations calculated using the gradient method

   In this study, the values of the gradient, variance, standard deviation of buffer levels, $|\overline{n_i} - \frac{N_i}{2}|$, $|p_b(i) - p_s(i)|$, $|P_u(i) - P_d(i)|$, $\frac{1}{|\overline{n_i} - \frac{N_i}{2}|}$, $\frac{1}{|p_b(i)-p_s(i)|}$, and $\frac{1}{|P_u(i)-P_d(i)|}$ are calculated at the optimal buffer allocation, obtained by the Gradient Algorithm.

   Figure 1-4 shows an example of the normalized values of the gradient, variance, standard deviation, $|\overline{n_i} - \frac{N_i}{2}|$, $|p_b(i) - p_s(i)|$, $|P_u(i) - P_d(i)|$ at the optimal point. Figure 1-5 shows an example of the normalized values of $\frac{1}{|\overline{n_i} - \frac{N_i}{2}|}$, $\frac{1}{|p_b(i)-p_s(i)|}$, and

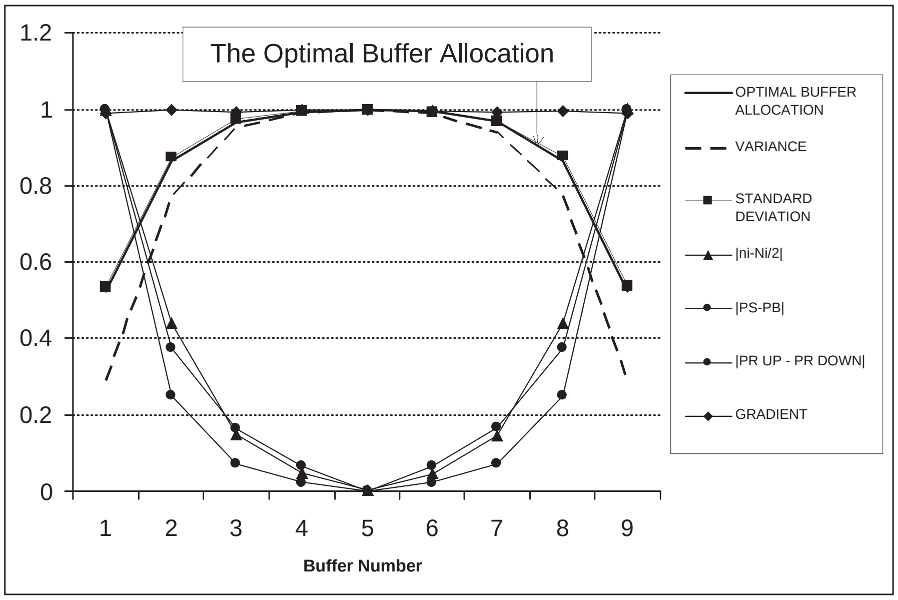


Figure 1-4: Values of the Gradient, Variance, Standard Deviation, $|\overline{n_i} - \frac{N_i}{2}|$, $|p_b(i) - p_s(i)|$, and $|P_u(i) - P_d(i)|$ at the Optimal Buffer Allocation

$\frac{1}{|P_u(i)-P_d(i)|}$ at the optimal point.

Table 1.1 shows the data used to generate the results shown in Figure 1-4 and Figure 1-5.

Table 1.1: The Line Parameters for Figure 1-4 and Figure 1-5

Number of Machines	10 identical machines
Total Buffer Spaces to be Allocated	900
Repair Rate r	0.015
Failure Rate p	0.01
Processing Rate μ	1

Figure 1-4 and Figure 1-5 show that the gradient, variance, standard deviation of buffer levels, $\frac{1}{|\overline{n_i}-\frac{N_i}{2}|}$, $\frac{1}{|p_b(i)-p_s(i)|}$, and $\frac{1}{|P_u(i)-P_d(i)|}$ have the same shapes as the shape of the optimal buffer allocation. Therefore, these directions can be used to allocate buffer sizes.

Figure 1-4 shows that $|\overline{n_i}-\frac{N_i}{2}|$, $|p_b(i)-p_s(i)|$, and $|P_u(i)-P_d(i)|$ have shapes that are inverse the shape of the optimal buffer allocation. Although their shapes are inverse the shape of the optimal buffer allocation, $|\overline{n_i} - \frac{N_i}{2}|$, $|p_b(i) - p_s(i)|$,

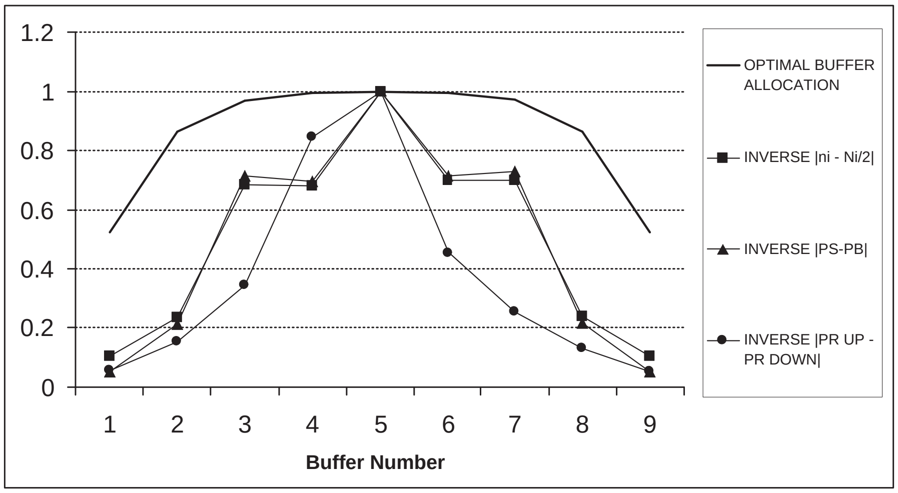


Figure 1-5: Values of $\frac{1}{|\overline{n_i}-\frac{N_i}{2}|}$, $\frac{1}{|p_b(i)-p_s(i)|}$, and $\frac{1}{|P_u(i)-P_d(i)|}$ at the Optimal Buffer Allocation

and $|P_u(i) - P_d(i)|$ can also be used to allocate buffer sizes. This is because the scalar a obtained from the $|\overline{n_i} - \frac{N_i}{2}|$, $|p_b(i) - p_s(i)|$, and $|P_u(i) - P_d(i)|$ directions are in the opposite direction of the scalar a obtained from the $\frac{1}{|\overline{n_i}-\frac{N_i}{2}|}$, $\frac{1}{|p_b(i)-p_s(i)|}$, and $\frac{1}{|P_u(i)-P_d(i)|}$ directions. Therefore, the direction of the movement, or $a\prod$ in the formula $N_{new} = N + a\prod$ will be in the direction towards N_{new}.

Chapter 2

Technical Problem Statement

2.1 Definitions and Properties

2.1.1 Parameters and Notation

In this research, we attempt to find the optimal buffer allocation of a production line, such that the production rate P of the line is maximized. The production line has k machines and $k - 1$ buffers. N_i is the size of Buffer B_i and $\overline{n_i}$ is the average buffer level of Buffer B_i. The production rate of the line P is a function of the buffer sizes $(N_1, ..., N_{k-1})$. Therefore, it is sometimes denoted as $P(N_1, N_2, N_3, ..., N_{k-1})$. Figure 2-1 shows a diagram of a transfer line with k = 3, in which squares represent machines, circles represent buffers, and arrows represent the work flow from the upstream portion of the line to the downstream portion of the line.

$$\begin{tikzcd}\boxed{M1} \arrow[r] & B1 \arrow[r] & \boxed{M2} \arrow[r] & B2 \arrow[r] & \boxed{M3}\end{tikzcd}$$

Figure 2-1: Transfer Line

When any machine upstream of Buffer B_i breaks down, that machine can not produce any parts. Eventually there are no parts coming into Buffer B_i so Buffer

B_i will be empty. Since Buffer B_i is empty, there are no parts to be processed by machines downstream of Buffer B_i. This phenomenon is called **starvation**. Likewise, when any machine downstream of Buffer B_i breaks down, it stops producing parts. Eventually Buffer B_i will be full because it can not transfer parts to the machine downstream of it. This causes the machine upstream of Buffer B_i stop transferring parts to Buffer B_i because Buffer B_i can not hold the parts. This phenomenon is called **blockage**.

2.1.2 Qualitative Properties

We assume the following properties:

- Continuity

  A small change in N_i results in a small change in P.

- Monotonicity

  The production rate P increases monotonically as N_i increases, provided all other quantities are held constant.

- Concavity

  The production rate P is a concave function of $(N_1, ..., N_{k-1})$.

2.2 Model Type and Problem Types

2.2.1 Model Type

There are three types of manufacturing systems models:

- Discrete Material, Discrete Time
- Discrete Material, Continuous Time
- Continuous Material Flow

The details of each model can be found in [Gershwin94]. In this paper, we use the continuous material flow model.

In the continuous flow model, material is treated as though it is a continuous fluid. Machines have deterministic processing time but they do not need to have equal processing times. The failure and repair times of machines are exponentially distributed. There are three machine parameters in this model. The first one is the failure rate p_i. The quantity $p_i\delta$ is the probability that Machine M_i fails during an interval of length δ while it is operating and the upstream buffer is not empty and downstream is not full. The second parameter is the repair rate r_i. The quantity $r_i\delta$ is the probability that the Machine M_i gets fixed while it is under repair during an interval of length δ. Finally, the last parameter is the operation rate μ_i. The quantity μ_i is the processing speed of Machine M_i when it is operating, not starved or blocked, and not slowed down by an adjacent machine. The quantity $\mu_i\delta$ is the amount of material processed by M_i during an interval of length δ. As in the discrete material models, the size of Buffer B_i is N_i. All μ_i, p_i, r_i, and N_i are finite, nonnegative real numbers.

The isolated efficiency of Machine M_i, denoted by e_i, is the efficiency of Machine M_i independent of the rest of the line. The isolated efficiency e_i is defined mathematically as

$$e_i = \frac{r_i}{r_i + p_i}$$

In the continuous flow model, μ_i is the processing rate of Machine M_i when it is operating, not starved or blocked, and not slowed down by an adjacent machine. Since Machine M_i's actual production rate is influenced by its efficiency, the isolated production rate ρ_i of Machine M_i is defined as

$$\rho_i = \mu_i e_i$$

2.2.2 Decomposition Algorithm

When there are more than two machines in a transfer line and all N_i are neither infinite or zero, the production rate of the line and the average inventory levels of buffers can not be calculated analytically. This is where the Decomposition Algorithm plays an important role to approximate the production rate and the average inventory levels, which otherwise can not be calculated. More specifically, ADDX algorithm [Burman95] is used to evaluate P and $\overline{n_i}$ when the system is modeled using the continuous flow model. The ADDX algorithm uses analytical methods and decomposition method [Gershwin87] to determine approximate solutions.

The following is a brief description of the Decomposition Algorithm. Decomposition Algorithm approximates the behavior of a k-machine line with $k-1$ hypothetical two-machine lines, as illustrated in Figure 2-2 for $k=4$. $L(i)$ represents the hypothetical two-machine line i. The parameters of $L(i)$ are estimated such that the flow of materials into Buffer B_i in $L(i)$ approximates the flow of materials into Buffer B_i in the original line L.

Machine $M_u(i)$ and Machine $M_d(i)$ represents the machine upstream and machine downstream of Buffer B_i in the artificial two-machine line model. The quantity $r_u(i), i=1,...,k-1$, represents the repair rate of $M_u(i)$. The quantity $r_d(i)$ represents the repair rate of $M_d(i)$. $p_u(i)$ represents the failure rate of Machine $M_u(i)$ and $p_d(i)$ represents the failure rate of Machine $M_d(i)$. Finally, $\mu_u(i)$ represents the processing rate of Machine $M_u(i)$ and $\mu_d(i)$ be the processing rate of Machine $M_d(i)$. The decomposition method works by calculating $r_u(i)$, $r_d(i)$, $p_u(i)$, $p_d(i)$, $\mu_u(i)$, and $\mu_d(i)$.

The starvation and blockage phenomena in Decomposition Algorithm are defined as the starvation of Machine $M_d(i)$ and the blockage of Machine $M_u(i)$. More specifically, $p_s(i)$ is defined as the probability of starving of Machine $M_d(i)$ and $p_b(i)$ is defined as the probability of blocking of Machine $M_u(i)$.

$P_u(i), i=1,...,k-1$, is defined as the production rate of the line upstream of

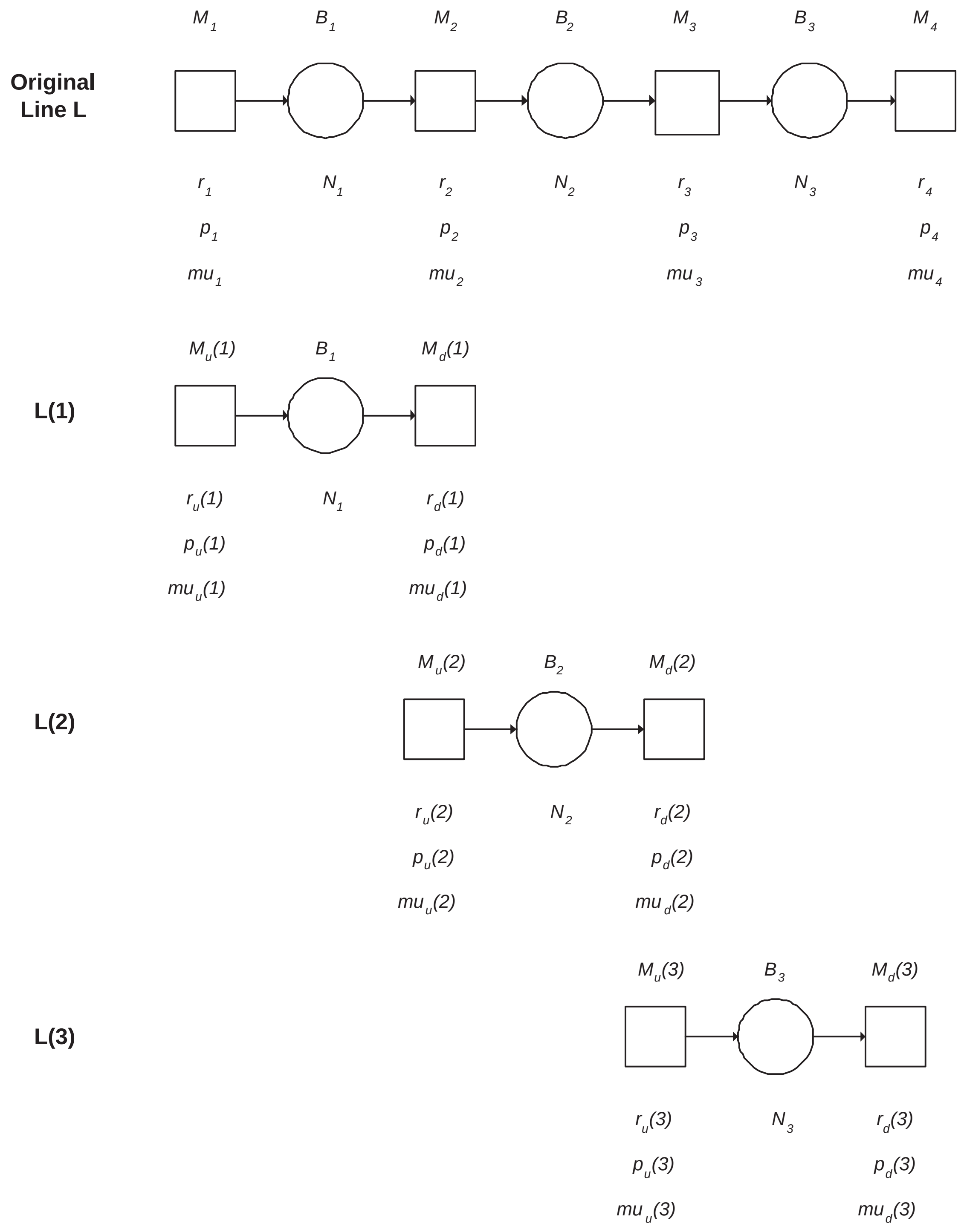


Figure 2-2: Decomposition of Line

Buffer B_i and $P_d(i), i = 1, ..., k-1$, is defined as the production rate of the line downstream of Buffer B_i.

2.2.3 Problem Types

- Primal Problem

  In the primal problem, we minimize the total buffer spaces N^{TOTAL} such that the optimal production rate P is equal to or higher than a specified value P^*. The primal problem is described mathematically as

  $$\text{Minimize } N^{TOTAL} = \textstyle\sum_{i=1}^{k-1} N_i$$

  subject to

  $$P(N_1, ..., N_{k-1}) \geq P^*;$$

  $$P^* \text{ specified}$$

  $$N_i \geq 0, i = 1, ..., k-1$$

  The input to the primal problem are the machine parameters and the specified P^*. The outputs, or the decision variables, are $(N_1, ..., N_{k-1})$ and N^{TOTAL}.

  The primal problem is difficult because the constraint $P(N_1, ..., N_{k-1}) \geq P^*$ is non-linear.

- Dual Problem

  In the dual problem, we are given a fixed N^{TOTAL} and we seek the buffer sizes $(N_1, ..., N_{k-1})$ such that the production rate $P(N_1, ..., N_{k-1})$ is maximized. That is,

  $$\text{Maximize } P(N_1, ..., N_{k-1})$$

subject to

$$N^{TOTAL} = \sum_{i=1}^{k-1} N_i;$$

$$N^{TOTAL} \text{ specified}$$

$$N_i \geq 0, i = 1, ..., k-1$$

The input to this problem are the machine parameters and N^{TOTAL}. The outputs are the optimal $P(N_1, ..., N_{k-1})$ and $(N_1, ..., N_{k-1})$. The dual problem is an easier problem than the primal because the constraint $N^{TOTAL} = \sum_{i=1}^{k-1} N_i$ is a plane.

The solutions to both problems can be found in [Schor00]. Schor uses the dual solution to solve the primal problem. In this paper, we will develop alternative algorithms to solve the dual problem.

2.3 Review of Schor's Dual Algorithm

In [Schor00], Schor invents an efficient algorithm, which is based on a gradient method, to solve the dual problem. Figure 2-3 shows the block diagram of Schor's Algorithm.

The algorithm starts with specifying an initial condition for buffer spaces. One initial condition that can be used is **equal buffer allocation**, i.e. every buffer is allocated $\frac{N^{TOTAL}}{k-1}$ spaces. After selecting the initial condition, a direction to move in the constraint space, which is the set of all points that satisfy $N^{TOTAL} = \sum_{i=1}^{k-1} N_i$, is determined. A linear search is conducted in that direction until a point which has the maximum production rate is encountered. This new point becomes the next guess. A new search direction is determined for this new guess. The algorithm continues until a terminating criterion is satisfied.

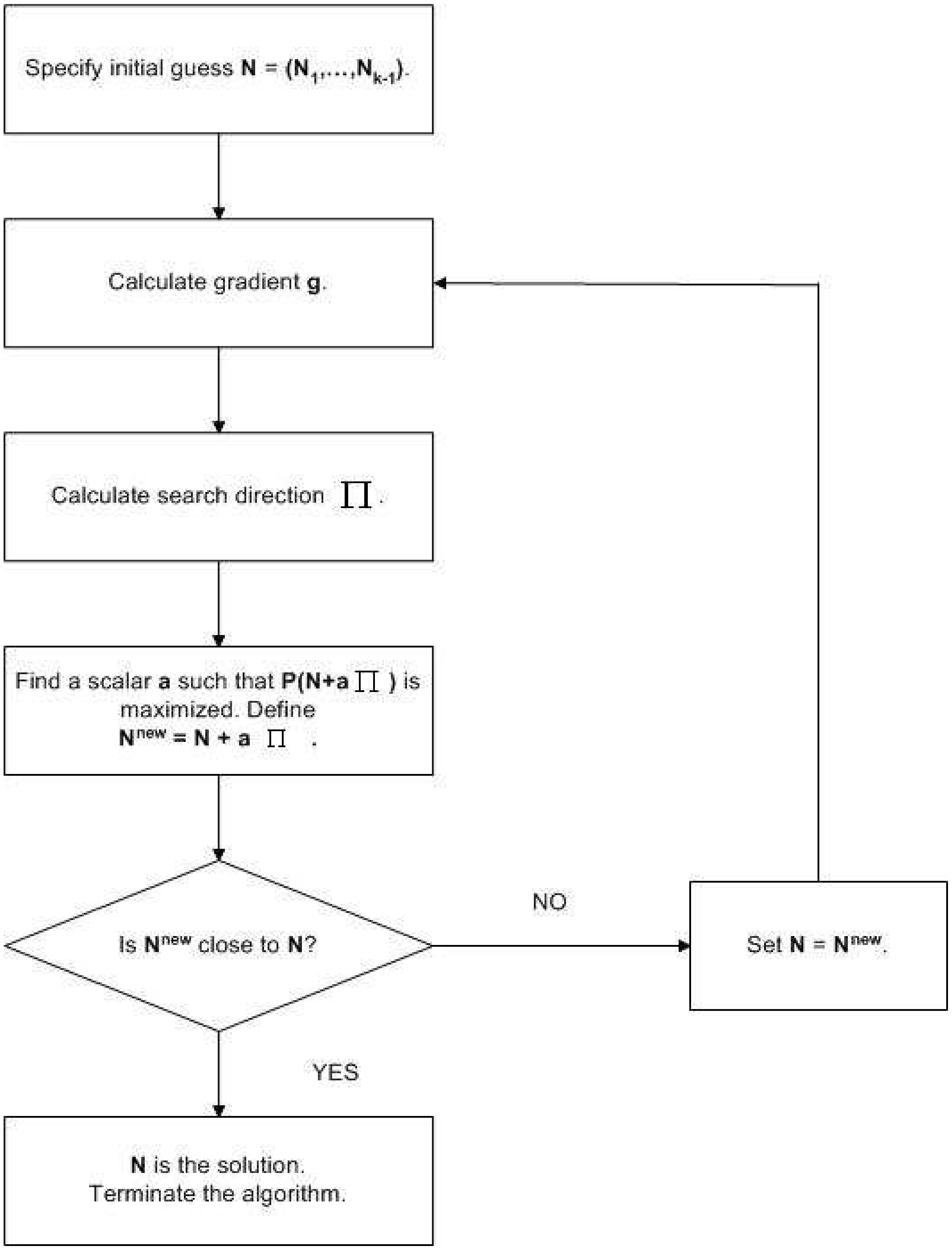

Specify initial guess $\mathbf{N} = (\mathbf{N}_1,\ldots,\mathbf{N}_{k-1})$.
Calculate gradient $\mathbf{g}$.
Calculate search direction $\prod$.
Find a scalar $\mathbf{a}$ such that $\mathbf{P}(\mathbf{N}+\mathbf{a}\prod)$ is maximized. Define $\mathbf{N}^{new} = \mathbf{N} + \mathbf{a}\prod$.
Is $\mathbf{N}^{new}$ close to $\mathbf{N}$?
NO
Set $\mathbf{N} = \mathbf{N}^{new}$.
YES
$\mathbf{N}$ is the solution. Terminate the algorithm.

Figure 2-3: Block Diagram of Schor's Algorithm

To determine the search direction, the gradient $\boldsymbol{g}$ is calculated as follows:

$$g_i = \frac{P(N_1, ..., N_i + \delta N, ..., N_{k-1}) - P(N_1, ..., N_i, ..., N_{k-1})}{\delta N}$$

In order to satisfy the constraint $N^{TOTAL} = \sum_{i=1}^{k-1} N_i$, the gradient needs to be projected onto the constraint space. The projected gradient on the constraint space is called the search direction $\prod$. Let us define

$$\overline{g} = \frac{1}{k-1} \sum_{i=1}^{k-1} g_i$$

$$\prod i = g_i - \overline{g}$$

The next step of the algorithm involves a linear search in the direction of $\prod$. Let N be the current guess and N^{new} be the next guess. N^{new} can be defined mathematically as $N^{new} = N + a \prod$, where N represents the current point on the constraint space and $a \prod$ represents how far we move on the constraint space until we reach N_{new}. The linear search requires finding a scalar a such that N_{new} has the highest production rate of all points along the line formed by $\prod$. The details of the linear search can be found on Appendix B.

Figure 2-4 shows the illustrations of N, the search direction vector $\prod$, and the constraint space for $k = 4$.

The beginning point of the arrow is the current guess N. The arrow itself is vector $\prod$, which is the projected gradient g onto the constraint space $N_1 + N_2 + N_3 = N^{TOTAL}$.

The details of Schor's Gradient Algorithm can be found on [Schor00].

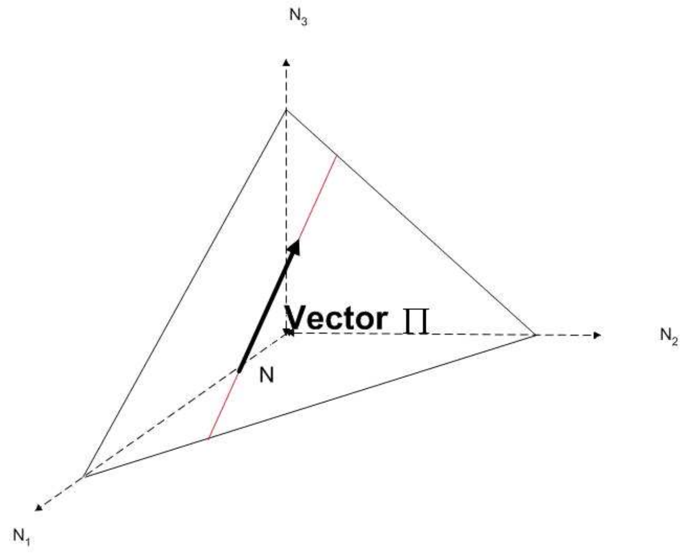


Figure 2-4: Constraint Space

Chapter 3

Variability and Optimality

3.1 Intuitive Justification

The purpose of this research is to find alternative directions that are easier to calculate than the gradient.

Buffers are used to diminish the propagation of disturbances from one part of a production system to the other. This is done by transforming the disturbances into variations of the amount of material in the buffers. If the buffer level does not change very much (e.g., when the upstream part of the system is much faster than the downstream part so the buffer is nearly always full), the buffer is not able to absorb the disturbance. Therefore, if buffer space is to be reallocated, buffers with little variation should get little space, and buffers with more variation should get more. As a consequence, an indication of relative variability can be used to decide buffer space.

The following are some indications of relative variability that we consider for using in place of the gradient:

(a) Variance and standard deviation of buffer levels

  Variance and standard deviation are direct measures of variability.

(b) Deviation of the average buffer level from half-full, or $|\overline{n_i} - \frac{N_i}{2}|$

The intuition behind using $|\overline{n_i} - \frac{N_i}{2}|$, $i = 1, ..., k-1$, as a substitute for the gradient is based on the work of Meerkov and his colleagues [Jacobs96]. In his paper, Meerkov studies the improvability of production line systems. Although his definition of "improvable" appears to be similar to "non-optimal," his goal is not to develop an optimal system. Instead, he seeks to improve the existing production line using the data that is available as the line operates.

Meerkov suggests that a production line is unimprovable with respect to work force if two conditions are satisfied:

i. Each buffer is, on average, half-full.
   This condition motivates the use of $|\overline{n_i} - \frac{N_i}{2}|$ as a substitute for component i of the gradient vector.

ii. $|p_b(i) - p_s(i)|$, $i = 1, ..., k-1$, should be as small as possible.
   This condition motivates the use of $|p_b(i) - p_s(i)|$, which is the next alternative method discussed.

The first condition is each buffer is, on the average, half full. If one buffer (let us call it Buffer B_i) is far from half-full (i.e., it is always nearly full or empty), the system can be improved by reallocating buffer space. If Buffer B_i is nearly always full or empty, or $|\overline{n_i} - \frac{N_i}{2}|$ is large, the buffer level in Buffer B_i does not vary much. Therefore, the capacity of buffer B_i should be reduced. In brief, $|\overline{n_i} - \frac{N_i}{2}|$ or $\frac{1}{|\overline{n_i} - \frac{N_i}{2}|}$ is an indication of relative variability and can be used to replace the gradient.

Unfortunately, many manufacturing systems designers believe that the capacity of buffer that is always full must be increased so that the buffer store more items. However, the variability of buffer that is always full is low and buffer with low variability should get little space. Therefore, good practice is to focus on half-full buffers and then whenever possible, reduce the capacity of usually full buffers and usually empty to increase those of half-full buffers.

(c) Difference between the fraction of time a buffer is empty and the fraction of time it is full, or $|p_b(i) - p_s(i)|$

The intuition behind using $|p_b(i) - p_s(i)|$, $i = 1, ..., k-1$, as a substitute for the gradient is also based on the work of Meerkov and his colleagues [Jacobs96]. Meerkov suggests that the second condition that must be satisfied in order to achieve a well-designed system is that $|p_b(i) - p_s(i)|$ should be as small as possible.

If $|p_b(i) - p_s(i)|$ is large, Buffer B_i is almost always blocked or starved. Therefore, there is little variation in the buffer level. Consequently, the capacity of Buffer B_i should be reduced.

Like $|\overline{n_i} - \frac{N_i}{2}|$ and $\frac{1}{|\overline{n_i} - \frac{N_i}{2}|}$, $|p_b(i) - p_s(i)|$ or $\frac{1}{|p_b(i) - p_s(i)|}$ is also an indication of relative variability and can be used to replace the gradient.

(d) Difference between the production rate of the part of the line upstream and the part of the line downstream of a buffer, or $|P_u(i) - P_d(i)|$

$|P_u(i) - P_d(i)|$, $i = 1, ..., k-1$, indicates the difference between the production rates of the upstream portion (treated as a complete system) of Buffer B_i and the downstream portion (also treated as a complete system) of Buffer B_i - if Buffer B_i were removed and the upstream line and the downstream line were allowed to run separately from each other.

If $|P_u(i) - P_d(i)|$ is large, then the production rate of the upstream portion of the line is much different from the production rate of the downstream portion of the line. Therefore, Buffer B_i will be almost always empty of full, the variability of Buffer B_i is small, and the capacity of Buffer B_i should be reduced.

Finally, $|P_u(i) - P_d(i)|$ or $\frac{1}{|P_u(i) - P_d(i)|}$, $i = 1, ..., k-1$, is also an indication of relative variability and can be used to replace the gradient.

3.2 Motivations of the New Directions

In this research, we strive to develop more efficient ways to allocate buffer spaces in a transfer line. The gradient technique has proved to be very efficient. However, the evaluation of the gradient is time-consuming. Moreover, the evaluation of the gradient gets slower as the production line gets longer. This long evaluation time limits the size of a system that can be optimized.

In this section, the number of computation steps for both the gradient and the alternative algorithms are compared.

For both the gradient and the alternative algorithms, the major computation per optimization step for each new guess consists of:

- One direction computation

  The direction computation is the computation calculated at the current point N to determine the direction to the next point N_{new} on the constraint space. For the gradient method, the direction is the gradient. For the other methods, the directions are the variance, standard deviation, and vectors with $|\overline{n_i} - \frac{N_i}{2}|$ or $\frac{1}{|\overline{n_i} - \frac{N_i}{2}|}$, $|p_b(i) - p_s(i)|$ or $\frac{1}{|p_b(i) - p_s(i)|}$, and $|P_u(i) - P_d(i)|$ or $\frac{1}{|P_u(i) - P_d(i)|}$ as their components.

- A few one-dimensional search computations

  The one-dimensional search computations are the computations to determine N_{new}, the point with the highest production rate of all points along the line formed by $\prod$ (the projected gradient onto the constraint space).

Figure 3-1 shows N, the gradient vector g, the projected gradient $\prod$, and the constraint space for $k = 4$.

Next, we show that alternative directions calculate approximate gradient direction with much less computation per optimization step.

Table 3.1 shows the number of computations needed to calculate the direction for each algorithm. There are two types of direction computations. The first

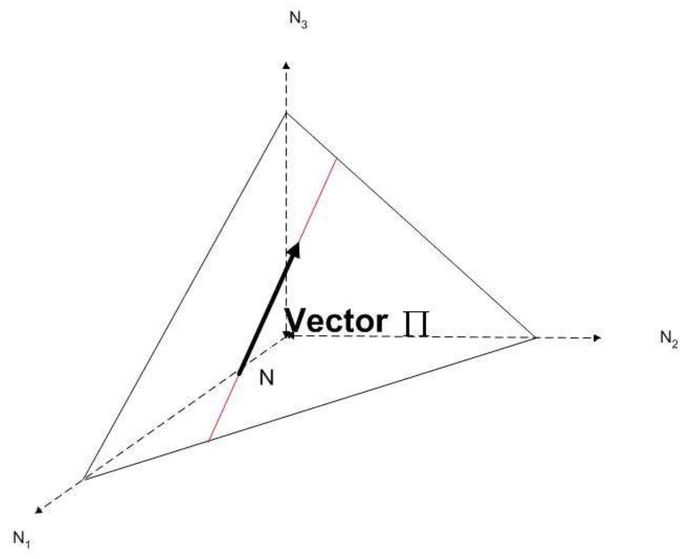


Figure 3-1: The Gradient and the Projected Gradient Vectors

one is the number of k-machine line computations, which is the number of times the decomposition algorithm is performed on the k-machine line. The other one is the number of shorter line computations, which is the number of times the decomposition algorithm is performed on part of the $k-$machine line.

Table 3.1: Direction Computation Comparisons for the Different Methods

Direction	Number of k-Machine Line Computations	Number of Shorter Line Computations
Gradient	k	0
Variance	1	0
Standard Deviation	1	0
$\lvert\overline{n_i} - \frac{N_i}{2}\rvert$	1	0
$\frac{1}{\lvert\overline{n_i} - \frac{N_i}{2}\rvert}$	1	0
$\lvert p_b(i) - p_s(i)\rvert$	1	0
$\frac{1}{\lvert p_b(i) - p_s(i)\rvert}$	1	0
$\lvert P_u(i) - P_d(i)\rvert$	0	$2(k-1)$
$\frac{1}{\lvert P_u(i) - P_d(i)\rvert}$	0	$2(k-1)$

Figure 3-2 shows how the k-machine line computations and the shorter-machine line computations fit into the major computation calculated at each optimization step.

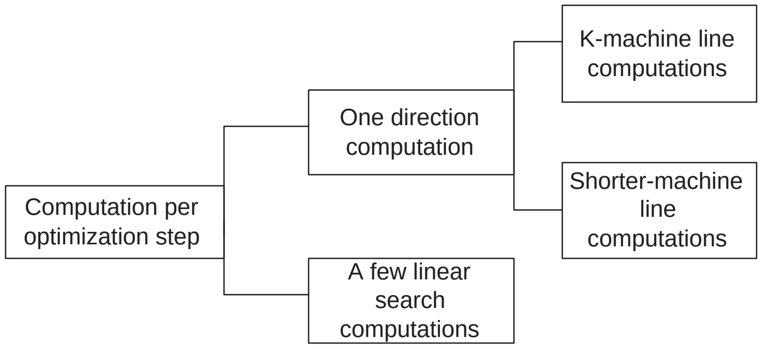

Computation per optimization step
One direction computation
A few linear search computations
K-machine line computations
Shorter-machine line computations

Figure 3-2: Types of Computations Calculated at Each Optimization Step

Next we explain how we obtain the number of direction computations shown in Table 3.1.

(a) Gradient direction

For the Gradient Algorithm, the number of computations to calculate the direction is k k-machine line computations.

The gradient is a $k-1$-vector. Component $i, i = 1, ..., k-1$, indicates the change in production rate of the line if the capacity of Buffer B_i is increased by a small amount. In the Gradient Algorithm, we calculate $k-1$ production rates associated with the capacity increase of the $k-1$ buffers. In addition, we also need to calculate the production rate associated with the original (unmodified) capacity of buffers. Therefore, the total computations needed to calculate the gradient is k computations.

As the production line gets longer, k gets bigger and the number of computations to calculate the gradient vector gets larger as well.

(b) Variance, standard deviation of buffer B_i's inventory levels, $|\overline{n_i} - \frac{N_i}{2}|$, $\frac{1}{|\overline{n_i} - \frac{N_i}{2}|}$, $|p_b(i) - p_s(i)|$, $\frac{1}{|p_b(i) - p_s(i)}|$ Directions

In these other algorithms, only one k-machine line computation is required. With only one decomposition algorithm called, the quantities $\overline{n_i}, N_i, p_b(i), p_s(i)$, $i = 1, ..., k-1$, are calculated simultaneously.

(c) $|P_u(i) - P_d(i)|$, $\frac{1}{|P_u(i) - P_d(i)|}$ directions

To calculate $|P_u(i) - P_d(i)|$, we hypothetically remove Buffer B_i. There will be two shorter lines resulting from the removal of Buffer B_i: the shorter line upstream of Buffer B_i and the shorter line downstream of B_i. To calculate $|P_u(i) - P_d(i)|$ for Buffer B_i, we run the Decomposition Algorithm twice: first on the upstream portion of the line and the other on the downstream portion. Since there are $k-1$ buffers and there are two decomposition algorithms run for each buffer, the number of computations to calculate $|P_u(i) - P_d(i)|$ or $\frac{1}{|P_u(i) - P_d(i)|}$ $\forall i, i = 1, ..., k-1$, is $2(k-1)$ shorter line computations.

3.3 Derivation of the New Algorithms

The variance and standard deviation of buffer levels, as well as vectors with component i $|\overline{n_i} - \frac{N_i}{2}|$, $|p_b(i) - p_s(i)|$, and $|P_u(i) - P_d(i)|$ are indications of buffer level variability and can be used to decide buffer sizes. In Schor's gradient algorithm, the gradient is used to decide how to change buffer sizes. In the new algorithms, the other directions are used to replace the gradient.

Figure 3-3 shows the block diagram of the new algorithms.

Let vector g' represent any of the alternative directions. In the new algorithms, g' is used to replace the original gradient vector g. The vector g' is also projected onto the constraint space. The projected vector is called $\prod'$. The rest of the algorithm is the same as the original Schor's Gradient Algorithm.

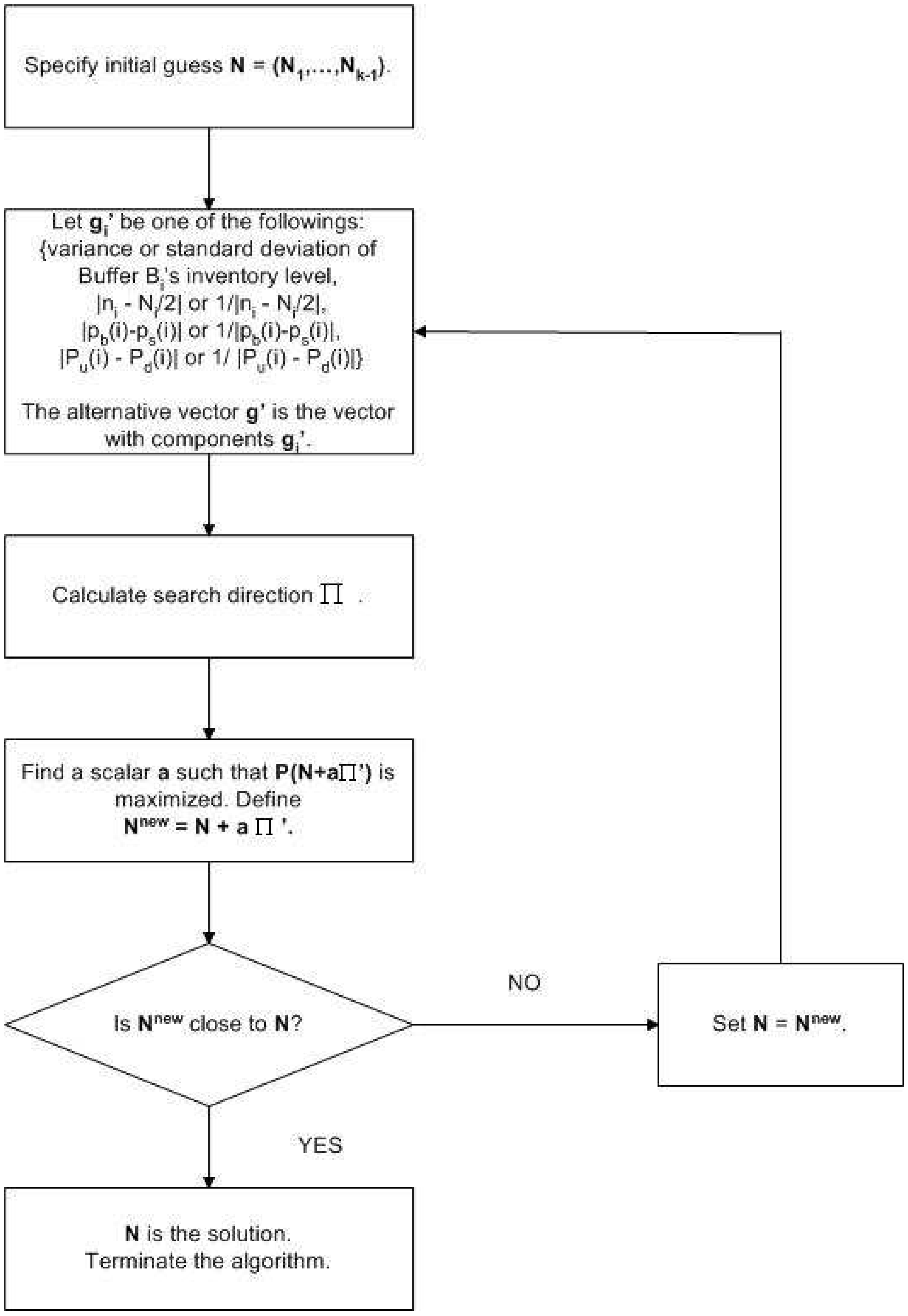

Specify initial guess $\mathbf{N} = (\mathbf{N}_1,\ldots,\mathbf{N}_{k-1})$.
Let $\mathbf{g}_i$' be one of the followings: {variance or standard deviation of Buffer B_i's inventory level, $|n_i - N_i/2|$ or $1/|n_i - N_i/2|$, $|p_b(i)-p_s(i)|$ or $1/|p_b(i)-p_s(i)|$, $|P_u(i) - P_d(i)|$ or $1/ |P_u(i) - P_d(i)|$}
The alternative vector $\mathbf{g}'$ is the vector with components $\mathbf{g}_i$'.
Calculate search direction Π .
Find a scalar $\mathbf{a}$ such that $\mathbf{P}(\mathbf{N}+\mathbf{a}\Pi')$ is maximized. Define $\mathbf{N}^{new} = \mathbf{N} + \mathbf{a}\,\Pi\,'$.
Is $\mathbf{N}^{new}$ close to $\mathbf{N}$?
NO
Set $\mathbf{N} = \mathbf{N}^{new}$.
YES
$\mathbf{N}$ is the solution. Terminate the algorithm.

Figure 3-3: Block Diagram of the New Algorithms

Chapter 4

Performance of the New Algorithms

The performance of the new algorithms will be evaluated in terms of:

1. Accuracy

   Accuracy measures how close the final production rates of the new algorithms to the final production rate of the gradient.

2. Reliability

   Reliability measures how often the final production rates of the new algorithms converge to the final production rate of the gradient algorithm.

3. Speed

   Speed measures how fast the new algorithms converge.

4.1 Accuracy

We access the accuracy of the new algorithms by studying the effects of the followings:

1. Identical versus non-identical machines

2. The location, existence, and severity of bottlenecks

Let *final P* be the production rate when each algorithm terminates. *Two-machine line counter* is defined as the number of times we evaluate the hypothetical two-machine line in the decomposition algorithm. The two-machine line counter is counted at each optimization step. *Final two-machine line counter* is the two-machine line counter when each algorithm terminates. The final two-machine line counter is used to measure the time it takes for each algorithm to terminate.

We compare the performance of all algorithms by comparing their final production rates and their final two-machine-line counters.

4.1.1 Identical versus Non-Identical Lines

Identical Machines

To study the impact of the identical-machine line on the performance of the new algorithms, we randomly generated 15 lines using the procedure described in Appendix *A*. The length of the line generated ranges from 8 machines to 36 machines. Figure 4-1 shows the average, the maximum, and the minimum of the proportion of the final production rates of the other algorithms to the final production rate of the gradient algorithm. Figure 4-2 shows the average, the maximum, and the minimum of the proportion of the final two-machine-line counters of the other algorithms to the final two-machine-line counter of the gradient algorithm.

Figure 4-1 and Figure 4-2 show that the variance and the standard deviation work very well as substitutes for the gradient. This is because the final production rates of the variance and standard deviation algorithms are as high as the final P of the gradient algorithm and the convergence times of the two algorithms are, on average, half that of the gradient algorithm. We do not recommend using $\frac{1}{|\overline{n_i}-\frac{N_i}{2}|}$ and $\frac{1}{|p_b(i)-p_s(i)|}$ to replace g_i (component i of the gradient) because their convergence times might be as long as four times the convergence times of the gradient algorithm. The other directions ($\frac{1}{|P_u(i)-P_d(i)|}$, $|\overline{n_i}-\frac{N_i}{2}|$, $|p_b(i)-p_s(i)|$, and $|P_u(i)-P_d(i)|$) do not outperform the gradient because their final production rates are slightly lower than the final production rate of the gradient. Although their average convergence times

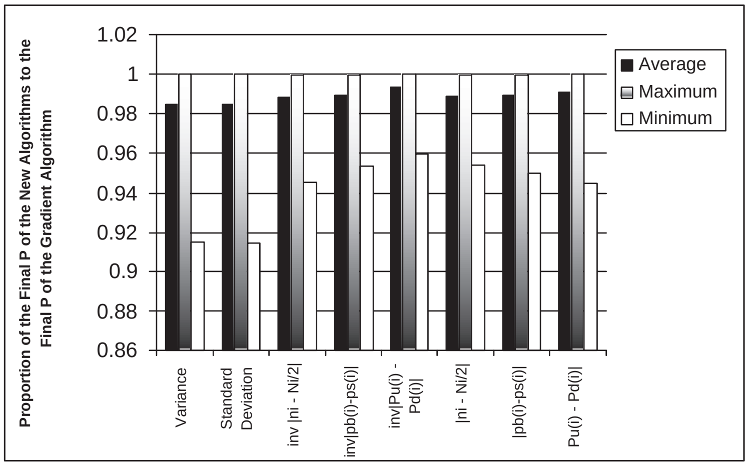


Figure 4-1: Final P of the Other Directions as Proportions of Final P of the Gradient in the Identical Machine Case

are lower than that of the gradient, occasionally their convergence times might be longer than the convergence time of the gradient.

Figure 4-3 shows that the P and the two-machine-line counter at each optimization step for one of the line generated. The parameters of the line are described in Table 4.1. The variance and standard deviation algorithms reach the final P of the gradient algorithm in only a few steps. On the other hand, the gradient algorithm takes longer to achieve and exceed the final P of the variance and the standard deviation algorithms.

Table 4.1: The Line Parameters of which Results are Shown in Figure 4-3

Number of Machines	30 identical machines
Total Buffer Space to be Allocated	745
Repair Rate r	7.096
Failure Rate p	0.874
Processing Rate μ	141.531

Non-Identical Machines

We also study the impact of non-identical machine lines on the performance of the new algorithms. Nineteen lines that consist of non-identical machines are generated

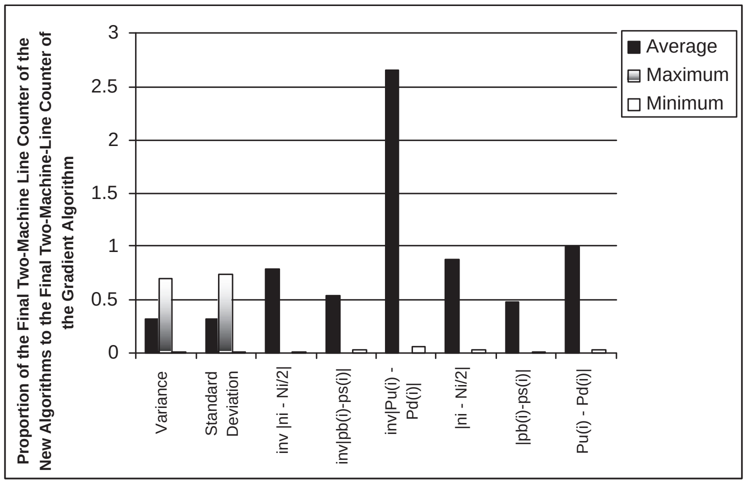


Figure 4-2: Final Two-Machine Line Counter of the Other Directions as Proportions of Final Two-Machine Line Counter of the Gradient in the Identical Machine Case

according to the procedure described in Appendix A. The line length generated ranges from 7 to 40 machine line and might contain up to three bottleneck machines. The selection and severity of the bottlenecks are randomly generated according to the procedure in Appendix A.

Figure 4-4 shows the average, the maximum, and the minimum of the proportion of the final P of the other algorithms to the final P of the gradient algorithm. Figure 4-5 shows the average, the maximum, and the minimum of the proportion of the final two-machine-line counters of the other algorithms to the final two-machine-line counter of the gradient algorithm.

Figure 4-4 and Figure 4-5 show that the variance and the standard deviation algorithms also do well in the non-identical machine case, although not as well as in the identical machine case. This is shown by the fact that the average final P of the two algorithms are 0.94 of the average final P of the gradient algorithm and the average convergence times are less than half the convergence time of the gradient algorithm. We do not recommend using $\frac{1}{|P_u(i)-P_d(i)|}$, $|\overline{n_i} - \frac{N_i}{2}|$, and $|P_u(i) - P_d(i)|$ to replace g_i because their final two-machine line counters are almost as high or higher than the final two-machine line counters of the gradient. In addition, their final two-machine line counters vary greatly. For example, the final two-machine line counter of the $\frac{1}{|P_u(i)-P_d(i)|}$ direction can be as small as 0.05 the final two-machine line counter

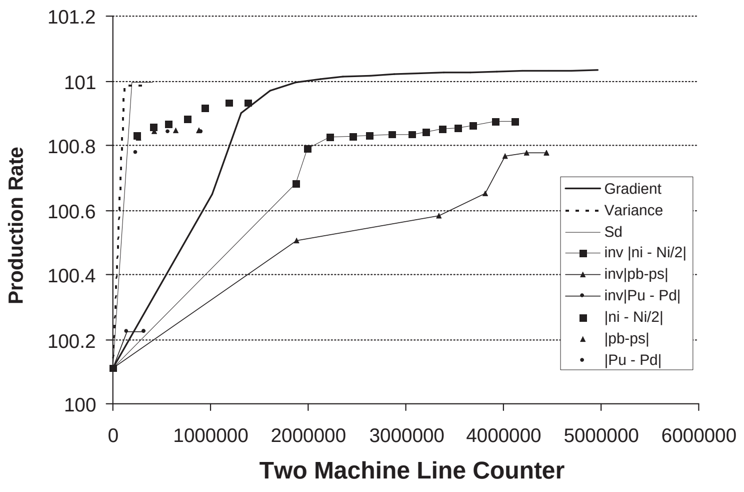


Figure 4-3: Production Rate versus Two-Machine Line Counter for an Identical Machine Case

of the gradient, but can be as big as 40 times that of the gradient.

4.1.2 The Location, Existence, and Severity of Bottlenecks

Existence of Bottleneck in Short and Long Lines

In this section, we investigate the performance of the new algorithms in the following cases:

- Short lines with bottlenecks
- Short lines with no bottlenecks
- Long lines with bottlenecks
- Long lines with no bottlenecks

Short lines are defined as lines with less than 30 machines and long lines as lines with more than 30 machines. Lines with bottlenecks can contain up to three bottlenecks, with $\rho_{bottleneck}$ between 0.5 and 0.8 of the minimum ρ of the other non-bottlenecks.

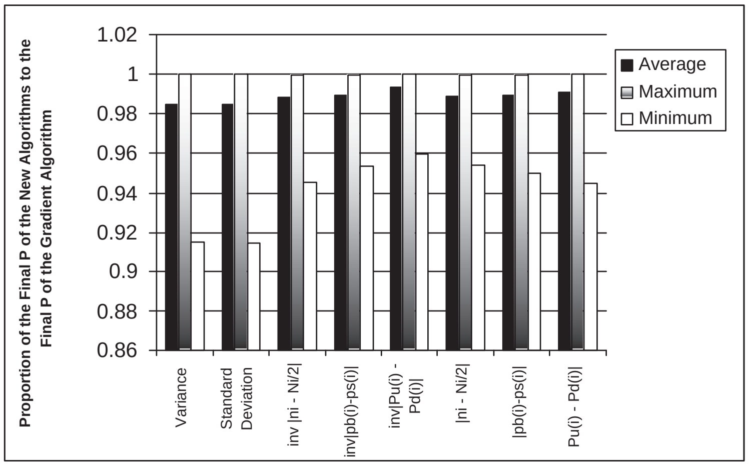


Figure 4-4: Final P of the Other Directions as Proportions of Final P of the Gradient in the Non-Identical Machine Case

To minimize the effects of outside factors (other than short lines versus long lines and lines with bottlenecks versus lines without bottlenecks) that influence the performance of the new algorithms, the short lines with bottleneck are created from the short lines with no bottleneck by converting a randomly chosen non-bottleneck into a bottleneck. This is done by imposing a low isolated machine production rate ρ on the bottleneck. Similarly, long lines with bottleneck are created from long lines without bottleneck by converting a previously non-bottleneck by imposing a low ρ on it. We created 10 short lines with bottlenecks and 10 short lines without bottlenecks. The results for short lines with bottleneck are shown in Figure 4-6 and Figure 4-7. The results for short lines without bottleneck are shown in Figure 4-8 and Figure 4-9.

Figure 4-6, Figure 4-7, Figure 4-8, and Figure 4-9 show that the new algorithms perform better in lines with no bottlenecks than in lines with bottlenecks. This is because in lines with no bottlenecks the final P of the new algorithms are in the neighborhood of the final P of the gradient algorithm and the final two-machine line counter of the new algorithms are smaller than that of the gradient algorithm. The results that the new algorithms perform better in non-bottleneck cases are even more apparent in longer lines as shown in Figure 4-10, Figure 4-11, Figure 4-12, and Figure 4-13. Figure 4-12 shows that the final P of the new algorithms are very close or higher than the final P of the gradient algorithm. Figure 4-13 demonstrates that, on

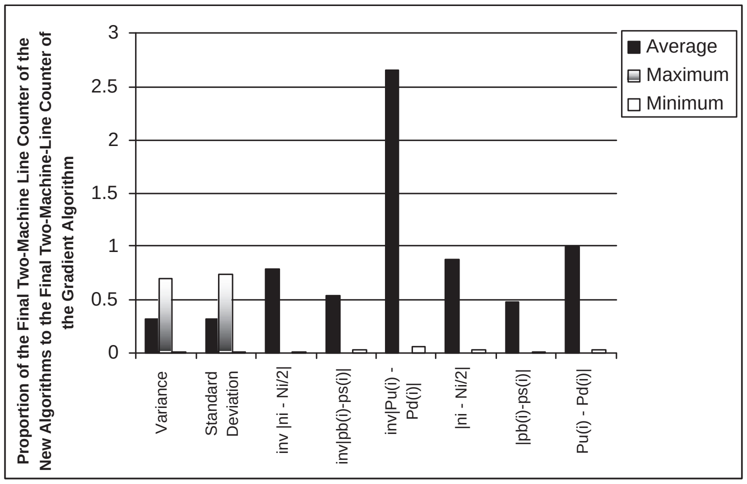


Figure 4-5: Final Two-Machine Line Counter of the Other Directions as Proportions of Final Two-Machine Line Counter of the Gradient in the Non-Identical Machine Case

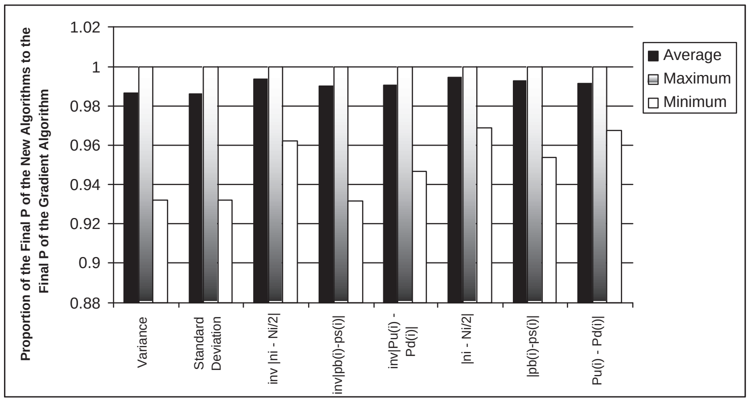


Figure 4-6: Final P of the Other Directions as Proportions of Final P of the Gradient Algorithm in Short Lines with Bottlenecks

average, the new algorithms converge much faster than the gradient algorithm.

Location and Severity of Bottlenecks

We also study the effects of locations of a bottleneck and severity of a bottleneck on the performance of the new algorithms. We first create lines with no bottlenecks. Later we create lines with a bottleneck on the upstream, lines with a bottleneck on the middle stream, and another lines with a bottleneck on the downstream. To minimize the effects of outside factors, the parameters of the bottleneck located upstream,

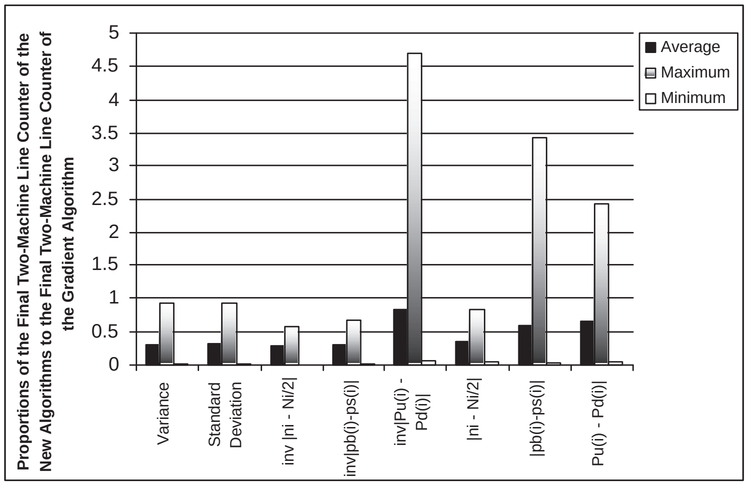


Figure 4-7: Final Two-Machine Line Counters of the Other Directions as Proportions of Final Two-Machine Line Counters of the Gradient Algorithm in Short Lines with Bottlenecks

middle stream, and downstream of the lines are the same. At first, these bottlenecks are made less severe, with ρ of the bottleneck is between 0.8 and 0.95 of the minimum ρ of the other machines. Later, the bottlenecks are made more severe, with ρ of each bottleneck is ≤ 0.8 of the minimum ρ of the other machines. As before, we first study the effects on identical machine lines and later on the non-identical machine lines. In the identical machine case, all machines, except the bottleneck, are identical. In the non-identical machine case, all machine are different with the bottleneck having the lowest ρ of all.

In both the identical and non-identical machine cases, the location of bottlenecks does not strongly affect the final P of the new algorithms, as shown in Figure 4-14, Figure 4-15, and Figure 4-16 for the identical machine case. These figures show that the proportion of the final P of all the algorithms to the final P of the gradient algorithm are similar, regardless of the location of the bottleneck. The speeds of the new algorithms, except the variance and the standard deviation algorithms, are slightly affected by the location of the bottleneck, as shown in Figure 4-17, Figure 4-18, and Figure 4-19. For example, Figure 4-17 shows that it takes at most 9 times the speed of the gradient algorithm for the $\frac{1}{|p_b(i)-p_s(i)|}$ algorithm to converge when the bottleneck is located upstream in the identical machine line. However, Figure

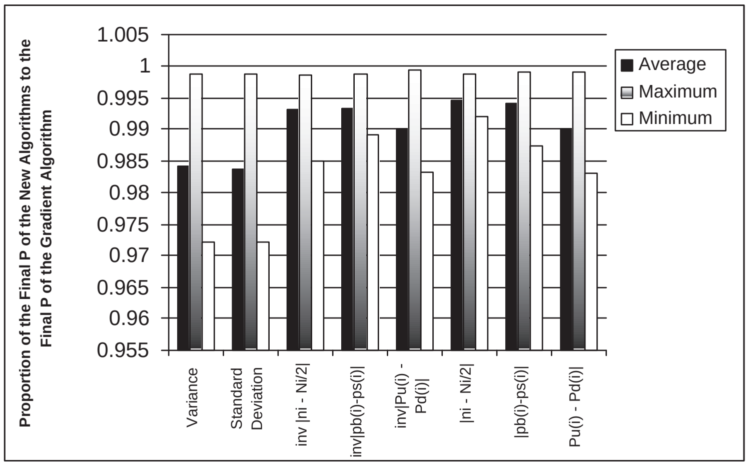


Figure 4-8: Final P of the Other Directions as Proportions of Final P of the Gradient Algorithm in Short Lines with No Bottlenecks

4-18 demonstrates that it takes at most only 2.5 times the speed of the gradient algorithm for the $\frac{1}{|p_b(i)-p_s(i)|}$ algorithm to converge when the bottleneck is located in the middle of the line. Unfortunately, we could not explain how the speed of the new algorithms vary with the locations of the bottleneck. As mentioned earlier, the speed of the variance and standard deviation algorithms is not affected by the location of the bottleneck.

Furthermore, Figure **??** and Figure **??** show that the new algorithms work better in the non or mild bottleneck cases than in severe bottleneck cases in the identical machine cases.

4.2 Reliability

To test the reliability of the new algorithms, we start each algorithm with three different initial buffer allocations and investigate if the new algorithms will converge to the final, or the neighborhood of the final, P of the gradient algorithm.

The three initial buffer allocation methods are as follows:

1. Equal Buffer Allocation

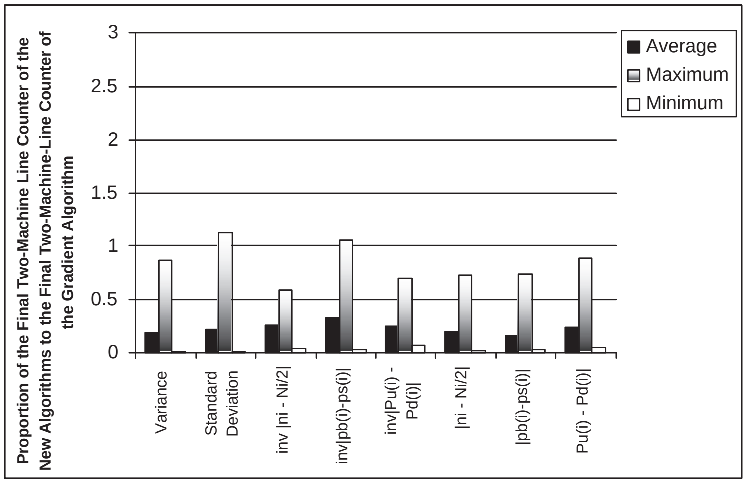


Figure 4-9: Final Two-Machine Line Counters of the Other Directions as Proportions of Final Two-Machine Line Counters of the Gradient Algorithm in Short Lines with No Bottlenecks

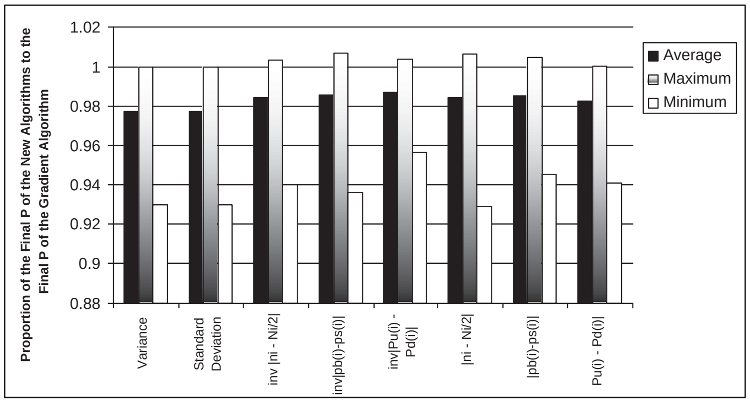


Figure 4-10: Final P of the Other Directions as Proportions of Final P of the Gradient Algorithm in Long Lines with Bottlenecks

Equal buffer allocation means that N_i is initially allocated as

$$N_i = \frac{N_{TOTAL}}{k-1}; i = 1, ..., k-1$$

2. Buffer Allocation Proportional to $\frac{1}{\rho_i + \rho_{i+1}}$

The intuition behind this type of buffer allocation is as follows: if two adjacent machines M_i and M_{i+1} have large isolated production rates ρ_i and ρ_{i+1}, parts in Buffer B_i tend to get processed immediately by Machine M_{i+1}. Therefore,

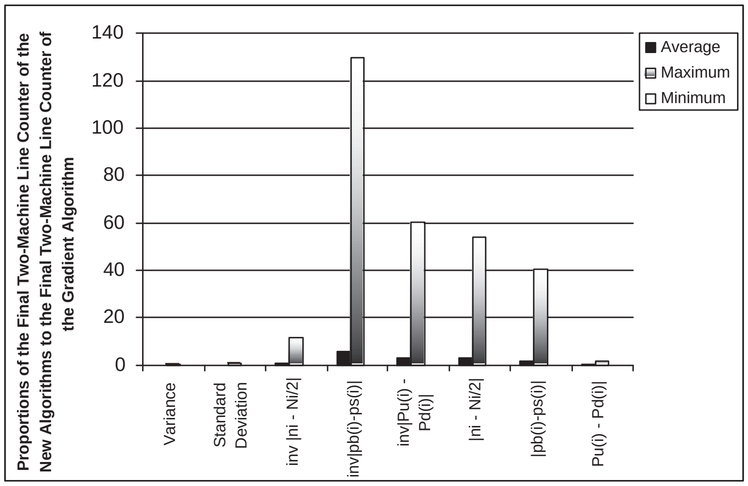


Figure 4-11: Final Two-Machine Line Counters of the Other Directions as Proportions of Final Two-Machine Line Counters of the Gradient Algorithm in Long Lines with Bottlenecks

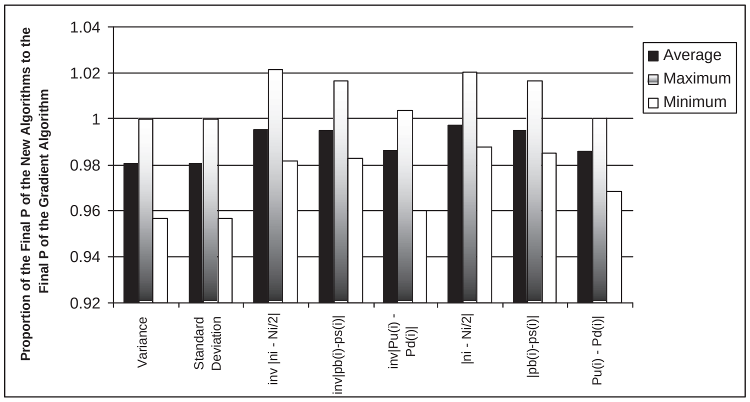


Figure 4-12: Final P of the Other Directions as Proportions of Final P of the Gradient Algorithm in Long Lines with No Bottlenecks

Buffer B_i should get little space. On the other hand, if Machines M_i and M_{i+1} are slow (or their ρ_s are small), Machine M_{i+i} most likely has not finished processing parts when Machine M_i finishes. Therefore, Buffer B_i should get more space to store parts that will get processed eventually by Machine M_{i+1}.

This method of initial buffer allocation requires that N_i is allocated as

$$N_i = \frac{1}{\rho_i + \rho_{i+1}} C; i = 1, ..., k-1$$

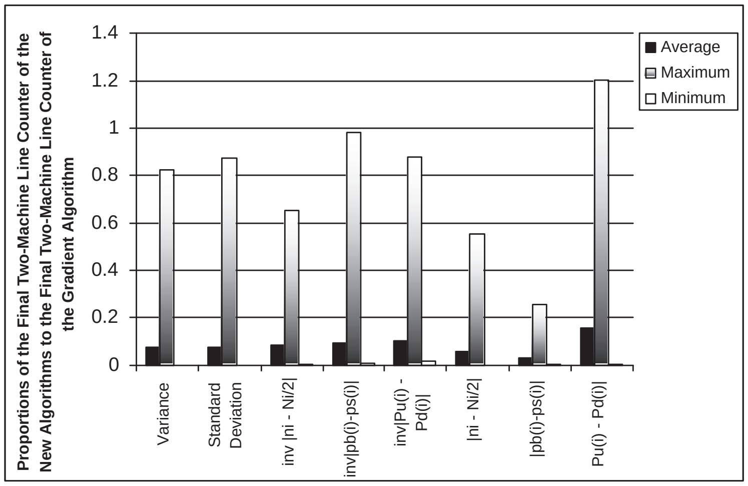


Figure 4-13: Final Two-Machine Line Counters of the Other Directions as Proportions of Final Two-Machine Line Counters of the Gradient Algorithm in Long Lines with No Bottlenecks

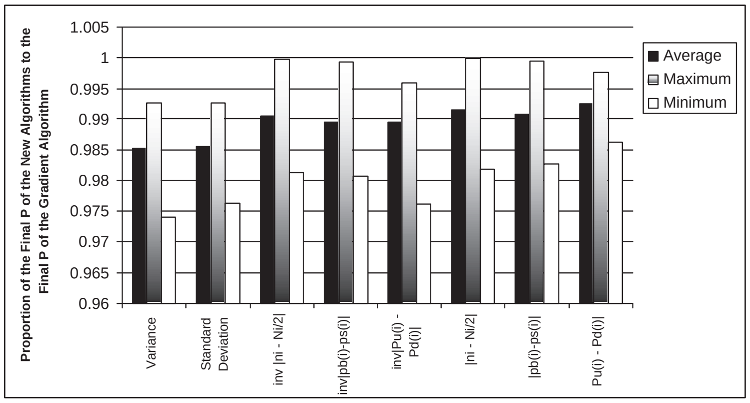


Figure 4-14: Final P of the Other Directions as Proportions of Final P of the Gradient Algorithm in Identical Machine Lines with One Bottleneck Located Upstream

where C is a normalizing factor, which is mathematically defined below.

$$C = \frac{N_{TOTAL}}{\sum_{i=1}^{k-1} \frac{1}{\rho_i + \rho_{i+1}}}$$

3. Random Buffer Allocation

Here we randomly allocate the initial buffer size subject to $\sum_{i=1}^{k-1} N_i = N_{TOTAL}$.

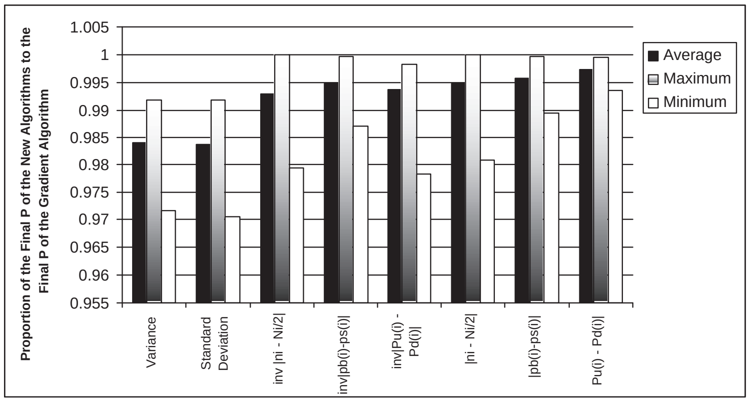


Figure 4-15: Final P of the Other Directions as Proportions of Final P of the Gradient Algorithm in Identical Machine Lines with One Bottleneck Located in the Middle of the Line

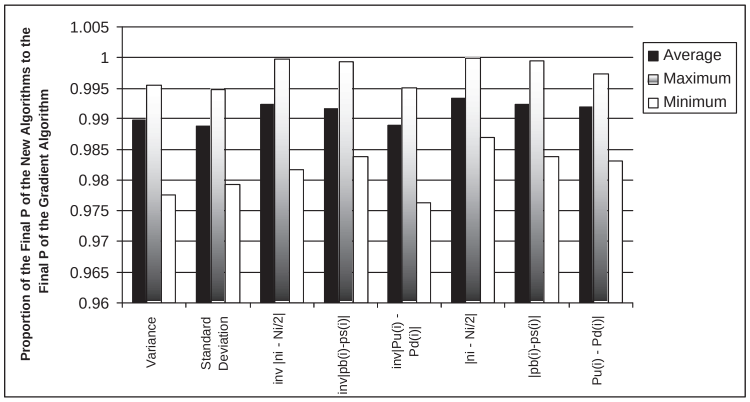


Figure 4-16: Final P of the Other Directions as Proportions of Final P of the Gradient Algorithm in Identical Machine Lines with One Bottleneck Located Downstream

Let C be a normalizing factor such that:

$$C = \frac{N_{TOTAL}}{\sum_{i=1}^{k-1} N_{TOTAL}U(0,1)}$$

and U(0,1) = a uniformly distributed random number from 0 to 1.

The initial N_i is allocated as

$$N_i = N_{TOTAL}U(0,1)C; i = 1, ..., k-1$$

We now examine the performance of the new algorithms using the three different

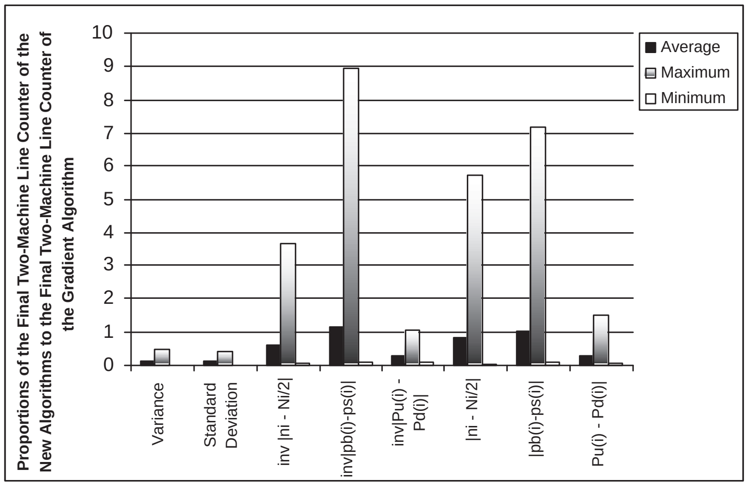


Figure 4-17: Final Two-Machine Line Counter of the Other Directions as Proportions of Final Two-Machine Line Counter of the Gradient Algorithm in Identical Machine Lines with One Bottleneck Located Upstream

initial buffer allocations. We first describe their performance in the identical machine case and later in the non-identical machine case.

4.2.1 Identical Machine Lines

We create 10 different lines and use the three different methods to initialize the buffer size. Figure 4-21, Figure 4-22, and Figure 4-23 show the performance of the new algorithms, in terms of their final production rates as proportions of the final production rate of the gradient, using the three initial buffer allocation methods. All three figures show the average, the maximum and the minimum performance measure of the 10 lines. The average final production rates of the new algorithms are at least 95% the final production rate of the gradient, showing that the new algorithms converge close to the optimal point of the gradient algorithm. Therefore, the new algorithms are reliable.

4.2.2 Non-Identical Machine Lines

We also create 10 lines and initialize the buffer size using the three different buffer allocation methods. Figure 4-24, Figure 4-25, and Figure 4-26 show the results for

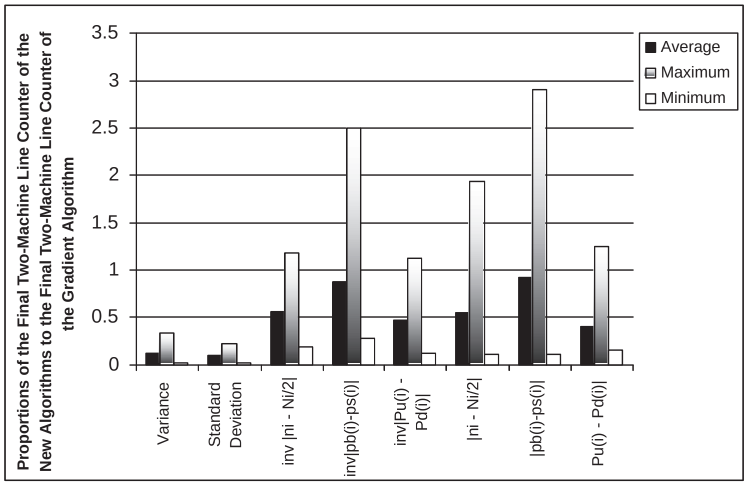


Figure 4-18: Final Two-Machine Line Counter of the Other Directions as Proportions of Final Two-Machine Line Counter of the Gradient Algorithm in Identical Machine Lines with One Bottleneck Located in the Middle of the Line

the non-identical machine case. Using the three initial buffer allocation methods, the average final production rate of the new algorithms are also at least 95% the final production rate of the gradient. Therefore, the new algorithms are also reliable in the non-identical machine case.

4.3 Speed

To evaluate the speed of the new algorithms, we compare the final two-machine-line counters of all algorithms as the line's length increases. The two-machine-line counters, which is how often the two-machine evaluation in the decomposition algorithm is called, is used to measure the completion time of all algorithms. We first describe the results for the identical machine case, and later describe the results for the non-identical machine case.

4.3.1 Identical Machine Lines

For the identical machine case, 5 different lines are generated. All figures in this section demonstrate the average results of the 5 lines.

Figure 4-27 shows the final two-machine line counters of the new algorithms as

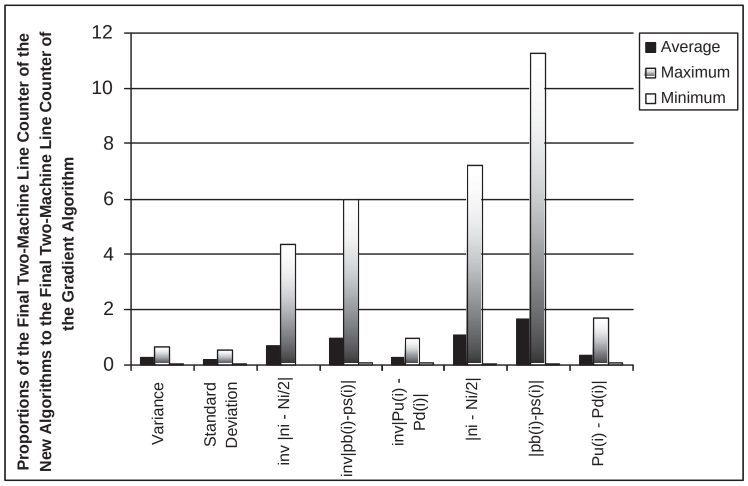


Figure 4-19: Final Two-Machine Line Counter of the Other Directions as Proportions of Final Two-Machine Line Counter of the Gradient Algorithm in Identical Machine Lines with One Bottleneck Located Downstream

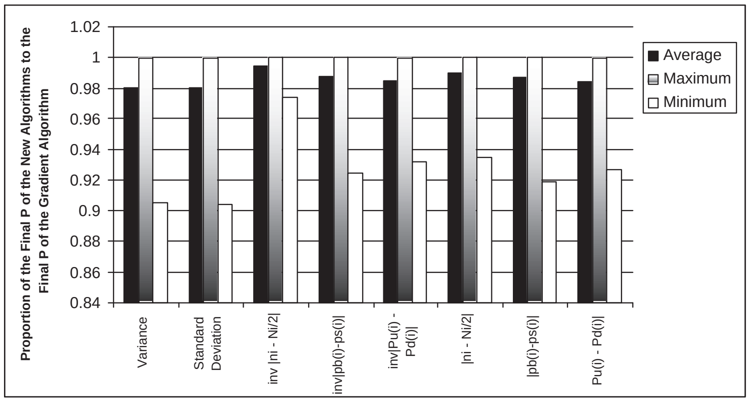


Figure 4-20: Final P of the Other Directions as Proportions of Final P of the Gradient Algorithm in Identical Machine Lines with One Severe Bottleneck Located Upstream

the line's length increases. It confirms that variance and standard deviation work very well to replace the gradient in the identical machine case. When lines are short, the final two-machine line counters of the variance and standard deviation methods are less than that of the gradient. When lines are long, their final two-machine line counters are similar to that of the gradient; in addition, their final production rates are slightly higher than the final production rate of the gradient method. The fact that the P of the variance and standard deviation methods are higher than the P of the gradient in long lines is shown in Figure 4-28. It demonstrates that the production

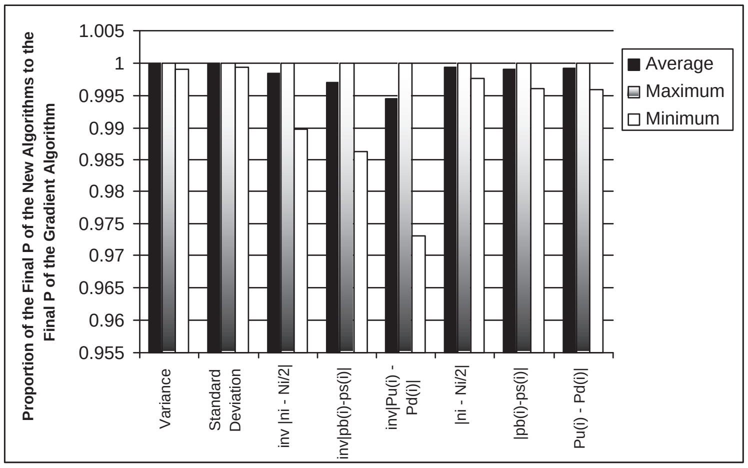


Figure 4-21: Final P of the Other Directions as Proportions of Final P of the Gradient in the Identical Machine Case, with Equal Initial Buffer Allocation

rate versus the two-machine line counter for an identical machine case, of which data is shown in Table 4.2.

Table 4.2: The Line Parameters of which Results are Shown in Figure 4-28

Number of Machines	42 identical machines
Total Buffer Space to be Allocated	4100
Repair Rate r	49.6796
Failure Rate p	0.535905
Processing Rate μ	840.604

4.3.2 Non-Identical Machine Lines

For the non-identical machine case, 5 different lines are generated, with a different machine added to the end of the line as k increases. All figures in this section demonstrate the average results of the 5 lines.

Figure 4-29 shows the speed of the new algorithms as line's length increases in the case of non-identical machine cases. It shows that, on average, the convergence time of the gradient algorithm is longer than the convergence times of the other algorithms.

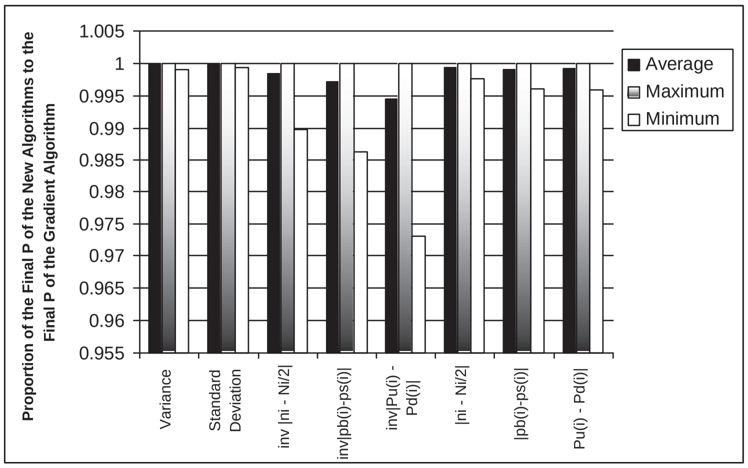


Figure 4-22: Final P of the Other Directions as Proportions of Final P of the Gradient in the Identical Machine Case, with Initial Buffer Allocation Proportional to $\frac{1}{\rho_i+\rho_{i+1}}$

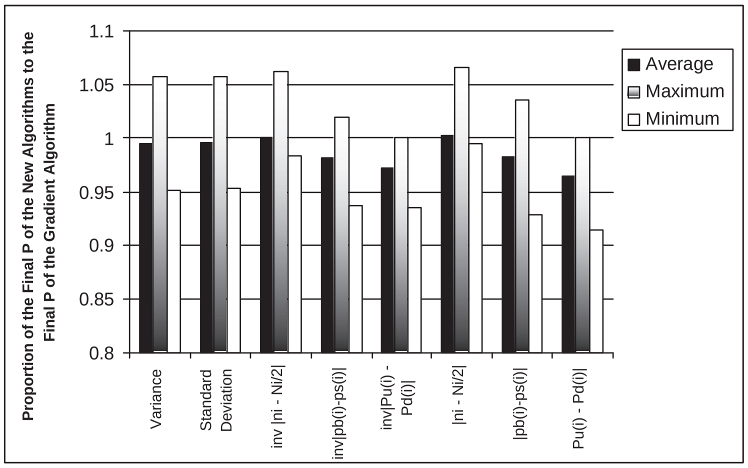


Figure 4-23: Final P of the Other Directions as Proportions of Final P of the Gradient in the Identical Machine Case, with Random Initial Buffer Allocation

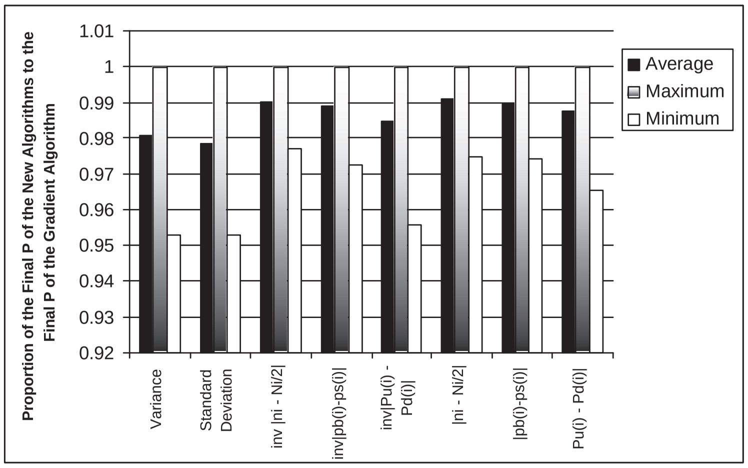


Figure 4-24: Final P of the Other Directions as Proportions of Final P of the Gradient in the Non-Identical Machine Case, with Equal Initial Buffer Allocation

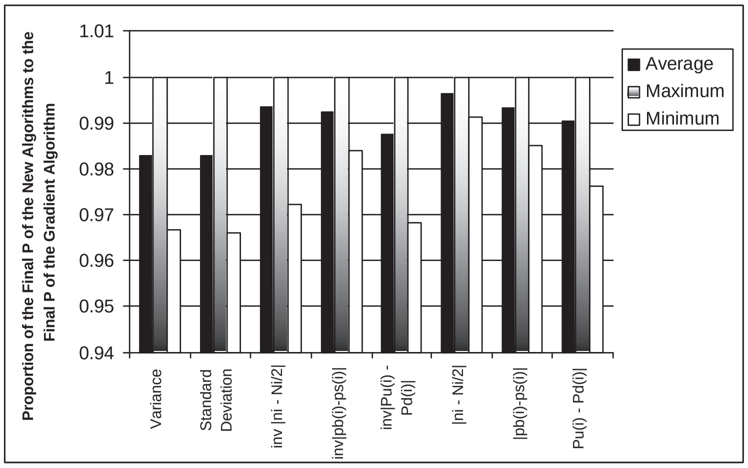


Figure 4-25: Final P of the Other Directions as Proportions of Final P of the Gradient in the Non-Identical Machine Case, with Initial Buffer Allocation Proportional to $\frac{1}{\rho_i+\rho_{i+1}}$

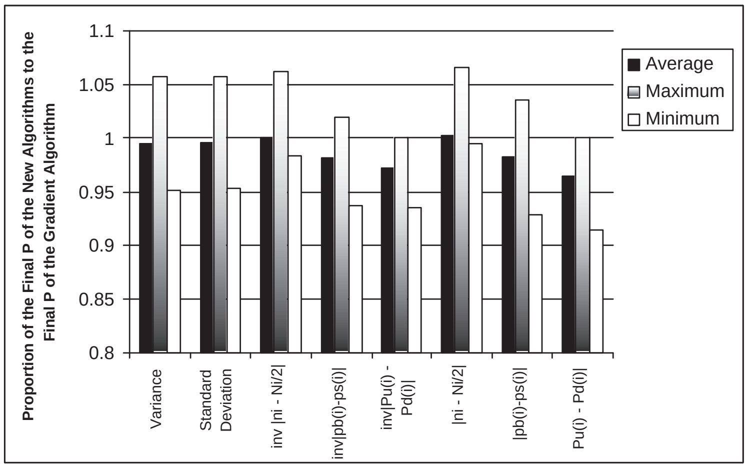


Figure 4-26: Final P of the Other Directions as Proportions of Final P of the Gradient in the Non-Identical Machine Case, with Random Initial Buffer Allocation

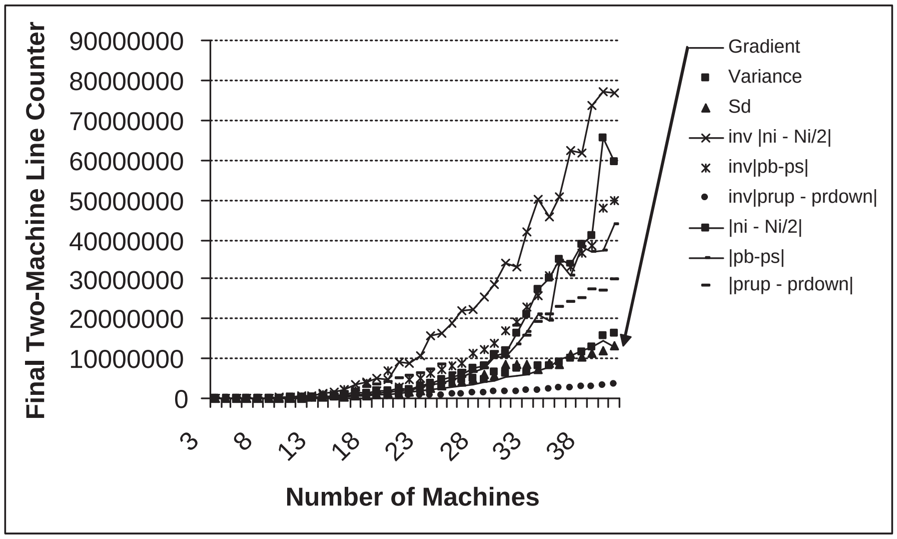


Figure 4-27: Final Two-Machine Line Counter as Line's Length Increases in the Identical Machine Case

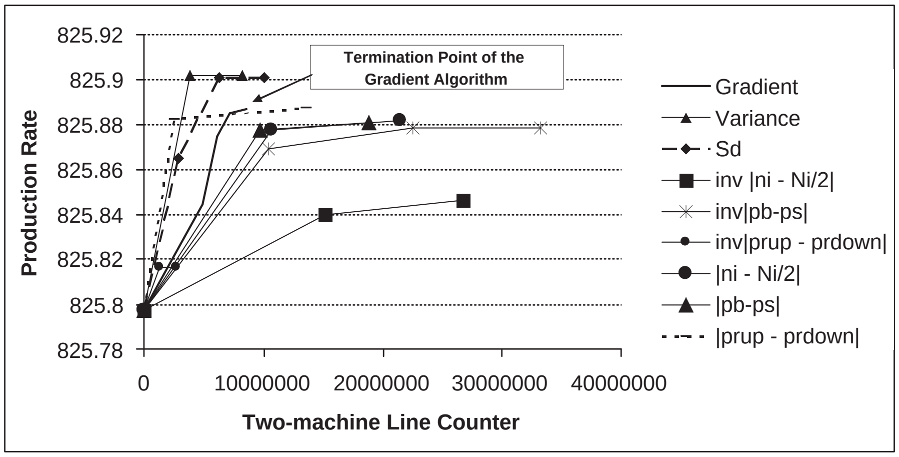


Figure 4-28: Production Rate versus Two-Machine Line Counter for an Identical Machine Case with $k = 42$

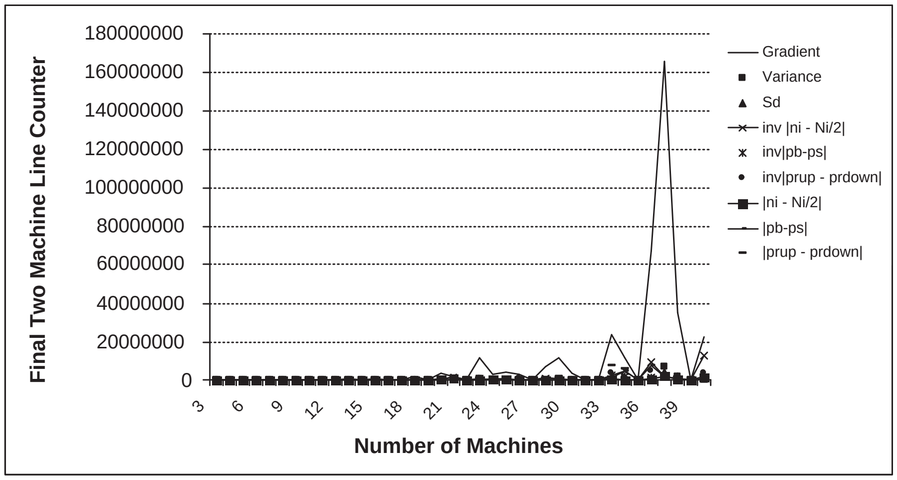


Figure 4-29: Final Two-Machine Line Counter as Line's Length Increases in the Non-Identical Machine Case

Chapter 5

Sensitivity Analysis

In this chapter, we test how sensitive the new algorithms are to small changes in machine parameters. As before, we first describe the results for identical machine lines and later for non-identical machine lines.

5.1 Identical Machine Lines

We first ran the new algorithms on an original line, of which data is described in Table 5.1.

Table 5.1: The Original Line's Parameters Used in Sensitivity Analysis of Identical Machine Lines

Number of Machines	From 3 to 42 identical machines
Total Buffer Space to be Allocated	$100(k-1)$
Repair Rate r	0.015
Failure Rate p	0.01
Processing Rate μ	1

Afterwards, we create eight lines, of which parameters are within 10% of the parameters of the original line. Initially, each of the eight lines has 3 machines. Later, we add one machine at a time to each line and perform the new algorithms on the longer line. We add a machine to each line until each line has 42 machines. We take the average of the two-machine-line counters at each k of the eight lines. Assuming that the two-machine-line counters at each k assumes a Normal distribution, we also compute the lower bound and upper bound of the two-machine-line counters at each k. The lower bound is computed using the formula $\mu_{two-machine-linecounter} - 2\sigma_{two-machine-linecounter}$. The upper bound is computed using $\mu_{two-machine-linecounter} + 2\sigma_{two-machine-linecounter}$. The average, lower bound, upper bound, and the original two-machine-line counter at each k for each algorithm are shown in Figure 5-1 until Figure 5-9. These figures show that all the algorithms, except the variance and the standard deviation algorithms, are sensitive to up-to 10% changes in machine parameters. For changes less than 10% of the original machine's parameters, all algorithms are expected to be less sensitive to the changes.

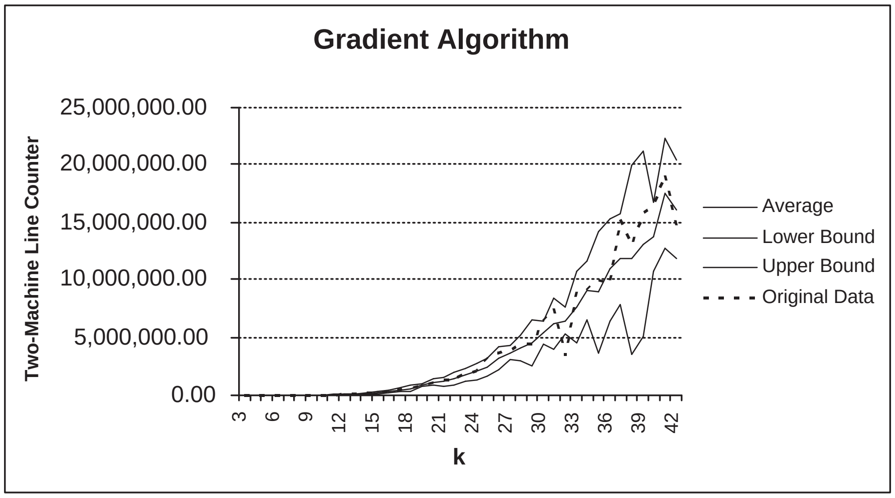


Figure 5-1: Sensitivity Results of the Gradient Algorithm on Identical Machine Lines

5.2 Non-Identical Machine Lines

We start with an original non-identical-machine line, of which parameters are described in Table 5.2. From this original line, we create eight lines, of which parameters are within 10% of the original line's parameters. Initially, each line has only three machines. We gradually add one machine at a time until the line's length becomes 23 machines. For each line, the total buffer space is $100(k-1)$. We run all algorithms on the modified and the original lines. Figure 5-10, Figure 5-11, and

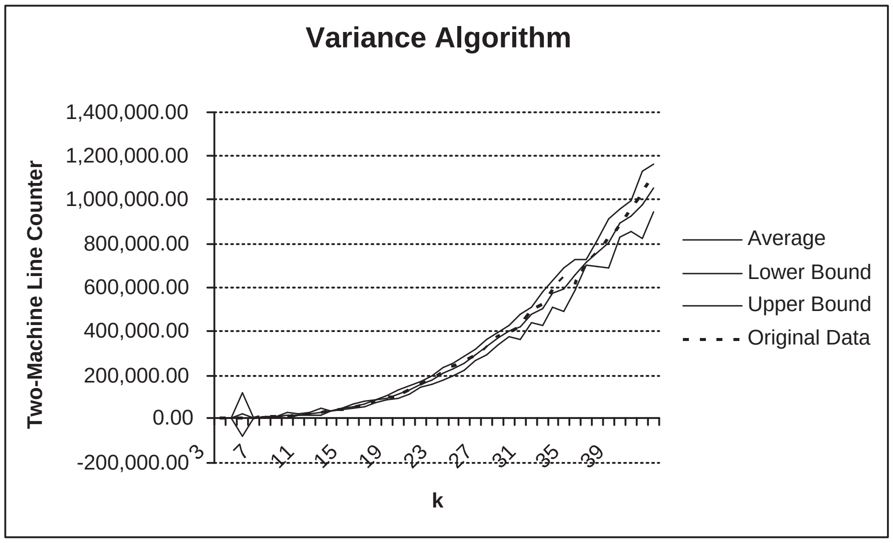


Figure 5-2: Sensitivity Results of the Variance Algorithm on Identical Machine Lines

Figure 5-12 show the average, the lower bound, the upper bound, and the original two-machine-line counters for the gradient, the variance, and the $\frac{1}{|\overline{n_i} - \frac{N_i}{2}|}$ algorithm. These figures show that as the lines get longer, all algorithms become more sensitive to the changes in parameters. The results for the standard deviation, $|\overline{n_i} - \frac{N_i}{2}|$, $|p_b(i) - p_s(i)|$, $|P_u(i) - P_d(i)|$, $\frac{1}{|p_b(i) - p_s(i)|}$, and $\frac{1}{|P_u(i) - P_d(i)|}$ are similar and therefore not shown.

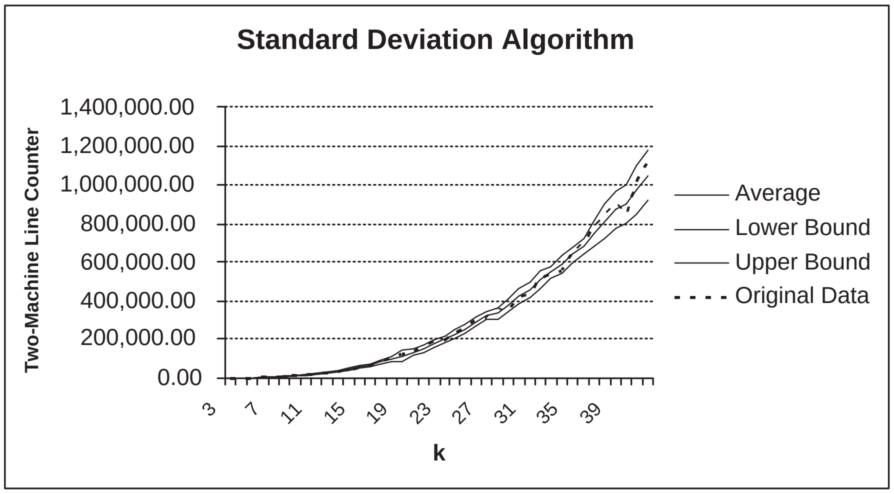


Figure 5-3: Sensitivity Results of the Standard Deviation Algorithm on Identical Machine Lines

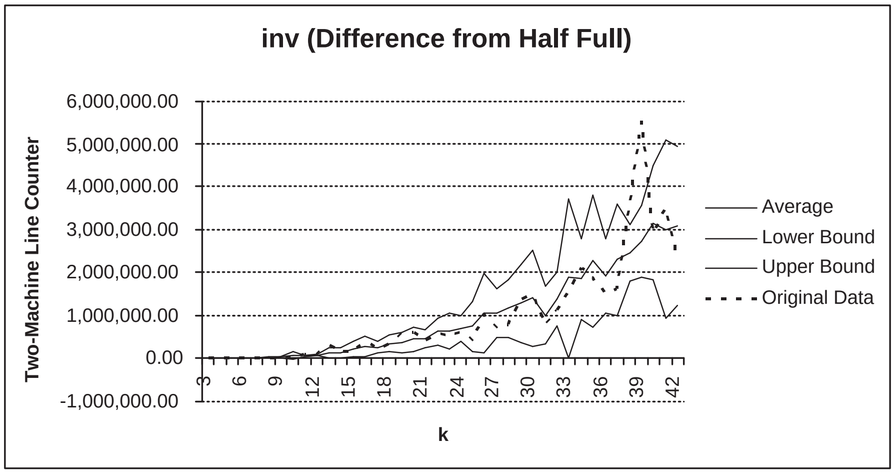


Figure 5-4: Sensitivity Results of the $\frac{1}{|\overline{n_i} - \frac{N_i}{2}|}$ Algorithm on Identical Machine Lines

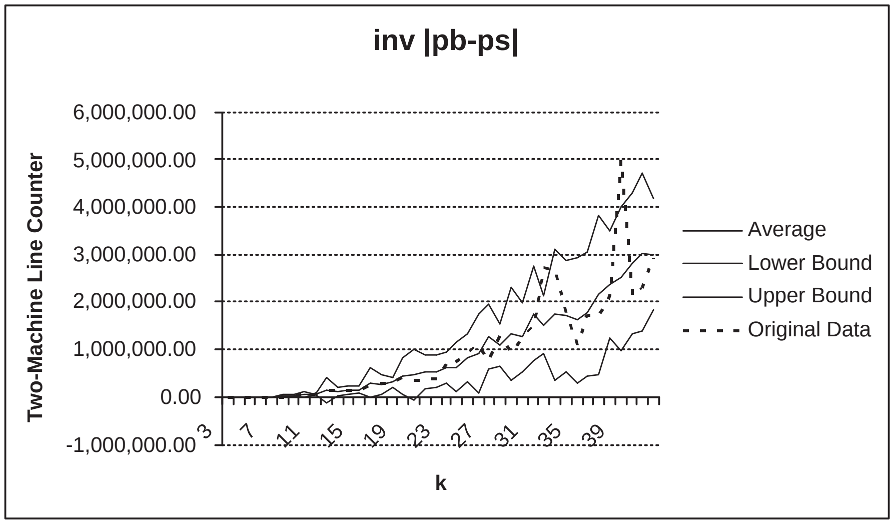


Figure 5-5: Sensitivity Results of the $\frac{1}{|p_b(i)-p_s(i)|}$ Algorithm on Identical Machine Lines

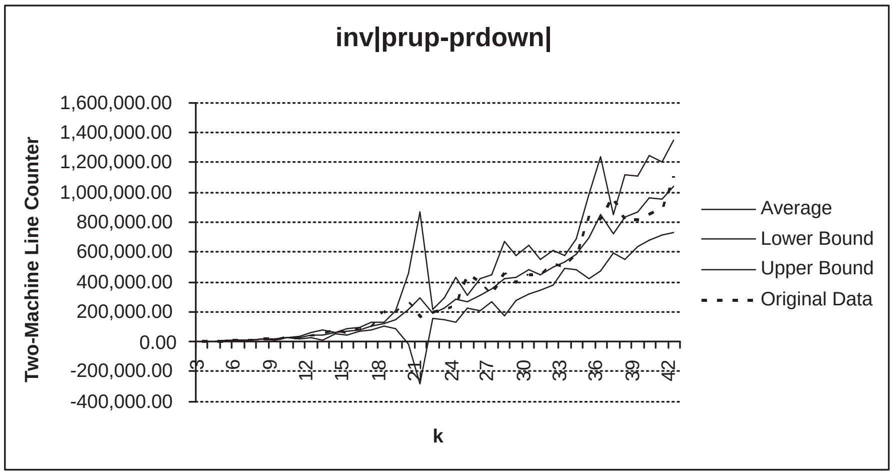


Figure 5-6: Sensitivity Results of the $\frac{1}{|P_u(i)-P_d(i)|}$ Algorithm on Identical Machine Lines

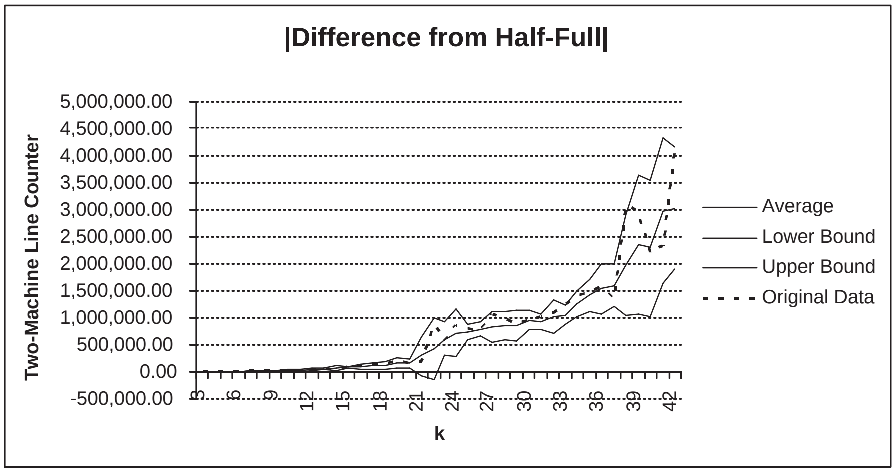


Figure 5-7: Sensitivity Results of the $|\overline{n_i} - \frac{N_i}{2}|$ Algorithm on Identical Machine Lines

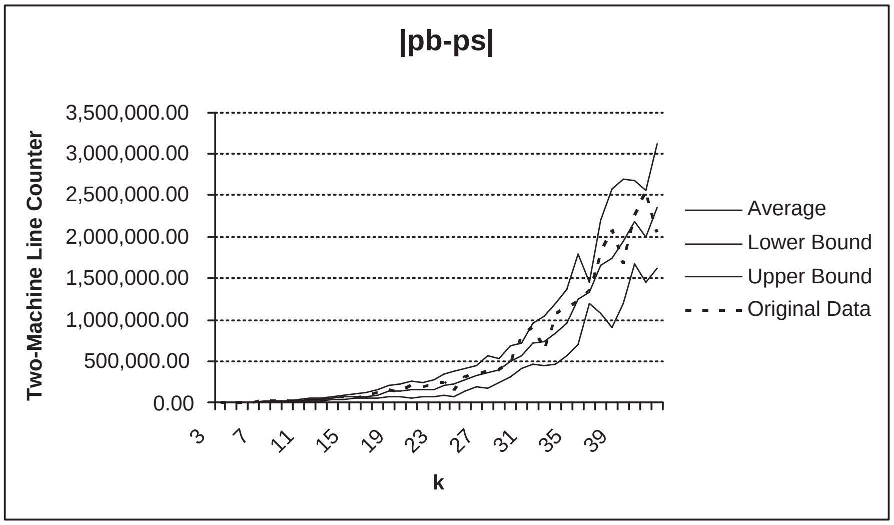


Figure 5-8: Sensitivity Results of the $|p_b(i) - p_s(i)|$ Algorithm on Identical Machine Lines

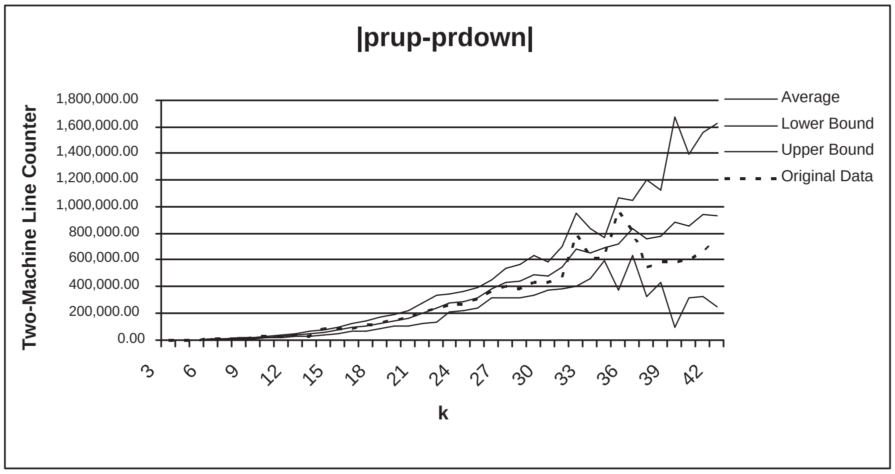


Figure 5-9: Sensitivity Results of the $|P_u(i) - P_d(i)|$ Algorithm on Identical Machine Lines

Table 5.2: The Original Line Parameters Used in Sensitivity Analysis of Non-Identical Machine Line

Machine No	Repair Rate r	Failure Rate p	Processing Rate μ
1	23.542	7.45766	207.159
2	19.4615	7.0366	198.491
3	23.2078	8.32248	202.648
4	29.1467	4.75568	173.817
5	23.9343	5.67139	181.414
6	16.0395	6.65849	155.188
7	19.0273	7.96522	203.335
8	20.8448	7.9356	205.085
9	28.78	8.19109	186.833
10	16.167	6.2691	198.694
11	29.0583	6.48092	188.112
12	31.4833	7.05615	183.841
13	27.0158	5.33145	189.174
14	21.0023	7.43117	197.2
15	16.0448	5.11554	200.766
16	31.4508	7.44149	192.053
17	20.5974	4.66799	191.73
18	29.1921	8.8115	194.431
19	23.8323	5.09999	187.437
20	25.9468	8.68883	192.26
21	23.9365	6.36989	195.852
22	21.3463	9.07593	205.645
23	17.5735	5.08346	196.552

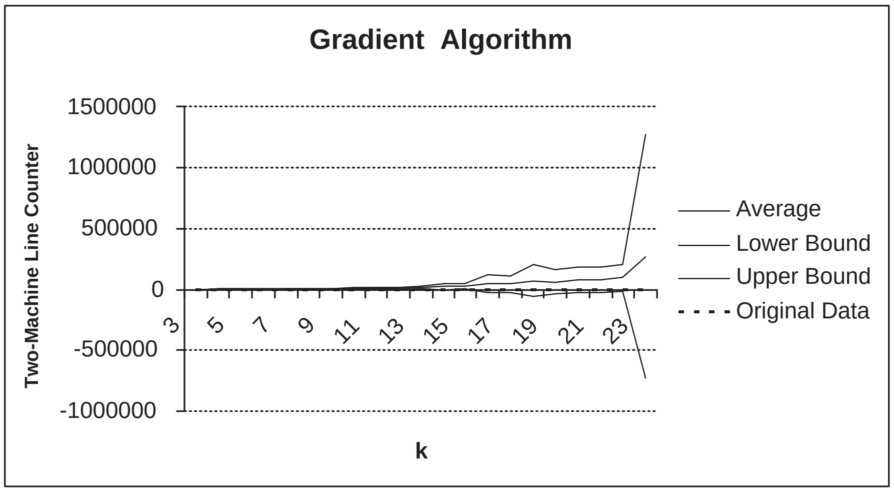


Figure 5-10: Sensitivity Results of the Gradient Algorithm on Non-Identical Machine Lines

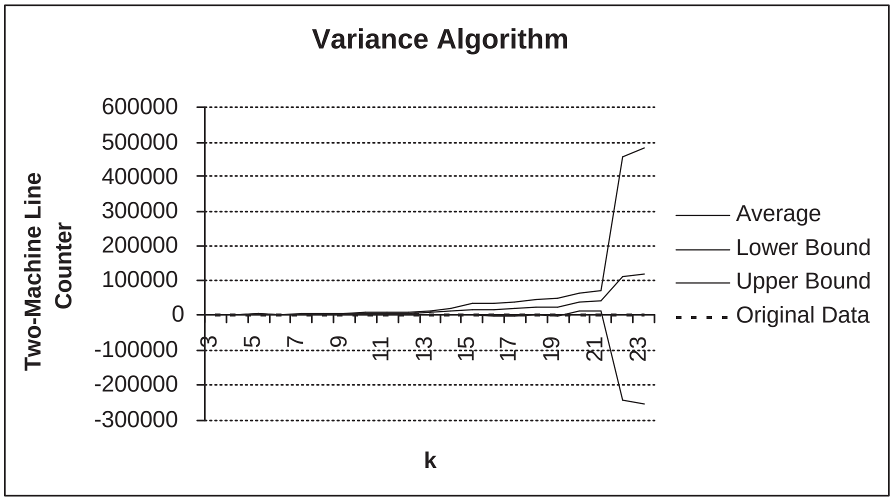


Figure 5-11: Sensitivity Results of the Variance Algorithm on Non-Identical Machine Lines

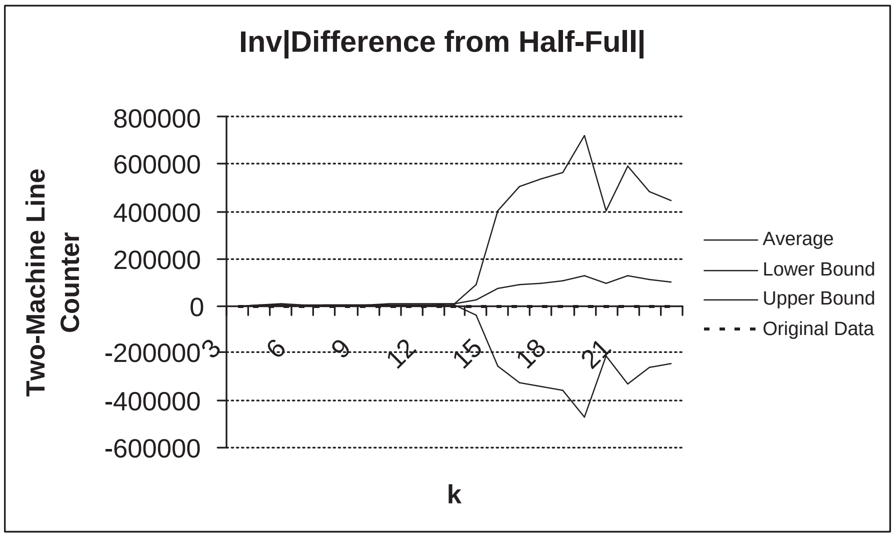


Figure 5-12: Sensitivity Results of the $\frac{1}{|\overline{n_i}-\frac{N_i}{2}|}$ Algorithm on Non-Identical Machine Lines

Chapter 6

Conclusion and Future Research

6.1 Conclusion

In the identical-machine lines, the variance and standard deviation work very well as substitutes for the gradient. This is because their final P are as high as the final P of the gradient algorithm and the convergence times of the two algorithms are, on average, half that of the gradient algorithm. We do not recommend using $\frac{1}{|\overline{n_i}-\frac{N_i}{2}|}$ and $\frac{1}{|p_b(i)-p_s(i)|}$ to replace the g_i because their convergence times might be as long as four times the convergence times of the gradient. The other directions ($\frac{1}{|P_u(i)-P_d(i)|}$, $|\overline{n_i} - \frac{N_i}{2}|$, $|p_b(i) - p_s(i)|$, and $|P_u(i) - P_d(i)|$) do not outperform the gradient.

The variance and the standard deviation algorithms also do well in the non-identical machine case, although not as well as in the identical machine case. We do not recommend using $\frac{1}{|P_u(i)-P_d(i)|}$, $|\overline{n_i} - \frac{N_i}{2}|$, and $|P_u(i) - P_d(i)|$ to replace the gradient in the non-identical machine case because their final two-machine line counters are almost as high or higher than the final two-machine line counters of the gradient. In addition, their final two-machine line counters vary greatly.

We also conclude that the new algorithms perform better in lines with no bottlenecks than in lines with bottlenecks. The results that the new algorithms perform better in the non-bottleneck cases are even more apparent in longer lines.

In both identical and non-identical machine cases, the location of bottlenecks does not strongly affect the final P of the new algorithms. The speeds of the new algorithms, except the variance and the standard deviation algorithms, are slightly affected by the location of the bottleneck. Unfortunately, we could not explain how the speed of the new algorithms vary with the locations of the bottleneck.

The new algorithms are reliable. When the buffer allocation is initialized using the three different buffer initialization methods (described in Chapter 4), the final P of the new algorithms are close to one another. The new algorithms are also reliable in the non-identical machine case.

In short identical-machine lines, the new algorithms converge faster than the gradient. As lines get longer, most of the new algorithms (except the variance and the standard deviation) take longer to converge than the gradient. When the identical-machine lines are long, the final two-machine line counters of the variance and the standard deviation are similar to that of the gradient and their final P are slightly higher than the final P of the gradient method. In non-identical machine lines, the gradient algorithm takes longer to converge even at the case of long lines.

We also test how sensitive the new algorithms are to small changes in machine parameters. We find that the new algorithms are sensitive to up to 10% changes to machine parameters. We believe that the new algorithms will be less sensitive to changes less than 10% of the parameters of the original line.

6.2 Future Research

The new directions, especially the variance and the standard deviation, have shown to be good substitutes for the gradient. Although their final P might not be as high as the final P of the gradient, most of them converge faster than the gradient. Therefore, one possible future research is to run together the new algorithms until a specified time period, observe the buffer allocation at that time, and finally use the gradient algorithm from that time on. By doing so, we expect to converge to the highest possible final P with less time than if we use the algorithms individually.

Figure 1-4 and Figure 1-5 show that the alternative directions have the same shape or inverse the shape of the optimal buffer allocation at the optimal point. Therefore, another possible future research is to use the alternative directions to replace the buffer allocation, instead of to replace to gradient. This means that at every buffer allocation, the alternative directions are calculated and Buffer B_i is allocated in proportion to the component i of the alternative direction. This process continues until the terminating criterion is satisfied. We expect that using the alternative directions to directly replace the buffer size might lead to a better result than using the alternative directions to replace the gradient.

Finally, another possible research involves using linear regression. First, we need to gather enough data from previous experiment. The data consists of gradient i, the variance i, the standard deviation i, $\frac{1}{|\overline{n_i}-\frac{N_i}{2}|}$, $\frac{1}{|p_b(i)-p_s(i)|}$, $\frac{1}{|P_u(i)-P_d(i)|}$, $|\overline{n_i} - \frac{N_i}{2}|$, $|p_b(i) - p_s(i)|$, and $|P_u(i) - P_d(i)|$ at the optimal buffer allocation. Later, we regress gradient i on all possible directions: the variance i, the standard deviation i, $\frac{1}{|\overline{n_i}-\frac{N_i}{2}|}$, $\frac{1}{|p_b(i)-p_s(i)|}$, $\frac{1}{|P_u(i)-P_d(i)|}$, $|\overline{n_i}-\frac{N_i}{2}|$, $|p_b(i)-p_s(i)|$, and $|P_u(i)-P_d(i)|$. From this regression, we get the regression coefficients. This possible research requires using the linear combination of all the possible directions as the direction to replace the gradient.

Appendix A

Generating Realistic Line Parameters

A.1 Motivation

In order for the algorithms' results to be meaningful, the input data fed into the algorithms should be as realistic as possible. In this appendix, we define an efficient algorithm to generate large numbers of data sets consisting of realistic machine parameters and realistic total buffer space to be allocated in a transfer line. An efficient algorithm is needed because we expect computers to spend more time performing the actual buffer optimization, rather than to spend more time generating the input data fed to the optimization algorithm.

A.2 Realistic Constraints on Parameters

The algorithm is generated such that the line parameters (r_i, p_i, μ_i, ρ_i, and N^{TOTAL}) follow a set of constraints. These constraints must be satisfied to create lines with no bottlenecks. Lines with bottlenecks are created using an algorithm that is a modification of the algorithm used to create lines with no bottlenecks. The implementations of both algorithms can be found in Section A.3.

The following are the set of constraints:

1. Similarity of r, p, μ, and ρ

   These four sets of constraints are used to ensure that the repair rate r, the failure rate p, the processing speed μ, and the isolated production rate ρ of all machines are not very different from one another. These similarity constraints are important because if one machine is very different from the other machines, either in terms of r, p, μ, or ρ, that different machine will disturb the flow of the production line. For example, if μ_3 is 1 part/unit time and μ_s of the other machines are 100 parts/unit time, the throughput of the flow line will depend solely on Machine M_3. Therefore, the capability of the other machines to produce 100 parts/unit time is wasted.

   The similarity constraints are as follows:

$$L_r \leq \frac{r_i}{r_j} \leq U_r$$

$$L_p \leq \frac{p_i}{p_j} \leq U_p$$

$$L_\mu \leq \frac{\mu_i}{\mu_j} \leq U_\mu$$

$$L_\rho \leq \frac{\rho_i}{\rho_j} \leq U_\rho$$

$$\forall i, j; i, j = 1, ..., k$$

   Here, L_r, L_p, L_μ, L_ρ, U_r, U_p, U_μ and U_ρ are constants. L_r, L_p, L_μ and L_ρ represent the lower bounds on the constraints. On the other hand, U_r, U_p, U_μ and U_ρ represent the upper bounds. Readers can use any bound values they deem applicable to their problem.

2. Isolated Efficiency

   The isolated efficiency constrains are used to ensure than the isolated efficiencies of all machines are realistic.

   The isolated efficiency constraint is:

$$L_e \leq \frac{r_i}{r_i + p_i} \leq U_e$$

$$\forall i; i = 1, ..., k$$

Similarly, L_e is the lower bound and L_e is the upper bound.

3. Relationship between μ and r

   Usually, it takes much more time to repair a machine than to produce a part on that machine. Therefore, it is desirable that the machine's processing speed μ is much greater than the repair rate r. To ensure that μ is much greater than r, we impose the following constraint:

$$\mu_i \geq M r_i$$

$$\forall i; i = 1, ..., k$$

   Here, M is a constant. Readers can choose any value of M applicable to their problem.

4. Total Buffer Spaces to be Allocated N^{TOTAL}

   This constraint is needed because during the down times of Machine M_i, the number of parts that fill Buffer B_{i-1} and empty Buffer B_i are limited. This constraint specifies a realistic range of the number of repair events it takes to empty of fill an average buffer.

   Let us observe the units of the following parameters:

   - N_i has units of **number of parts**
   - μ_j has units of **number of parts produced per unit time**
   - r_j has units of **number of repair events per unit time**

Let us first analyze the fraction $\frac{N_i}{\mu_j}$, which has units of (number of parts/buffer)(unit time/number of parts) = unit time/buffer. If we require that Machine M_i and Machine M_j to be next to each other, we can interpret $\frac{N_i}{\mu_j}$ as the time it takes to fill or empty a buffer; $\forall |i-j| \leq 1$.

Multiplying $\frac{N_i}{\mu_j}$ and r_j, we get $\frac{N_i r_j}{\mu_j}$, which can be interpreted as the expected number of repair events required to empty or fill a buffer.

In reality, the number of repair events to empty or to fill a buffer is also limited. We propose that a realistic expected number of repair events required to empty or fill a buffer is between $\frac{1}{5}$ and 5. As before, readers can modify this range with the one they believe more applicable to their problem.

Recall that $\frac{N_i r_j}{\mu_j}$ can be interpreted as the expected number of repair events required to empty or fill a buffer. If we let $N^* = \sum_{i=1}^{k} \frac{\mu_i}{r_i}$, we can interpret $\frac{\sum_{i=1}^{k-1} N_i}{\sum_{i=1}^{k} \frac{\mu_i}{r_i}} = \frac{N^{TOTAL}}{N^*}$ as an approximate measure of the number of repair events to empty or fill an average buffer.

The total buffer spaces constraint is the constraint that specifies a realistic range of the number of repair events it takes to empty or fill an average buffer. That is,

$$L_N \leq \frac{N^{TOTAL}}{N^*} \leq U_N$$

Here. L_N and U_N are constants, with L_N represents the lower bound of the constraint and U_N represents the upper bound.

A.3 Implementation

A.3.1 Difficulties with Filtering

We need a simple and fast algorithm to generate r_i, p_i, and μ_i such that all four set of constraints are satisfied. We first consider an algorithm that generates r_i, p_i, and μ_i of each machine independently. This is done by generating r_i, p_i, and μ_i from a uniform distribution between their lower bound and upper bound. This step is repeated to generate r_j, p_j, and $\mu_j, \forall j = i+1, i+2, ..., k$. This algorithm discards the parameters of machines that do not satisfy the specified set of constraints. This filtering process continues until k sets of machine parameters that satisfy all the constraints are found. As expected, this filtering algorithm takes very long time. Moreover, the longer the line, the longer it takes to find the set of machine parameters that satisfy all the constraints.

MORE SPECIFIC HOW LONG?

We propose a different algorithm that avoids using any filtering approach. We will first describe an algorithm to generate line with non-bottleneck machines and later describe an algorithm to generate lines with bottleneck machines.

A.3.2 Generating Lines with non-Bottleneck Machines

The description of the algorithm is as follows:

1. The number of machines k is randomly generated from a uniform distribution.

2. Generating parameters of Machine M_1

   (a) The repair rate r_1 is first generated from a uniform distribution between 0 and the maximum repair rate specified.

   (b) p_1 needs to be generated such that the isolated efficiency constraint is satisfied. That is,

   $$L_e \leq \frac{r_1}{r_1 + p_1} \leq U_e \tag{A.1}$$

   Rearranging (A.1) results in:

   $$\frac{1 - U_e}{U_e} r_1 \leq p_1 \leq \frac{1 - L_e}{L_e} r_1$$

   p_1 can now be generated from a uniform distribution from $\frac{1-U_e}{U_e} r_1$ to $\frac{1-L_e}{L_e} r_1$.

(c) μ_1 needs to be generated such that the relationship between μ and r constraint is satisfied. That is,

$$\mu_1 \geq Mr_1$$

μ_1 can now be generated from a uniform distribution from Mr_1 to the maximum μ_1 specified.

3. Generating the parameters of Machine M_2.

Since the parameters of Machine M_1 are already generated in step 2, we can use the parameters of Machine M_1 to generate the parameters of Machine M_2.

(a) Generating r_2

Machine M_2 needs to be generated such that the repair rate r_2 satisfies the similarity of r constraint. That is,

$$L_r \leq \frac{r_2}{r_1} \leq U_r \tag{A.2}$$

Rearranging (A.2) yields:

$$L_r r_1 \leq r_2 \leq U_r r_1$$

r_2 can now be generated from a uniform distribution $[L_r r_1, U_r r_1]$.

(b) Generating p_2

At this point, the parameters r_1, p_1, μ_1, and r_2 are already generated from the previous steps. To generate p_2, we need to satisfy all the constraints that involve p_2. The first constraint is the similarity of p constraint, which is:

$$L_p \leq \frac{p_2}{p_1} \leq U_p \tag{A.3}$$

The second one is the isolated efficiency constraint, which is:

$$L_e \leq \frac{r_2}{r_2 + p_2} \leq U_e \tag{A.4}$$

Rearranging (A.3) yields:

$$L_p p_1 \leq p_2 \leq U_p p_1$$

Rearranging (A.4) yields:

$$\frac{1 - U_e}{U_e} r_2 \leq p_2 \leq \frac{1 - L_e}{L_e} r_2$$

Since p_2 needs to satisfy (A.3) and (A.4), there are two lower bounds and two upper bounds resulting from the two sets of inequalities.

The two lower bounds are: $L_p p_1$ and $\frac{1-U_e}{U_e} r_2$; while the two upper bounds are $U_p p_1$ and $\frac{1-L_e}{L_e} r_2$.

p_2 can now be generated from a uniform distributed between the two tightest bounds, i.e from the highest lower bound to the lowest upper bound. That is, p_2 is generated from a uniform distribution $[\max(L_p p_1, \frac{1-U_e}{U_e} r_2), \min(U_p p_1, \frac{1-L_e}{L_e} r_2)]$.

(c) Generating μ_2

Machine M_2 needs to be generated such that the processing rate μ_2 satisfies all the constraints that involve μ_2.

The first constraint is the similarity of μ constraint, which is:

$$L_\mu \leq \frac{\mu_2}{\mu_1} \leq U_\mu \tag{A.5}$$

The second constraint is the similarity of ρ constraint, which is:

$$L_\rho \leq \frac{\rho_2}{\rho_1} \leq U_\rho \tag{A.6}$$

Finally, the last constraint is the relationship between μ and r constraint.

$$\mu_2 \geq Mr_2 \tag{A.7}$$

Rearranging (A.5) yields:

$$L_\mu \mu_1 \leq \mu_2 \leq U_\mu \mu_1$$

Rearranging (A.6) yields:

$$L_\rho \rho_1 \leq \rho_2 \leq U_\rho \rho_1,$$

Since $\rho = \frac{\mu r}{r+p}$, (A.6) can also be written as:

$$L_\rho \rho_1 \leq \frac{\mu_2 r_2}{r_2 + p_2} \leq U_\rho \rho_1$$

$$L_\rho \rho_1 \frac{r_2 + p_2}{r_2} \leq \mu_2 \leq U_\rho \rho_1 \frac{r_2 + p_2}{r_2}$$

Since μ_2 has to satisfy (A.5),(A.6), and (A.7), there are three lower bounds and two upper bounds for μ_2.

The three lower bounds are: $L_\mu \mu_1, L_\rho \rho_1 \frac{r_2+p_2}{r_2}$, and Mr_2 ; whereas the two upper bounds are: $U_\mu \mu_1$ and $U_\rho \rho_1 \frac{r_2+p_2}{r_2}$.

μ_2 can now be generated from a uniform distribution between the two tightest bounds, i.e from the highest lower bound to the lowest upper bound. That is, μ_2 is generated from a uniform distribution

$[\max(L_\mu \mu_1, L_\rho \rho_1 \frac{r_2+p_2}{r_2}, Mr_2), \min(U_\mu \mu_1, U_\rho \rho_1 \frac{r_2+p_2}{r_2})]$.

4. Generating the parameters of Machine $M_3, M_4, M_5, ..., M_k$

   The parameters of Machines $M_3, M_4, M_5, ..., M_k$ are generated in the same way as the parameters of Machine M_2 are generated. We first identify the set of constraints that r_i, p_i, and μ_i need to satisfy. These set of constraints are written with respect to the r, p, and μ of each of the earlier machines. Each of the constraint will contribute a lower bound and/or an upper bound. We next find the tightest lower and the tightest upper bound of all the constraints and generate r, p, or μ from a uniform distribution from the tightest lower bound to the tightest upper bound. The greater the k is (or the farthest away the machine is located downstream), the larger the number of machines located upstream of Machine M_k and therefore, the larger the number of constraints that need to be written for r_k, p_k, and μ_k.

5. Generating the total amount of buffer space to be allocated, N^{TOTAL}.

   Recall that N^{TOTAL} needs to satisfy:

$$L_N \leq \frac{N^{TOTAL}}{N^*} \leq U_N \tag{A.8}$$

   where $N^* = \sum_{i=1}^{k} \frac{\mu_i}{r_i}; \forall i, i = 1, ..., k-1$.

   Rearranging (A.8) yields:

$$L_N N^* \leq N^{TOTAL} \leq L_N N^*$$

   N^{TOTAL} can now be generated from a uniform distribution $[L_N N^*, U_N N^*]$.

A.3.3 Generating Lines with Bottleneck Machines

The description of the algorithm to generate lines with bottleneck machines is as follows:

1. Generate all machine parameters and the total buffer spaces to be allocated, using the algorithm used to generate non-bottleneck machines.

2. Randomly choose the number, location, and severity of the bottlenecks. The severity of the bottleneck is measured by a severity factor γ. A high γ indicates a less severe bottleneck. Likewise, a low γ indicates a more severe bottleneck.

3. Generate the parameters of the bottleneck machines such that:

- The isolated production rate of the bottleneck machine is less than γ multiplied by the minimum isolated production rate of the other non-bottleneck machines. That is,

$$\rho_{bottleneck} \leq \gamma \min \rho_{non-bottleneckmachines} \tag{A.9}$$

  Or,

- The isolated production rate of the bottleneck machine is between two severity factors, γ_1 and γ_2, multiplied by the minimum isolated production rate of the non-bottleneck machines. That is,

$$\gamma_1 \min \rho_{non-bottleneckmachines} \leq \rho_{bottleneck} \leq \gamma_2 \min \rho_{non-bottleneckmachines} \tag{A.10}$$

Since $\rho = \mu \frac{r}{r+p}$, we can reduce the value of either μ, r, or p of the machine randomly chosen as the bottleneck so that the modified machine becomes a bottleneck. We call the previously non-bottleneck machine randomly chosen to become a bottleneck as the *original machine*.

There are three types of bottleneck machines:

(a) μ-bottleneck

If μ of the original machine is reduced, the bottleneck machine becomes a μ-bottleneck.

(b) r-bottleneck

If r of the original machine is reduced, the bottleneck machine becomes an r-bottleneck.

(c) p-bottleneck

Finally, if p of the original machine is reduced, the bottleneck machine becomes a p-bottleneck.

For simplicity, in this research we only modify the μ of the original machines, creating only μ-bottlenecks.

Rearranging (A.10) yields:

$$\frac{r_{bottleneck} + p_{bottleneck}}{r_{bottleneck}} \gamma_1 \min \rho_{non-bottleneck machines} \leq \mu$$

and

$$\mu \leq \frac{r_{bottleneck} + p_{bottleneck}}{r_{bottleneck}} \gamma_2 \min \rho_{non-bottleneck machines}$$

Finally, the μ bottleneck can now be generated from a uniform distribution between the lower bound and the upper bound of above inequality.

A.4 Example

To see how the algorithm performs, a transfer line with 40 unreliable machines is generated. The N_{TOTAL} generated was 2402. Figure A-1 shows the upper bounds, the lower bounds, and the values of r generated for all machines. Figure A-2 shows the upper bounds, the lower bounds, and the values of p generated. Finally, Figure A-3 shows the upper bounds, the lower bounds, and the values of μ for all machines. In Figure A-3, μ_{27} falls outside its lower and upper bounds because Machine M_{27} is the bottleneck.

From Figure A-1, Figure A-2, and Figure A-3, we observe that the upper and lower bounds do not converge to one point. We do not want the bounds to converge because if they do, the downstream machines become increasingly similar to one another–which do not represent a realistic transfer line.

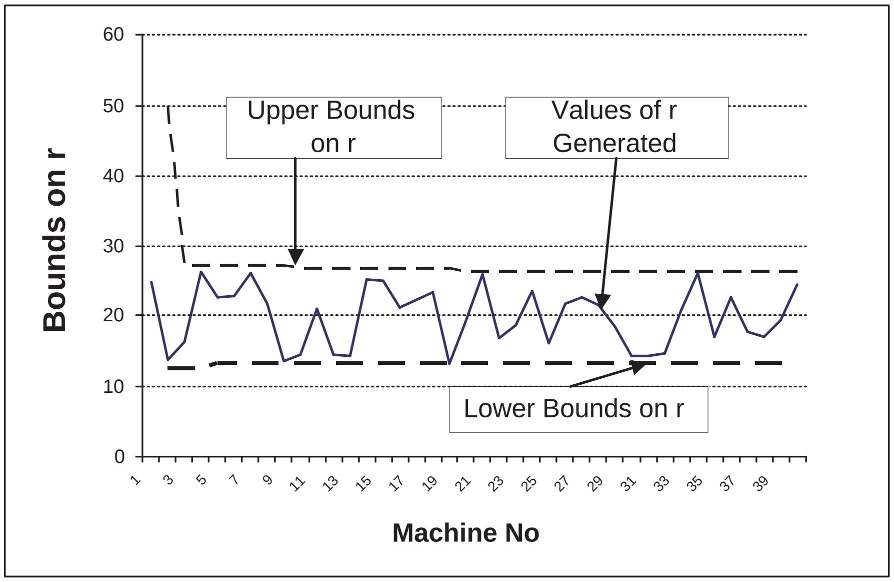


Figure A-1: Upper Bounds, Lower Bounds, and the Values of r Generated

A.5 Conclusion

This algorithm can be used to generate realistic machine parameters. It avoids any filtering and is fast and reliable.

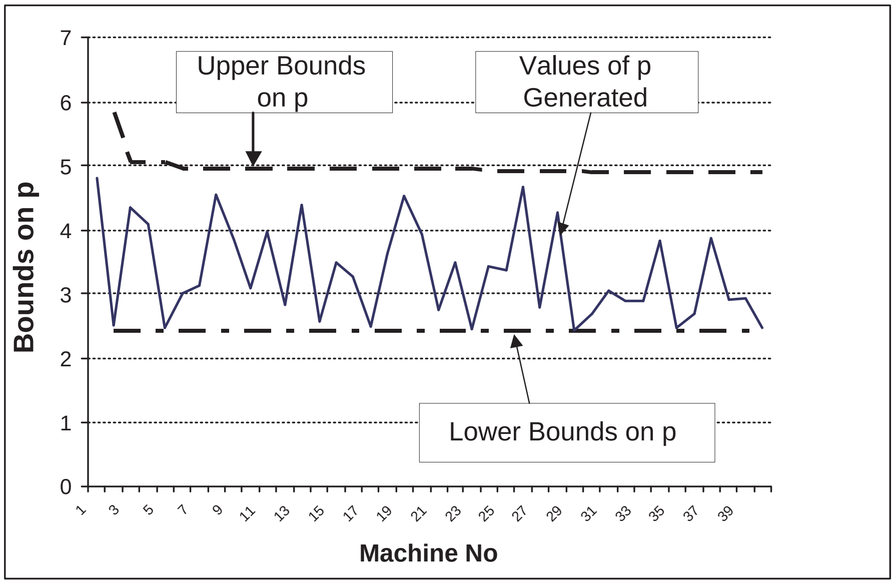


Figure A-2: Upper Bounds, Lower Bounds, and the Values of p Generated

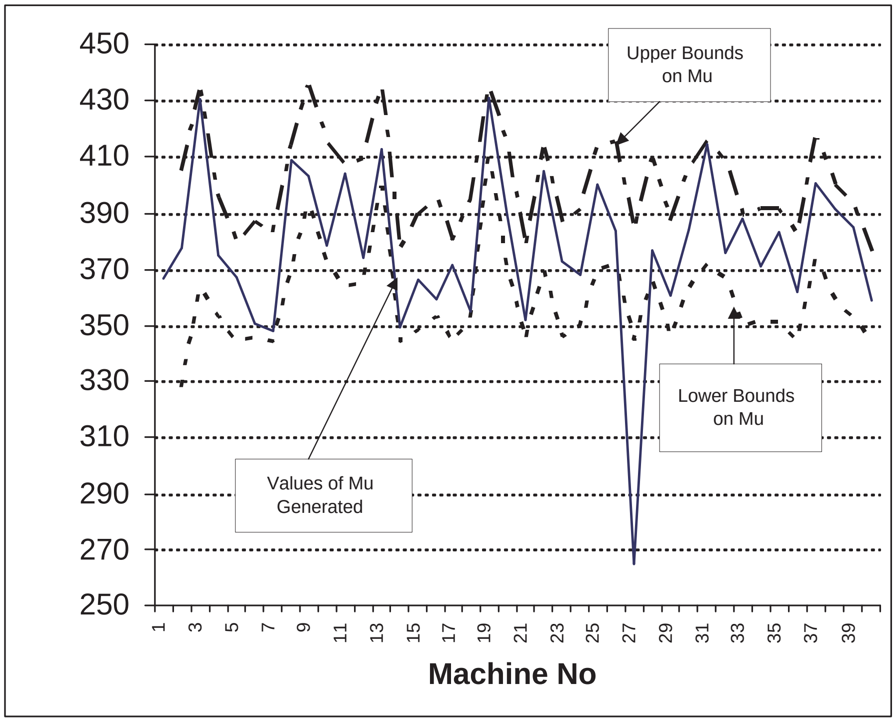


Figure A-3: Upper Bounds, Lower Bounds, and the Values of μ Generated

Appendix B

Linear Search to Find the Next Point on the Constraint Space

Schor's Gradient Algorithm starts from an initial guess N. After selecting the initial guess, a direction to move in the constraint space $N^{TOTAL} = \sum_{i=1}^{k-1} N_i$ is determined. A linear search is conducted along that direction until a point that has the maximum production rate is encountered. This new point, N^{new}, becomes the next guess. A new search direction is calculated from this new guess.

Figure B-1 shows the illustrations of N, N^{new}, the search direction vector $\prod$, and the constraint space for $k = 4$.

Mathematically, N^{new} is defined as $N + a \prod$. The linear search requires finding a scalar a in the direction of vector $\prod$ such that the production rate of N^{new} is the highest production rate of all points along $\prod$ on the constraint space.

B.1 Binary Search

Before we describe our algorithm, we first describe the well-known Binary Search Algorithm [Binary04], which is used to search a number from a sorted array. The Binary Search function takes four parameters: the search term, the array of the sorted data, and the high and low values of data to check. The low value and high value are added so that we can collapse the search area during each level of recursion. The search term is then compared to the value in the middle of the range. If it matches or if the high and low are equal, then the Binary Search terminates. Otherwise, we compare the search term to the middle value of the range and if the middle is too high, we return the results of searching the lower half of the array. If the middle is too low, then we search the top half.

B.2 New Search Algorithm

Our algorithm is a slight modification of the Binary Search. Like the Binary Search, the new algorithm also identifies a region and later collapses the region until the search term is found. The search term is the scalar a^*, of which $N + a^* \prod$ has the highest P of all. Unlike the Binary Search, which sorts an array of data, our algorithm needs to be able to compare the production rates associated with the scalar a_s. In order to help compare the P values and to find the highest P of all, we introduce a *minmid* value and a *midmax* value, in addition to the low value (or *min*), the high value (or *max*), and the middle value (or *mid*) of the region. The *minmid* value is the middle value between *min* and *mid*. The *midmax* value is the middle value between *mid* and *max*. *Minmid* and *Midmax* are required in order to capture the concavity of the production rate function.

The followings are the steps of the algorithm, assuming we are now located at the initial guess, N:

1. Calculate how far we can go before we violate the constraint $N^{TOTAL} = \sum_{i=1}^{k-1} N_i$.

   Let a^*_{min} be a scalar such that $N + a^*_{min} \prod = N^{min}$, the point on the edge of the constraint space, in the direction opposite of $\prod$.

   Similarly, let a^*_{max} be a scalar such that $N + a^*_{max} \prod = N^{max}$, the point on the edge of the constraint space, in the same direction as $\prod$. The algorithm begins by first calculating a^*_{min} and a^*_{max}.

   Figure B-1 shows the location of N^{min} and N^{max}.

Due to the concavity property of the dual problem, the production rates of all points along the line parallel to by $\prod$ follow a concave function. a^*_{min} and a^*_{max} are the two points at the opposite ends of the horizontal axis of the production rate function.

Figure B-2 shows a^*_{min} and a^*_{max}, as the opposite ends of the line formed by $\prod$. Here, the horizontal axis is the same as the dotted line formed by vector $\prod$ in Figure B-1.

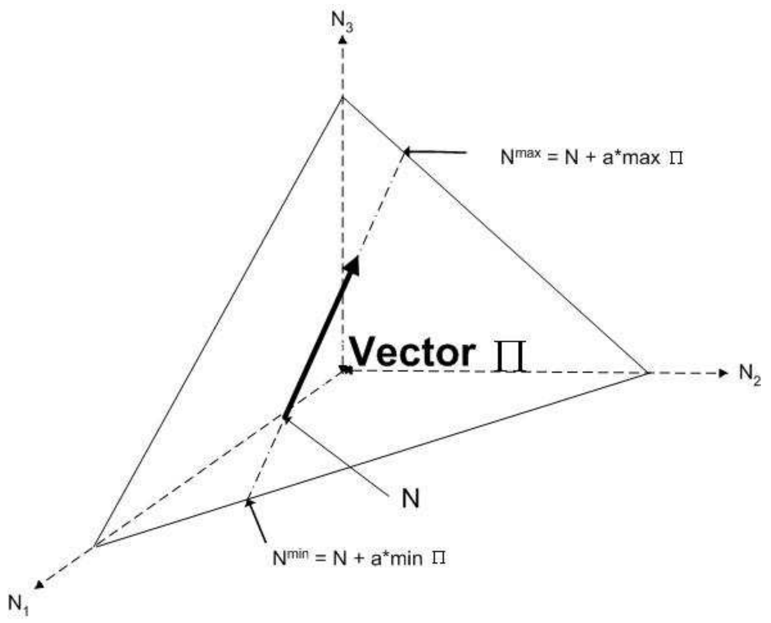


Figure B-1: N_min,N_max and $\prod$ on the Constraint Space for $k = 4$

Let a_{min} be the current left end of the horizontal axis of the production rate function, and a_{max} be the current right end of the horizontal axis of the production rate function.

Let

$temp_{amin}$ be a variable that holds the temporary value of a_{min},

$temp_{amax}$ be a variable that holds the temporary value of a_{max},

$temp_{amid}$ be a variable that holds the middle value of $temp_{amin}$ and $temp_{amax}$,

$temp_{aminmid}$ be a variable that holds the middle value of $temp_{amin}$ and $temp_{amid}$,

$temp_{amidmax}$ be a variable that holds the middle value of $temp_{amid}$ and $temp_{amax}$.

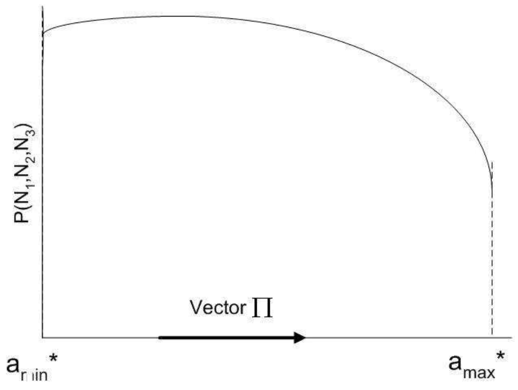


Figure B-2: a^*_{min}, a^*_{max}, and the Production Rate Function along the Line Parallel to $\prod$ for $k = 4$

Initially, the region to consider is the whole line parallel to $\prod$. Therefore, the algorithm starts by setting

$$temp_{amin} = a_{min} = a^*_{min}$$

$$temp_{amax} = a_{max} = a^*_{max}$$

2. Calculate the value of the midpoint of the region considered

   If we are going through the algorithm for the first time, we let $temp_{amid} = 0$. Figure B-3 helps explain why $temp_{amid}$ needs to be set to 0 for the first iteration.

   Let

   P_{amin} be the production rate of $temp_{amin}$,

   $P_{aminmid}$ be the production rate of $temp_{aminmid}$,

   P_{amid} be the production rate of $temp_{amid}$,

   $P_{amidmax}$ be the production rate of $temp_{amidmax}$,

   P_{amax} be the production rate of $temp_{amax}$, and

   P_max be the optimal point of the P function.

   In Figure B-3, the P function is concave with $P_{amin} \geq P_{amax}$. If $temp_{amid} \neq 0$ (in this case, $temp_{amid} = \frac{temp_{amin}+temp_{amax}}{2}$), the next region to be considered is the region between $temp_{amin}$ and $temp_{amid}$. Therefore, the scalar a with the actual highest P of all will not be included in the region. To ensure that the scalar a with the highest P be included in the next region, we set $temp_{amid} = 0$.

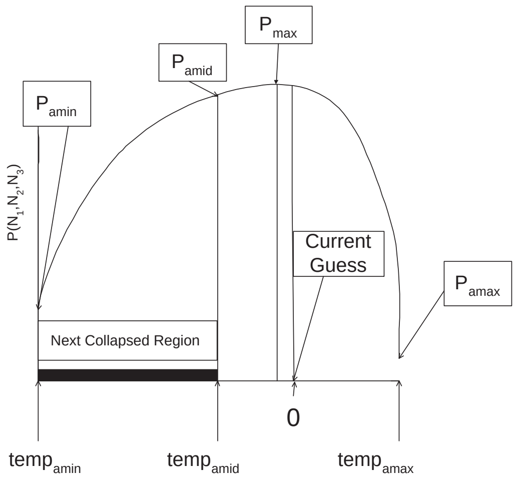


Figure B-3: An Example when at First Iteration $temp_{amid} \neq 0$

Figure B-4 shows the locations of $temp_{amin}, temp_{aminmid}, temp_{amid}, temp_{amidmax}, temp_{amax}$, and the scalar a with has the highest production rate, P_{max}.

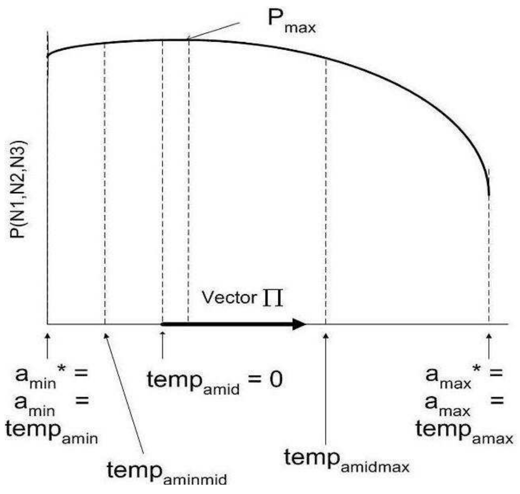


Figure B-4: Locations of $temp_{amin}, temp_{aminmid}, temp_{amid}, temp_{amidmax}, temp_{amax}$

After the first iteration of the algorithm, we can set $temp_{amid}$ to be the middle value of $temp_{amin}$ and $temp_{amax}$. That is,

$$temp_{amid} = \frac{temp_{amin} + temp_{amax}}{2}$$

3. Calculate $temp_{aminmid}$ and $temp_{amidmax}$

   $temp_{aminmid}$ and $temp_{amidmax}$ are calculated as follows:

$$temp_{aminmid} = \frac{temp_{amin} + temp_{amid}}{2}$$

$$temp_{amidmax} = \frac{temp_{amid} + temp_{amax}}{2}$$

4. Calculate the production rate of $temp_{amin}, temp_{aminmid}, temp_{amid}, temp_{amidmax}$, and $temp_{amax}$

   Figure B-5 shows the locations of P_{amin}, $P_{aminmid}$, P_{amid},$P_{amidmax}$, and P_{amax}.

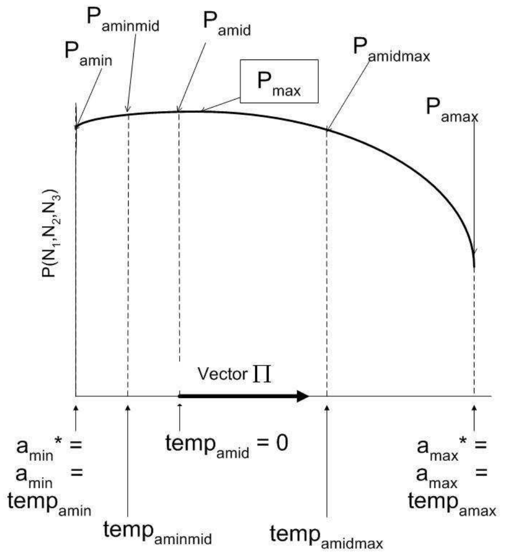


Figure B-5: Locations and Values of P_{amin}, $P_{aminmid}$, P_{amid},$P_{amidmax}$, and P_{amax}

5. Analyze and compare P_{amin}, $P_{aminmid}$, P_{amid}, $P_{amidmax}$, and P_{amax}

In this step, we compare P_{amin}, $P_{aminmid}$, P_{amid}, $P_{amidmax}$, and P_{amax} so that we can start collapsing the region.

The steps to comparing P_{amin}, $P_{aminmid}$, P_{amid}, $P_{amidmax}$, and P_{amax} are as follows:

(a) If the P Function is Monotonically Increasing

It is,

$$P_{amin} \leq P_{amid} \leq P_{amax}$$

Figure B-6 shows a monotonically increasing P function, with all the values of P_{amin}, $P_{aminmid}$, P_{amid},$P_{amidmax}$, and P_{amax}.

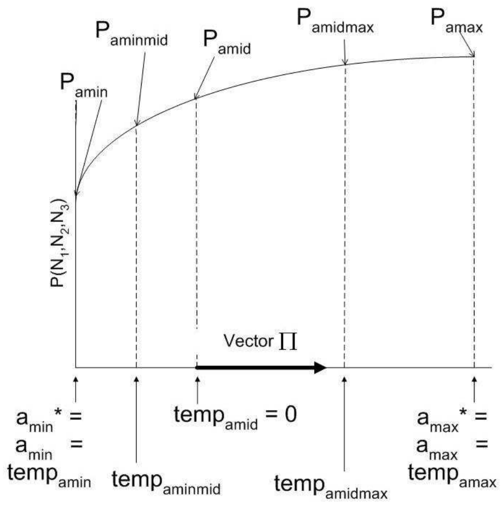


Figure B-6: A Monotonically Increasing Production Rate Function

In the case of a monotonically increasing P function, we only need to search the region between $temp_{amid}$ and $temp_{amax}$ because this is the region that includes the scalar a with the optimal production rate P_{max}. Therefore, the new a_{min}, a_{max}, and a_{mid} are calculated as follows:

$$a_{min} \quad = \quad temp_{amid}$$

$$\begin{aligned} a_{max} &= temp_{amax} \\ a_{mid} &= \frac{temp_{amid} + temp_{amax}}{2} \end{aligned}$$

Let P_{avg} be the production rate of the midpoint of the collapsed region. a_{mid} is now the middle point of a_{mid} and a_{max}. Therefore, P_{avg} is the production rate of $temp_{amidmax}$. That is,

$$P_{avg} = P_{amidmax}$$

Figure B-7 shows the new collapsed region, with the locations of the new a_{min}, a_{mid}, and a_{max}.

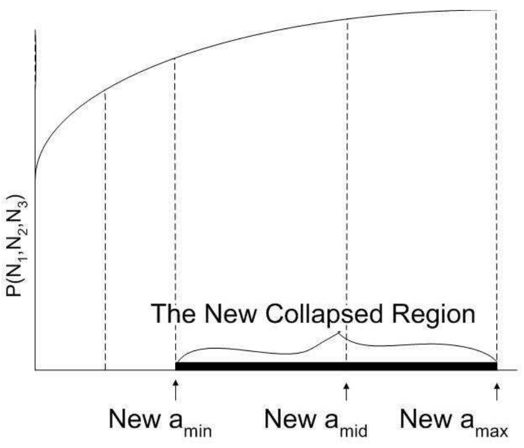


Figure B-7: The New a_{min}, a_{mid}, and a_{max} for the Monotonically Increasing Function

(b) If the P Function is Monotonically Decreasing

It is,

$$P_{amin} \geq P_{amid} \geq P_{amax}$$

Figure B-8 shows a monotonically decreasing P function, with all the values of P_{amin}, $P_{aminmid}$, P_{amid},$P_{amidmax}$, and P_{amax}.

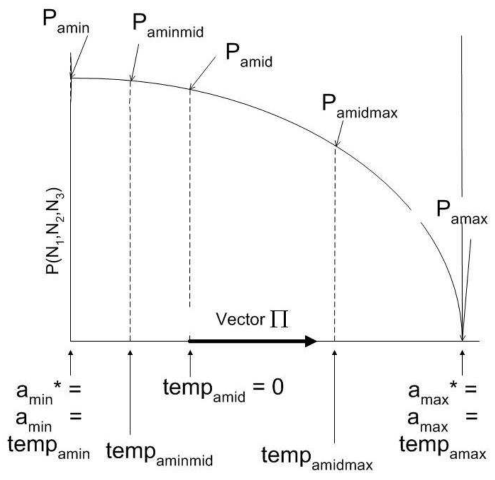


Figure B-8: A Monotonically Decreasing Production Rate Function

In the case of a monotonically decreasing P function, we only need to search the region between $temp_{amin}$ and $temp_{amid}$ because this is where the scalar a, associated with the optimal P, will be located. Therefore, the new a_{min}, a_{max}, a_{mid}, and P_{avg} are calculated as follows:

$$
\begin{aligned}
a_{min} &= temp_{amin} \\
a_{max} &= temp_{amid} \\
a_{mid} &= \frac{temp_{amin} + temp_{amid}}{2}
\end{aligned}
$$

$$P_{avg} = P_{aminmid}$$

Figure B-9 shows the new collapsed region, with the locations of the new a_{min}, a_{mid}, and a_{max}.

(c) If the P Function is Strictly Concave

That is,

$$P_{amin} \leq P_{amid}$$

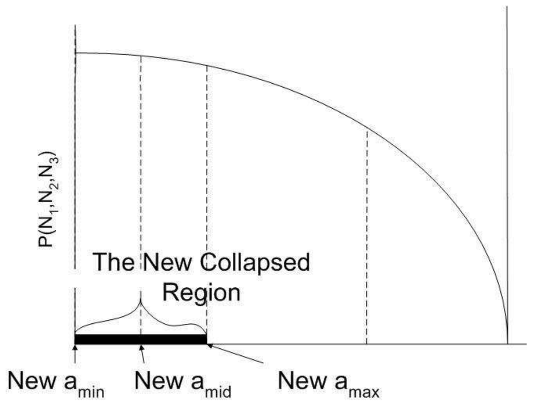


Figure B-9: The New a_{min}, a_{mid}, and a_{max} for the Monotonically Decreasing Function

and

$$P_{amax} \leq P_{amid}$$

An example of a strictly concave function is shown in Figure B-10.

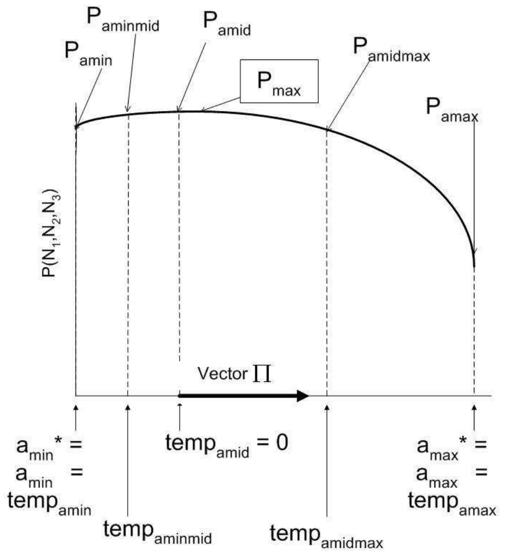


Figure B-10: A Strictly Concave Production Rate Function

A strictly concave P function can assume three shapes:

i. The Function between $temp_{aminmid}$ and $temp_{amidmax}$ (Inner Function) is also Strictly Concave

   It is,

$$(P_{aminmid} \leq P_{amid})$$

   and

$$(P_{amid} \geq P_{amidmax})$$

   If this is the case, again modify a_{min}, a_{max}, and a_{mid} so that the new region will contain the optimal production rate value.

$$\begin{aligned} a_{min} &= temp_{aminmid} \\ a_{max} &= temp_{amidmax} \\ a_{mid} &= temp_{amid} \\ P_{avg} &= P_{amid} \end{aligned}$$

   Figure B-11 follows in this category. That is $(P_{aminmid} \leq P_{amid})$ and $(P_{amid} \geq P_{amidmax})$.

   Figure B-12 shows the new collapsed region.

ii. The Function between $temp_{aminmid}$ and $temp_{amidmax}$ (Inner Function) is Monotonically Decreasing

   That is,

$$P_{aminmid} \geq P_{amid} \geq P_{amidmax}$$

   In this case, the region that contains P_{max} will be the region between $temp_{amin}$ and $temp_{amid}$. Therefore, the new a_{min}, a_{max}, and a_{mid} are

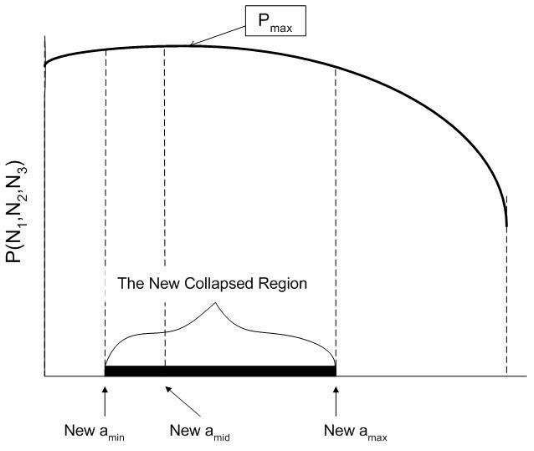


Figure B-11: The Inner Production Rate Function is also a Strictly Concave Function

set such that:

$$\begin{aligned} a_{min} &= temp_{amin} \\ a_{max} &= temp_{amid} \\ a_{mid} &= \frac{temp_{amin} + temp_{amid}}{2} \\ P_{avg} &= P_{aminmid} \end{aligned}$$

iii. The Function between $temp_{aminmid}$ and $temp_{amidmax}$ (Inner Function) is Monotonically Increasing

It is,

$$P_{aminmid} \leq P_{amid} \leq P_{amidmax}$$

In this case, the region that contains P_{max} will be the region between $temp_{amid}$ and $temp_{amax}$. Therefore, the new a_{min}, a_{max}, and a_{mid} are set such that:

$$\begin{aligned} a_{min} &= temp_{amid} \\ a_{max} &= temp_{amax} \end{aligned}$$

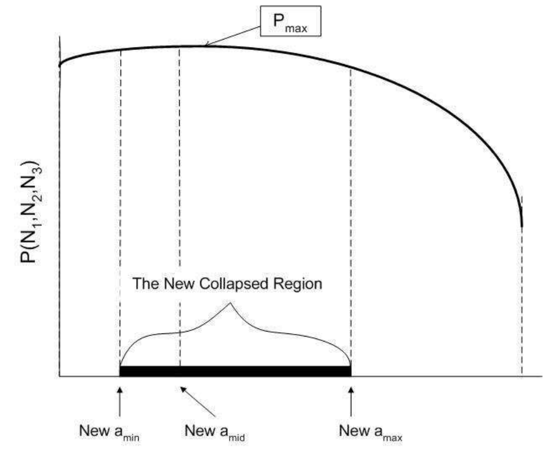


Figure B-12: The New Collapsed Region for the Strictly Concave Inner Production Rate Function: New Collapsed Region

$$\begin{aligned} a_{mid} &= \frac{temp_{amid} + temp_{amax}}{2} \\ P_{avg} &= P_{amidmax} \end{aligned}$$

At this step, the new a_{min} and a_{max} have been found.

6. Checking the termination criterion

   To determine whether we need to terminate the algorithm, we check whether a_{min} is close to a_{max}. If they are close to each other, stop the algorithm. If not, repeat the algorithm until a_{min} and a_{max} are close to each other.

Now we will demonstrate how to reach the P_{max} for function in Figure B-10. Figure B-10 is a strictly concave function. Its inner function is also a strictly concave function. Therefore, we collapse the region by letting $a_{min} = temp_{aminmid}, a_{max} = temp_{amidmax}$, and $a_{mid} = temp_{amid}$.

The new collapsed region is shown in Figure B-13.

Afterwards, we divide the new region in four areas, separated by $temp_{amin}$, $temp_{amidmid}$, $temp_{amid}$, $temp_{amidmax}$, and $temp_{amax}$. The P function on the new region is also a strictly concave function. We again collapse this new region according to the rules developed for strictly concave functions.

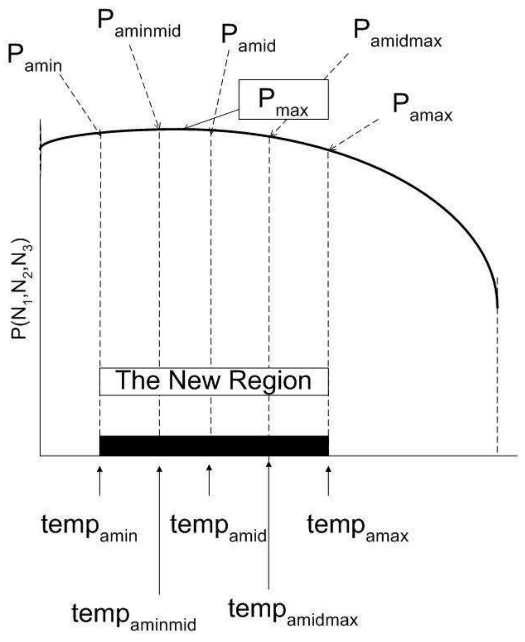


Figure B-13: New Collapsed Region

Figure B-14 shows the new collapsed region.

We continue analyzing the P function of the new region and collapsing the region until the termination criterion, which is when a_{min} is close to a_{max}, is satisfied.

Figure B-15 shows another new collapsed region.

At this point, the termination criteria very likely gets satisfied. Therefore, the algorithm terminates. The scalar a, which is equal to the last $temp_{amid}$, has been found.

Figure B-16 shows the block diagram of the algorithm to find the scalar a.

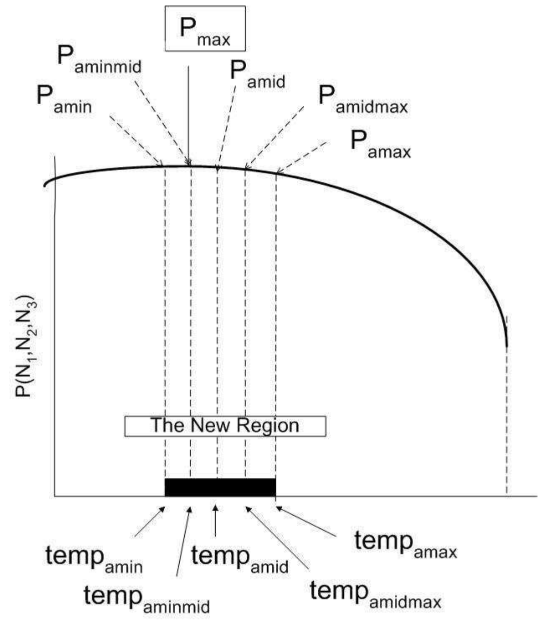


Figure B-14: Another New Collapsed Region

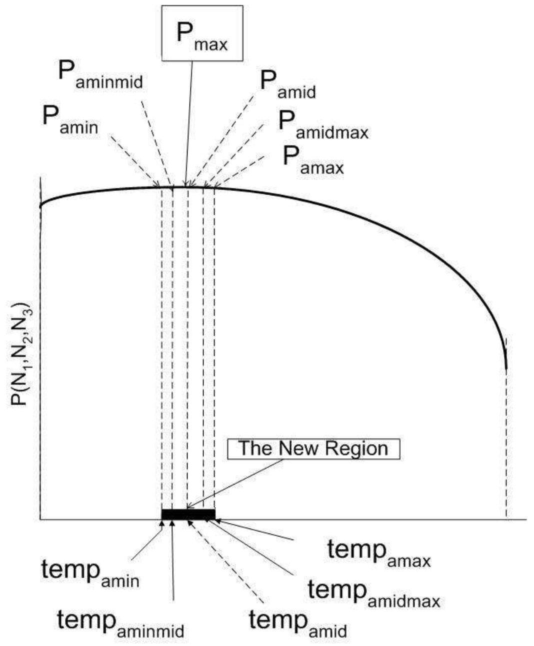
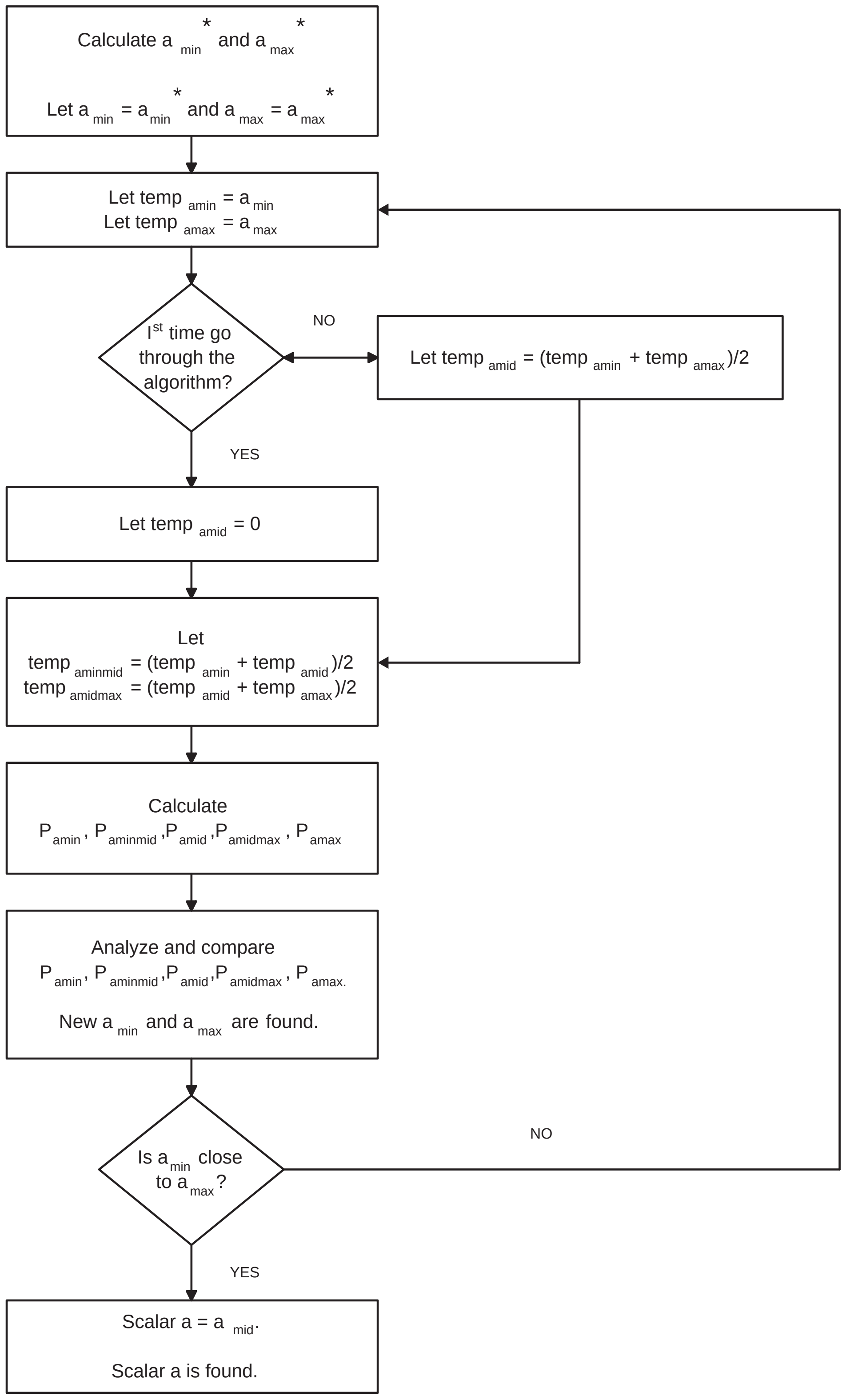


Figure B-15: The Final Collapsed Region



Calculate a min * and a max *
Let a min = a min * and a max = a max *
Let temp amin = a min
Let temp amax = a max
Ist time go through the algorithm?
NO
Let temp amid = (temp amin + temp amax)/2
YES
Let temp amid = 0
Let
temp aminmid = (temp amin + temp amid)/2
temp amidmax = (temp amid + temp amax)/2
Calculate
P amin , P aminmid ,P amid ,P amidmax , P amax
Analyze and compare
P amin, P aminmid ,P amid ,P amidmax , P amax.
New a min and a max are found.
Is a min close to a max ?
NO
YES
Scalar a = a mid.
Scalar a is found.

Figure B-16: Block Diagram of the Linear Search for Scalar a Algorithm

Bibliography

[Adan89] I. Adan and J. Van Der Wal (1989), "Monotonicity of the Throughput in Single Server Production and Assembly Networks with Respect to the Buffer Sizes," *Queuing Networks with Blocking*, H. G. Perros and T. Altiok, Editors, Elsevier Science Publishers, pp. 345-356.

[Binary04] "Binary Search." Internet Source. Http://www.bitesizeinc.net/power.programming.binary.search.html. 7 Jan. 2004.

[Burman95] M. Burman (1995), "New Results in Flow Line Analysis", Ph.D. Thesis, MIT, Cambridge MA, 1995.

[Chow87] We-Min Chow (1987), "Buffer Capacity Analysis for Sequential Production Lines with Variables Processing Times," *International Journal of Production Research*, Vol. 25, No.8, pp. 1183-1196.

[Schor00] S. B. Gershwin and J. E. Schor (2000), "Efficient Algorithms for Buffer Space Allocation," *Annals of Operations Research*, Volume 93, pp. 117-144.

[Gershwin87] S. B. Gerhswin (1987), "An Efficient Decomposition Method for the Approximate Evaluation of Tandem Queues with Finite Storage Space and Blocking," *Operations Research*, March-April, pp. 291-305.

[Gershwin94] S. B. Gershwin (1994), *Manufacturing System Engineering*, Prentice-Hall, 1994

[Ho79] Y. C. Ho, M. A. Eyler, and T. T. Chien (1979), "A Gradient Technique for General Buffer Storage Design in a Production Line," *International Journal of Production Research*, Vol. 17, No. 6, pp. 557-580.

[Jacobs96] D. Jacobs, C.-T. Kuo, J.-T. Lim and S. M. Meerkov (1996), "Improvability of Production Lines: Theory and Applications", *Lecture Notes in Control and Information Sciences*, vol. 214, pp. 121-141.

[Kinsey02] M. Kinsey and B. Ganesan (2002), "Investigation into the Relationship between Optimum Buffer Allocation and Buffer Level Variance in a Finite Buffer Line."

[Meester90] L. E. Meester and J. G. Shanthikumar (1990), "Concavity of the Throughput of Tandem Queuing Systems with Finite Buffer Storage Space," *Advances in Applied Probability*, Vol. 22, pp. 764-767.

[Park93] T. Park (1993), "A Two-Phase Heuristic Algorithm for Determining Buffer Sizes of Production Lines," *International Journal of Production Research*, Vol. 31, No. 3, pp. 613-631.

[Spinellis00] D. D. Spinellis and C. T. Papadopoulos (2000), "Stochastic Algorithms for Buffer Allocation in Reliable Production Lines, " *Mathematical Problems in Engineering*, 5: 441-458.