DEVELOPING METHODS TO SOLVE THE WORKFORCE ASSIGNMENT PROBLEM CONSIDERING WORKER HETEROGENEITY AND LEARNING AND FORGETTING

by

Natasa S. Vidic

B.S., University of Belgrade, Serbia, 1987

M.S., University of Delaware, U.S.A., 1992

Submitted to the Graduate Faculty of

Swanson School of Engineering in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

University of Pittsburgh

2008

UNIVERSITY OF PITTSBURGH

SWANSON SCHOOL OF ENGINEERING

This dissertation was presented

by

Natasa S. Vidic

It was defended on

March 27, 2008

and approved by

Matthew Bailey, Assistant Professor, Management Department, Bucknell University

Brady Hunsaker, Assistant Professor, Industrial Engineering Department

Jayant Rajgopal, Associate Professor, Industrial Engineering Department

David A. Nembhard, Industrial and Manufacturing Engineering, Penn State University

Dissertation Director: Bryan A. Norman, Associate Professor, Industrial Engineering Department

DEVELOPING METHODS TO SOLVE THE WORKFORCE ASSIGNMENT PROBLEM CONSIDERING WORKER HETEROGENEITY AND LEARNING AND FORGETTING

Natasa S. Vidic, PhD

University of Pittsburgh, 2008

In this research we studied how the assignment of a fully cross-trained workforce organized on a serial production line affects throughput. We focused on two serial production environments: dynamic worksharing on a production line, similar to bucket brigade systems and a fixed assignment serial-production line where workers work on a specific task during a given time period.

For the dynamic assignment environment we concentrated on the impact of different assignment approaches and policies on the overall system performance. First, we studied two worker two station lines when incomplete dominance is possible as well as the effects of duplicating tooling at these lines. One focus of this research was to optimally solve the dynamic worksharing assignment problem and determine exact percentages of work performed by each worker under the assumptions presented. We developed a mixed integer programming formulation for n workers and m stations that models one-cycle balanced line behavior where workers exchange parts at exactly one position. This formulation is extended to incorporate multiple production lines. We also developed a two-cycle formulation that models a condition when workers exchange parts at exactly two positions in a periodic manner. We also determined

throughput levels when workers productivity changes over time due to workers' learning and forgetting characteristics.

A fixed worker assignment system considers a serial production setting in which work is passed from station to station with intermediate buffers between stations. We considered two models. The first model assumed that workers perform tasks based on their steady-state productivity rate. The second model assumed that workers' productivity rates vary based on their learning and forgetting characteristics. Heuristic methods were developed and implemented to solve these two models and to determine optimal throughput levels and optimal worker assignments. We were also able to demonstrate the importance of introducing learning and forgetting into these types of worker assignment problems. A final focus of this research was the comparison of the dynamic worksharing and fixed worker assignment environments.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGMENTS

I would like to thank Dr. Bryan Norman for his continuous support and steady guidance throughout my graduate studies. He was always available to offer constructive criticism and creative ideas of crucial significance to my work and was always willing to unselfishly donate his time to our frequent discussions. I am grateful for his patience and understanding despite many distractions and constraints that I had to deal with during the last 5 years. And above all I am thankful to Dr. Norman for always being there for me and for being a great friend.

I am also grateful to Drs. Bailey, Hunsaker, Nembhard and Rajgopal for serving on my dissertation committee and for helping me focus my efforts towards a better final product. I have enjoyed immensely learning from all of them. Their valuable insight and the different perspectives they brought to bear in this work is greatly appreciated.

I am grateful to my parents and my sisters who were always there to offer encouragement, love and support that helped me greatly to overcome all the challenges throughout my life.

Finally, I would like to thank my husband and my children who showed enormous patience, understanding and support and who are always my inspiration and my motivation. This dissertation is dedicated to Nikola, Sofia and Dana.

NOMENCLATURE

I	The set of workers $i = 1,2,...n$.
J	The set of tasks $j = 1,2,...m$.
k_{ij}	Steady-state production rate for worker i doing task j.
$Output$	Total throughput. $Output \geq 0$.
x_{ij}	Variable indicating the fraction of time worker i spends doing task j. $x_{ij} \geq 0$.
w_j	The output from station/task j. $w_j \geq 0$.
z_{ij}	Binary variable indicating whether worker i does task j.
$idle_i$	Variable indicating the amount of time worker i is idle. $idle_i \geq 0$.
L	The set of lines $l = 1,2,...o$.
S_p	The sets of tasks for each line $p = 1,2,...,l$.
v	Number of workers at each line (given constant).
$Output_o$	Total throughput from line o. $Output_o \geq 0$.
s_{io}	Binary variable indicating whether worker i is assigned to line o.
$idle'_i$	Variable indicating the amount of time worker i is idle during the first cycle,
$idle''_i$	Variable indicating the amount of time worker i is idle during the second cycle.
x'_{ij}	Variable indicating the amount of time worker i spends doing task j during the first cycle.

x''_{ij} Variable indicating the amount of time worker i spends doing task j during the second cycle.

w'_j The output from station (task) j during the first cycle.

w''_j The output from station (task) j during the second cycle.

z'_{ij} Binary variable indicating whether worker i does task j during the first cycle.

z''_{ij} Binary variable indicating whether worker i does task j during the second cycle.

y_u Measure of the productivity rate corresponding to u units of cumulative work. $y_j \geq 0$.

$u_{i,j}$ Cumulative number of units produced by worker i at task (station) j. $u_{i,j} \geq 0$.

$p_{i,j}$ Imputed prior expertise of worker i doing task j,

$r_{i,j}$ Learning parameter of worker i doing task j, $p_{i,j} + r_{i,j} > 0$,

$\alpha_{i,j}$ Forgetting rate of worker i doing task j,

$R_{u_{i,j}}$ Recency of individual experience based on units (ratio of average elapsed time to most recent unit time),

$st_{u_{i,j}}$ Starting time of unit $u_{i,j}$,

$st_{0_{i,j}}$ Starting time of the first unit of worker i doing task j,

t_0 Starting time for the first period of work.

T The set of time periods $t = 1,2,...t$.

q_{ijt} Binary variable indicating whether worker i does task j during time t.

O_{ijt} The output from worker i performing task j during time t. $O_{i,j,t} \geq 0$.

B_{jt} Buffer inventory level at task j at the end of period t. $B_{j,t} \geq 0$.

BI_j Beginning inventory for task j.

pr_{ijt} Productivity if worker i does task j during time t. $pr_{i,j,t} \geq 0$.

$R_{i,j,t}$ Recency of individual experience based on time periods (ratio of average elapsed time to most recent period time).

M Very large number.

1.0 INTRODUCTION

The manufacturing industry today faces a wide variety of challenges. Manufacturing companies around the world are attempting to improve their productivity, reduce manufacturing process complexity, and gain better production insight in order to stay on top of their industry. In today's manufacturing market achieving high production efficiency is extremely important. Traditional production lines such as classical assembly lines are often inflexible and can be inefficient because it is difficult to balance the workload among the workers. In order to improve flexibility, maximally use resources, and maximize throughput many companies are applying worksharing concepts. Worksharing involves more than one worker doing a task rather than having tasks done by only one worker. A key prerequisite for worksharing is cross-training so that employees can perform more than one operation. Cross trained workers represent flexible capacity and can be shifted dynamically to where they are needed (Hopp and Van Oyen 2004). In addition to reducing idle time and buffer sizes, cross-training workers is known to have other benefits, such as job enhancement, flexibility and reduced risk of worker injuries (Chen and Askin 2006). McClain (2000) points out that in some settings worksharing allows the line to be balanced by alternating which worker does a particular task, effectively splitting the shared tasks, so that the line balances itself.

Today's manufacturing market is changing fast and is very competitive. When focusing on achieving high efficiency we also have to include the fact that today products have shorter life cycles and there is higher service diversity. Nembhard and Norman (2002) point out that many organizations face bigger challenges due to faster product changes, service diversity and intense competition. Organizations and service industries are facing new market challenges because of the accelerated rate of process innovation and as a result, products have shorter life cycles. Work activities need to be restructured and reorganized more often and workers need to master new tasks more frequently. A result of these frequent process and product changes is that workers learn new tasks very often. The amount of time the workforce spends on the steep part of the learning curve increases as new products are introduced (Uzumeri and Nembhard 1998). However, research in the area of worker-task assignment generally assumes steady-state productivity and the assignment of workers to tasks based on workers' individual learning and forgetting characteristics has received very little attention in the literature. The necessity of incorporating worker learning and relearning behavior has been addressed by several researchers (Wisner and Pearson 1993) and it is misleading to assume steady-state productivity rates when workers master new tasks very often (Shafer et al. 2001).

In this research we consider how the assignment of a fully cross-trained workforce deployed on a serial production line affects throughput. Our overall aim is to develop methods to solve the worker assignment problem on a serial production line where serial operations vary in complexity and workforce is heterogeneous. We concentrate on two serial production environments: dynamic worksharing on a production line, similar to bucket brigades systems and a fixed assignment serial-production line where workers work on a specific task and there is no sharing during a given time period.

Dynamic assignment such as a bucket brigade system considers a serial production setting where workers walk to adjacent stations and carry the work towards the last station. This research considers the case where there are *n* workers and *m* stations and there are no buffers between the stations. Manufacturing systems organized in this way are one example of a "pull system". Generally, in "pull systems" the work-in-process (WIP) is controlled and is minimal. An example production line is presented in Figure 1.

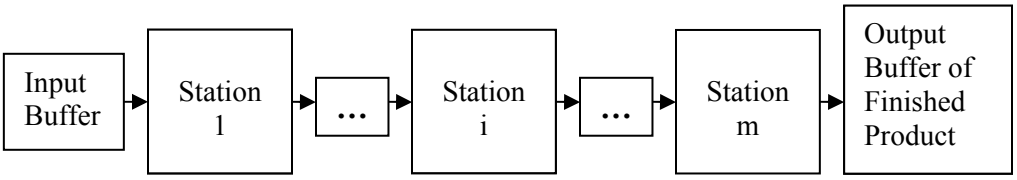


Figure 1: Serial Production Line: Dynamic Assignment Environment

This type of production environment was introduced by Toyota and is called the "Toyota Sewn Products Management System" (TSS). It is used in many industries such as apparel, handbags, shoes, suitcases, furniture, etc. The bucket brigade system differs from a TSS system in that there are no pre-assigned worker zones to limit the movement of workers. Bartholdi and Eisenstein (1996) describe a bucket brigade system and present a sufficient condition for such lines to be self-balancing. Under a bucket brigade system, each worker starts working on a part and processes it at each station until the part is either finished or the next worker in the order takes over the part. The order of the workers is preserved at all times. More detailed analysis of workers' movement is presented in Chapter Three. Dynamic worksharing is analyzed and tested under the assumption that the workers' productivity rate is at steady-state production level. Two

mixed integer formulations are developed. We also simulate throughput levels when workers productivity changes over time due to workers' learning and forgetting characteristics.

A fixed worker assignment system is also considered for a serial production setting, in which work is passed from station to station, with intermediate buffers between stations. There are *n* workers, *m* stations and *p* time periods. Workers can be assigned to any station and the size of the intermediate buffers can be either unrestricted or restricted. This production environment with three stations is presented in Figure 2.

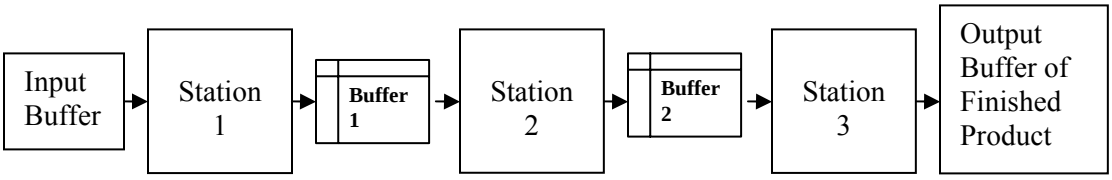


Figure 2: Serial Production Line: Fixed Assignment Environment

For a fixed assignment environment we analyze two types of models. The first model assumes that workers perform tasks based on their steady-state productivity rate. The second model assumes that workers' productivity rates vary based on their learning and forgetting characteristics.

For both production environments we look at workers performing at steady-state production levels as well as production levels when learning and forgetting is present. The learning model used in our research was introduced by Mazur and Hastie (1978) and was modified to include the effects of forgetting by Nembhard and Uzumeri (2001). The workers'

productivity changes over time and is dependent on the number of times the workers were assigned to each task, and the recency of their experience.

1.1 OVERVIEW OF DISSERTATION

The organization and areas of concentration of this dissertation are presented in Figure 3.

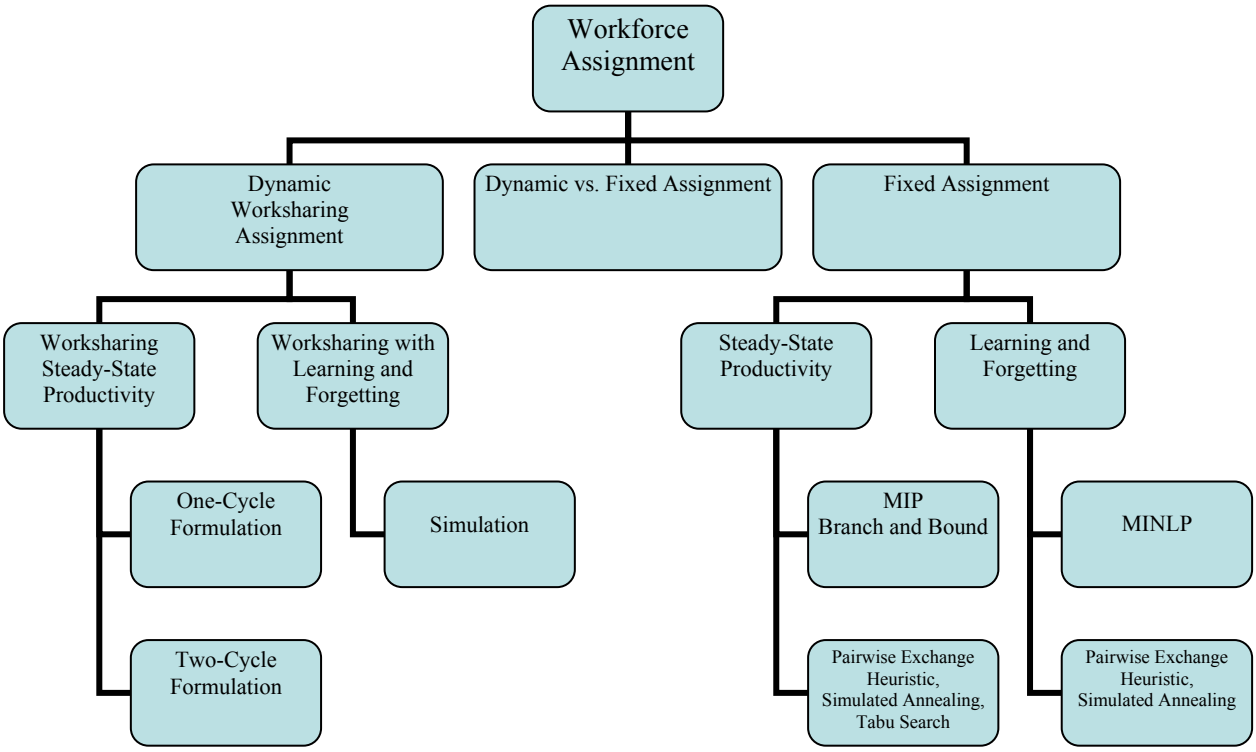


Figure 3: Workforce assignment

In Chapter Two, a review of the relevant literature is presented. We look at recent literature covering workforce flexibility, cross-training, workgroup selection, worker assignment, worksharing, bucket brigade systems and learning and forgetting. In Chapter Three we discuss dynamic worksharing assignment environments and present the analysis of a two worker two station production line. We look at optimal levels of worksharing and present a detailed case analysis based on workers' steady-state production rates. This chapter also includes the analysis of a two worker two station line with duplicate tooling on either one or both stations.

A one-cycle formulation for the dynamic assignment problem is presented in Chapter Four including the model formulation, numerical results and conclusions. For a one-cycle formulation workers exchange parts at exactly one location and they repetitively perform the same amount of work on each part. The extended formulation which considers several production lines coupled together either as an independent or linked system is also presented in Chapter Four. The two-cycle formulation for a dynamic assignment environment is presented in Chapter Five. For a two-cycle formulation workers exchange parts at two locations and they perform repetitive two-cycle work, so the amount of work performed on every other part is identical. Chapter Six discusses dynamic assignment when worker learning and forgetting is introduced, thus, workers do not perform at steady state levels. A learning and forgetting model is presented. We look at dynamic worksharing systems with n workers and m stations and compare situations when workers perform at their steady state levels versus when learning and forgetting is included in the analysis.

We analyze a fixed assignment environment when workers work at steady state production levels in Chapter Seven. A production line with intermediate buffers is presented as well as a mixed integer model formulation. Smaller size instances for the fixed assignment MIP

are solved by the branch-and-bound technique employed by CPLEX. Due to the complexity of the problem, meta-heuristics including Simulated Annealing are implemented when solving larger instances. The MIP formulation is extended to include the effects of learning and forgetting and the resulting MINLP formulation is also presented in Chapter Seven. The fixed assignment MINLP formulation is solved by Simulated Annealing.

A comparison of dynamic versus fixed assignment environments is presented in Chapter Eight. We compare final outputs for the fixed assignment serial production line where workers are rotated during fixed time intervals and a dynamic worksharing environment given the same number of workers, number of stations and total production horizon. We examined different WIP scenarios and present observations and conclusions. Directions for future research and conclusions as well as contributions of this research are summarized in Chapter 10. Different algorithms and codes are given in the Appendices.

1.2 CONTRIBUTIONS

We first list the contributions relevant to the dynamic worksharing assignment.

- We were able to define possible benefits achieved from worksharing and the impact of different assignment approaches and policies on system performance. Our goal was to make recommendations according to our results with an objective of achieving maximum efficiency and throughput. Another important focus of this research was to optimally solve the dynamic worksharing assignment problem and determine exact portions of work performed by each worker under the assumptions presented.

- We first analyzed two worker two station lines when incomplete dominance is possible. We also investigated the effects of duplicate tooling at these lines.
- We developed a mixed integer programming formulation for *n* workers and *m* stations that models one-cycle balanced line performance where workers exchange parts at exactly one position. We are able to obtain optimal positioning of the workers and the optimal amount of work performed by each worker.
- We also extended the one-cycle formulation to incorporate multiple production lines.
- We also developed a two-cycle formulation that models a condition when workers exchange parts at exactly two positions in a periodic manner. We compare the optimal throughput obtained from the one-cycle and two-cycle formulations for lines with two or more workers.

For the fixed assignment environment we considered two formulations:

  - the MIP formulation with constant productivity rates, and
  - the MINLP formulation with learning and forgetting effects.
- We implemented heuristic methods to solve these problems and to determine the optimal throughput levels as well as the optimal assignment of workers.
- We were also able to show the importance of introducing learning and forgetting into these types of worker assignment problems. Different scenarios were analyzed and tested and it is shown that the total throughput differs considerably and is misleading when workers are assigned based on their steady-state productivity rate.

Another important contribution of this research is the comparison of these environments.

- We were able to determine and define advantages and disadvantages of these two assignment methods given the same production conditions such as number of workers, number of stations, workers' production rates and the length of the production horizon.

2.0 LITERATURE REVIEW

Our research focuses on dynamic worksharing on a production line, similar to bucket brigade systems, and a fixed assignment serial-production line where workers work on a specific task and there is no worksharing during a given time period. We consider a fully cross-trained heterogeneous workforce. Numerous papers have been written on the workforce assignment problem, including some that consider cross trained workers or multi-functional workers and worksharing. In this section we review those papers most related to this research.

Our work related to the fixed assignment environment is an extension of the work done by Nembhard and Norman (2002) and Leopairote (2004). Regarding the dynamic worksharing environment, our research concentrates on a production line where workers are dynamically assigned, similar to bucket brigades, and tasks are discrete. Most of the bucket brigade literature assumes that work is evenly spread along the line, or that tasks are continuous, and workers' speeds are constant along the line. The most relevant work regarding bucket brigade applications with discrete tasks is done by Lim and Yang (2006). Also, other relevant work was done by McClain et al. (2000).

We study production lines where workers perform at steady-state production rates as well as production lines where workers productivity changes due to the presence of learning and forgetting. Thus, we will also discuss learning and forgetting models in more detail.

2.1 WORKSHARING AND BUCKET BRIGADE SYSTEMS

Two types of implementation of worksharing are frequently discussed in the literature: dynamic assembly line balancing and moving worker modules (Chen and Askin 2006). Dynamic task assignment in the traditional serial line model with partially cross-trained workers is addressed by Askin and Chen (2006) with the objective to maximize throughput. In the implementation of dynamic line balancing the identity of shared tasks has to be determined as well as the operational task rules (Anuar and Bukchin, 2006). Chen and Askin (2006) analyzed the tradeoff between the cost of work-in-process inventory and cross-training in dynamic balancing systems, and also tried to determine the best operating policies for shared tasks Overall, they concluded that worksharing improved output in the analyzed environments by 5.6% over static balanced assignments

Many researchers have studied worksharing and dynamic line balancing when working zones overlap. Among them are Ostolaza et al. (1990), McClain et al. (1992), Schultz et al. (1998) and McClain et al. (2000). McClain et al. (1992) concluded that dynamic line balancing can increase efficiency even when inventory buffers are absent. McClain et al. (2000) analyzed worksharing in a variety of situations including different worker to machine ratios, unequal workers, uncertain processing times, and handoffs with and without preemption. They concluded that worker sequencing is quite important, as slowest to fastest performs well in some situations and poorly in other. Their hypotheses consider work zones, inventory buffers and that there is complete dominance regarding workers' velocities.

Hopp et al. (2004) studied two cross-training strategies for serial production systems with flexible servers. They stated that the primary benefits of workforce agility in this environment are capacity balancing, and variability buffering, which provides a solution for worker idleness caused by variability in processing times. The number of workers is equal to the number of stations. Hopp and Van Oyen (2004) outline approaches for accessing and classifying manufacturing and service operations in terms of their suitability for use of cross-trained workers. They define production agility as the ability to achieve heightened levels of efficiency and flexibility while meeting objectives for quality and customer service.

Sennott, Van Oyen and Iravani (2006) modeled and analyzed serial production lines with specialists at each station and a single, cross-trained floating worker who can work at any station. They formulated a Markov Decision Process which models K-station production lines. The model includes holding costs, set-up costs and set-up times at each station. They performed a numerical study for two and three station lines. They concluded that problems with both specialists and cross-trained workers are extremely difficult to optimize, and that the burden of maximizing performance falls on the worker with the greatest flexibility. Gel et al. (2002) formulated a model and a Markov Decision Process to explore the optimal control of systems with two workers and evaluate half-full buffer policies. Askin and Chen (2006) extended this work by looking at non preemptive tasks and trade-offs between the cost of cross training and the cost of work-in process inventory.

In a typical flow line, workers are assigned to fixed stations and the bottleneck station determines the production rate. In many types of flow lines there are fewer workers than stations and workers perform tasks at a portion of the line while maintaining their positioning. A different production line has been implemented in some manufacturing environments in which workers

carry a part and walk to adjacent stations and there are fewer workers than stations. This type of line is often referred to as a bucket brigade system.

Under a bucket brigade policy, each worker starts a job and processes it at each station until he or she is blocked by a downstream worker, or gets bumped by a downstream worker. There are applications of this concept in many industries and some of the current users of bucket brigades include: McGraw-Hill**,** Blockbuster Music, Coach Leatherware, Champion Products, Subway**,** Tug Manufacturing, United Technologies Automotive**,** Revco Drug Stores, Inc. (now CVS), Anderson Merchandise, Readers Digest**,** The Gap **(**Old Navy, The Gap, and Banana Republic), etc.

Bartholdi and Eisenstein (1996) analyzed systems using the bucket brigades policy under the assumption of deterministic processing times and non-identical workers, each with a different processing rate. The authors introduced the Normative Model of bucket brigades where the work content of the product is continuously and evenly distributed along the line. The authors provided the first detailed analysis of the dynamics of bucket brigade systems and proved the conditions under which a bucket brigade is self-balancing. They concluded that when workers are assigned on a production line from slowest to fastest, and move according to the traditional bucket brigade rules, the production rate converges to a value that is the maximum possible among all ways of organizing workers and stations. The weakness of the model is that it assumes constant worker velocities over time and that the work content is continuously distributed along the line. Another important result is that the authors state that balancing the line is always possible, or that there exist worker positions such that after completion of each item workers reset to exactly the same positions to begin work. The authors present the detailed discussion in

support of deterministic processing times. For more discussion on deterministic versus stochastic processing times see Bartholdi and Eisenstein (1996).

Bartholdi et al. (2001) addresses the case of stochastic processing times and proves a similarity between the deterministic and stochastic systems as the number of stations goes to infinity. Bartholdi et al. (2005) extended the Normative Model and assumed that walk-back times and hand-off times are significant. The adoption of bucket brigades resulted in a reduced number of tasks for each worker and higher overall productivity. Real case studies reported in the literature as well as a survey of papers considering bucket brigade topics are given in Bratcu and Dolgui (2005).

Bartholdi et al. (1999) studied the dynamics of two and three worker bucket brigade production lines and discussed the types of asymptotic behavior possible in practice as a function of the workers relative speeds, which is assumed to be constant over the entire line. For a two worker line the authors conclude that two modes of asymptotic behavior are possible. The first is that the movements of workers would spontaneously converge to a fixed point (one-cycle) balanced line with the optimal production rate. The second mode of behavior is that the movements of the workers would converge to a two-cycle balanced line with a suboptimal production rate. For three workers, the authors defined four regions of possible asymptotic behavior. Region 1 is defined as one cycle or convergence to a fixed point with the optimal production rate. Regions 2 and 3 cover situations when workers are not ordered form slowest to fastest (thus blocking is present) and the positions of the workers would alternate between two and three cycle positions with suboptimal production rates. The fourth region is defined as region k where the systems in this region can converge to a k cycle for some values of $k > 3$. The authors also suggest that it is better for management to sequence workers from slowest to fastest

and include very different workers (fast and slow) on the same team in order to achieve the maximum production rate. They also suggest that the greater the range in velocities on a team, the greater the rate of convergence.

Bucket brigades are most successful in applications where the skills required to perform the operations on the line are very similar, such as warehouse picking, fast food preparation, and textile sewing operations (Hopp and Van Oyen, 2004). A two worker bucket brigade was studied by Armbruster and Gel (2006) where one worker is faster than the other over some part of the production line and slower over another part of the line. They assumed deterministic processing times, continuous tasks and instantaneous walking speeds. The original no passing rule is modified as workers are allowed to pass each other. Two environments were analyzed: one with passing and one with blocking. The study presents conditions under which bucket brigades remain effective. The authors concluded that if the order of the workers is switched when one passes another the bucket brigade self-organizes itself. Their conclusion is that the system may not always balance itself on a fixed point but rather to two stable positions where workers exchange jobs. Workers would hand over jobs at exactly two fixed locations that they visit periodically.

Lim and Yang (2006) analyzed the throughput of a bucket brigade system with discrete work stations meaning that the work is not continuously and uniformly spread along the line. They considered fully and partially cross trained workers and assumed constant worker speeds on all tasks. The authors developed an analytical procedure to determine policies that maximize the throughput and studied complete dominance cases including the case when workers speeds are equal. The authors defined regions, depending on the work content of the stations and the

workers' speeds, where it is optimal to order workers either slowest to fastest or fastest to slowest in order to achieve optimal throughput.

In this research, the dynamic environment discussed in Chapter One is similar in many ways to a bucket brigade system with discrete tasks and deterministic production rates, however there are some key differences. The first difference is that we allow the last worker in the order to be idle. The second and key difference is that workers' production rates are worker/station dependent so complete dominance along the line may not be possible. That is, one worker may be faster at one station but slower at another station relative to second worker.

2.2 WORKFORCE FLEXIBILITY AND CROSS-TRAINING

Many researchers have discussed the benefits of cross-training and the impact of cross-training to the overall performance of a system. In our research, we focus on a fully cross-trained workforce and benefits of employing a flexible workforce on a serial production line. We now summarize the most relevant work. Norman et al. (2002) developed a mixed integer programming model to assign workers to tasks in manufacturing cells. The model considers both technical and human skills with the objective to maximize an organization's effectiveness. This model performs better than one that considers only technical skills. Slomp et al. (2005) studied the need for cross-training workers in a cellular manufacturing environment. They developed an integer programming model that can be used to select workers to be cross-trained for particular machines. Their study has shown that cross-training decisions in a cellular manufacturing environment should support the forming of effective 'chains' between workers and tasks in order to shift loads from a loaded worker to a less loaded worker. This model is helpful when

management is making decisions regarding the trade-off between training costs and workload balance among workers. In related worker assignment research, Molleman and Slomp (1999) developed a linear goal programming model subject to worker and skill requirements. In their extended study (2000), they formulated linear programming models and presented a hierarchical procedure for worker cross-training in order to reduce the workload of the bottleneck worker. Campbell and Diaby (2002) developed an assignment heuristic for allocating cross-trained workers to multiple departments at the beginning of a shift. Their formulation of the problem is a variant of the generalized assignment problem.

Cesani and Steudel (2005) studied labor flexibility in cellular manufacturing systems and specifically focused on the impact of different labor allocation strategies on system performance. They studied concepts such as workload balancing, workload sharing and the presence of bottleneck operations. They concluded that both factors, the level of shared workload and the workload assigned to individual operators, are very important when determining the performance of the system.

Quantitative studies in the area of workgroup selection in cellular manufacturing have been found in the literature including Norman et al. (2002), Askin and Huang (1997), Askin and Huang (2001) and Bhaskar and Srinivasan (1997). Askin and Huang (1997) compared two integer-programming models for assigning workers to cells and evaluated the training program for each worker. In their extended study (2001), they developed a multi-objective model to create work teams for cellular manufacturing systems.

Hopp and Van Oyen (2004) outline approaches for accessing and classifying manufacturing and service operations in terms of their suitability for use of cross-trained

workers. They define production agility as the ability to achieve heightened levels of efficiency and flexibility while meeting objectives for quality and customer service.

Nembhard and Prichanont (2007) investigated the impacts of workers' multifunctionality in a heterogeneous serial production system where workers differ based on their individual learning and forgetting characteristics. The authors conducted a simulation study and concluded that the managerial decisions on multifunctionality should be made in conjunction with worker rotation rate, degree of task similarity, bottleneck position, etc.

2.3 LEARNING AND FORGETTING

As we discussed in Chapter One, one of the goals of this research is to show the importance of incorporating workers' learning and forgetting characteristics when studying worker-task assignment problems. Research in this area mostly assumes steady-state productivity and the assignment of workers to stations based on workers' productivity that changes over time has not been discussed extensively.

The individual learning model used in our research was presented by Mazur and Hastie (1978). In a comparative study of learning models, Nembhard and Uzumeri (2000a) suggested that the hyperbolic learning model is useful when representing individual learning patterns. The initial learning model was modified to incorporate the effects of forgetting (Nembhard and Uzumeri, 2000b). The three parameter hyperbolic model was modified by including a measure that the authors termed recency of experiential learning. The authors (2000b), state that the recency measure represents the ratio of the average elapsed time to the elapsed time of the most recent unit produced. A comparative study of forgetting models is presented by Nembhard and

Osothsilp (2001), and the authors concluded that the recency model outperformed other models consistently in terms of all of the criteria presented in the paper. The hyperbolic-recency learning and forgetting model is presented in detail in Chapter Six.

Our research that concentrates on a fixed assignment environment is an extension of the work presented in Nembhard and Norman (2002). The authors developed a worker-task assignment model, where individual worker learning and forgetting is incorporated. The formulation was based on log-linear learning and forgetting where learning is dependent on a worker's initial productivity level and how often the task is performed. Forgetting is dependent on a worker's time absent from a task. An extensive list of related references relevant to our work also is presented in Nembhard and Norman (2002). Leopairote (2004) analyzed the same model where learning and forgetting was modeled using a hyperbolic-recency learning-forgetting model. The primary aim was to improve system efficiency and to develop policies for worker selection, assignment, and scheduling.

Jaber and Sikstrom (2004) compared three learning and forgetting models: the learn-forget-curve model, the recency model, and the power integration-diffusion model and discussed their differences and similarities. Their conclusions regarding tasks that are more motor than cognitive included that the recency model suggests that fast (slow) learners forget faster (slower), which is different from both the power-diffusion model as well as the learn-forget-curve model. In these cases the recency model is inconsistent with the other two models. The other two models suggest that as learning becomes slower forgetting becomes faster for motor tasks. They concluded that when a task is more cognitive than motor, or for a moderate learning scenario, all three models produced very similar predictions.

One of the basic assumptions of almost all of the previous bucket brigade work is that worker production rates are constant. There has been some consideration of learning and the use of bucket brigade systems in environments where worker production rates change over time. Munoz and Villalobos (2002) addressed the effectiveness of bucket brigades in the presence of high labor turnover and concluded that bucket brigade systems are highly effective in situations when frequent restitution of skills is present due to high labor turnover. They concluded from simulation studies that bucket brigades outperform alternative ways of organizing workers when there is high labor turnover.

Armbruster et al. (2007) also analyzed bucket brigades with workers learning. The authors primarily used an exponential learning model and considered both passing and blocking environments. Their concluded that self-balancing property of bucket brigades is very robust and defined conditions under which some managerial input is needed. They also concluded that a bucket brigade system with all workers learning will always lead to self-organized production.

We will study a dynamic assignment environment, discussed in Chapter One, where workers productivity changes over time due to workers' learning and forgetting characteristics and compare the system performance when workers produce at the steady-state rate. This analysis is presented in Chapter Six. We will also study a fixed assignment environment when workers learning and forgetting is present. This analysis is presented in Chapter Seven.

3.0 DYNAMIC WORKSHARING ASSIGNMENT

In this chapter we concentrate on the dynamic assignment environment discussed in Chapter One. We consider how the assignment of a fully cross-trained workforce deployed on a serial production line consisting of two workers and two stations affects throughput. We assume that worksharing is present if workers are assigned to the same station but not at the same time.

3.1 MODELING ASSUMPTIONS

We consider a serial-production line with two workers and two stations. Workers perform at steady-state production rates, processing times are deterministic and each part needs processing on the same sequence of stations (stations 1 and 2 in this case). There may be only two tasks to complete or multiple tasks may have been grouped into the two stations. Additionally, the work content at the stations need not be identical. Only one worker can work at a station at any point in time and only one part can be processed at the station. We assume that workers are fully cross-trained and that there are no restrictions on their assignments, thus any worker can potentially work at any station. Throughout this research, we assume that our models also apply to cases where the workforce is partially cross-trained. If a worker is not trained for a specific task a very low production rate would be assumed for that specific worker task combination. Tasks at stations have different complexity levels and worker speeds are not necessarily uniformly

dominated (meaning that worker 1 (W1) can be faster than worker 2 (W2) on station 1 (S1), but slower on station 2 (S2)). No buffer inventory is permitted between the stations. The order of the workers is preserved and no passing is allowed. We assume that preemption is allowed, thus workers can interrupt each other at any time. The production rate k_{ij} is defined as the number of parts produced by worker i at station j in a given period of time (e.g., one hour), and $k_{ij} \geq 0$.

In order to determine the best assignment policy we assign workers in several ways: 1) no-sharing, 2) sharing is allowed but the last worker is always busy (never idle), and 3) sharing is allowed and the last worker is allowed to be idle. For no-sharing assignments, workers perform a task at their assigned station, and when finished either wait for a downstream worker to finish his/her task and take over the part, or if a downstream worker is already finished, he/she takes the part. For sharing options workers can go forward or backwards when finished at their assigned station. Both the forward and backward movements are adopted from the bucket brigade rules introduced by Bartholdi and Eisenstein (1996). Workers walk forward until blocked by a downstream worker or a part is processed at the last station and finished. Also, workers walk backward when a part is finished or they were interrupted by a downstream worker, and take work from a previous worker in the order. During this phase each worker interrupts his/her predecessor to take over his/her work, and the first worker in the order starts a new part once the first station is empty. This assumption differs from previous bucket brigade literature where the first worker in the order starts the work at the same time as the line resets (workers change positions). The first worker in this case needs to wait for the station to be empty, as only one worker can work at the station at one time. Our assumption is that both handoff times and walking times (forward or backwards) are significantly smaller than

production times, so we can safely assume that these occur instantaneously. This also implies that the positions of workers on the line change instantaneously when a part is completed.

Notice that because the tasks have different complexity levels and the workforce is heterogeneous, ordering workers from slowest to fastest may not be possible. This implies that a worker can be blocked if he/she arrives at a station that is busy. This assumption is the same as Lim and Yang (2006) and different than Armbruster/Gel's (2006) definition of blocking. Armbruster and Gel (2006) assume that a blocked worker is not idle, but rather he/she moves at the speed of the worker in front of him/her. This is not feasible under our assumptions as only one worker can be at a station at any point in time.

3.2 WORKER SCHEDULING OF TWO WORKER TWO STATION LINE WITH WORKSHARING

In this section, our objective is to study a line with two workers and two stations and maximize the total throughput of the line. We define the objective function as minimizing the total time per part while satisfying assignment constraints. We would like to determine the optimal output rate considering the workers' station dependent production rates. Also, we want to determine the optimal amount of time that worker *i* spends working at station *j*. Thus, one of the main questions that we attempt to answer is what is the optimal worksharing in the long run (percentage of each task done by each worker)? Also, given the workers' production rates which assignment options are the best?

First we will analyze two worker and two station lines where there is only one set of tools at each station. Based on our observations we then consider the possible benefits of duplicating one or both stations.

Recall that we define the objective as minimizing the total time required per part produced. Time t_{ij} is defined as the time required for worker i to finish work at station j, and is equal to $60/k_{ij}$ where 60 is the length of one hour. We define x_{ij} as the portion of time that worker i spends working at station j during the production horizon. The production horizon is defined as the total production time including all idle times. If we assume that workers' production rates are not constant over the entire line then we have a different production rate per worker per station. Let a, b, c, d represent the four possible k_{ij} values for a two worker two station line. Assuming that $a \leq b \leq c \leq d$, then there are 24 (or 4!) possible combinations of relative worker speeds based on the different workers' production rates. We analyze the 12 cases presented in Table 1.

Table 1: Production rates for two worker two station line

Case 1	*S1*	*S2*	*Case 2*	*S1*	*S2*	*Case 3*	*S1*	*S2*	*Case 4*	*S1*	*S2*
W1	*a*	*b*	*W1*	*b*	*a*	*W1*	*a*	*b*	*W1*	*b*	*a*
W2	*c*	*d*	*W2*	*c*	*d*	*W2*	*d*	*c*	*W2*	*d*	*c*
Case 5	*S1*	*S2*	*Case 6*	*S1*	*S2*	*Case 7*	*S1*	*S2*	*Case 8*	*S1*	*S2*
W1	*a*	*d*	*W1*	*d*	*a*	*W1*	*a*	*d*	*W1*	*d*	*a*
W2	*b*	*c*	*W2*	*b*	*c*	*W2*	*c*	*b*	*W2*	*c*	*b*
Case 9	*S1*	*S2*	*Case 10*	*S1*	*S2*	*Case 11*	*S1*	*S2*	*Case 12*	*S1*	*S2*
W1	*a*	*c*	*W1*	*c*	*a*	*W1*	*a*	*c*	*W1*	*c*	*a*
W2	*b*	*d*	*W2*	*b*	*d*	*W2*	*d*	*b*	*W2*	*d*	*b*

The additional 12 cases are not considered because due to symmetry they are equivalent to the 12 cases analyzed. For example, if we analyze Case 1 with production rates of *a* and *b* for W1 on S1 and S2 respectively, and *c* and *d* for W2 on S1 and S2 respectively, the analysis of this case is identical to that of W1 having production rates of *c* and *d* on S1 and S2, respectively, and W2 having production rates *a* and *b* on S1 and S2, respectively.

Given all possible combinations for the workers' production rates, we can now discuss different approaches to assign the workers. All possible assignment options are presented in Table 2 and explained in detail in the following sections.

3.2.1 No-sharing

The no sharing option assumes a fixed assignment. Thus, for two workers, they can either be assigned in W1 (assigned at S1) - W2 (assigned at S2) order or W2 (assigned at S1) - W1 (assigned at S2) order. The bottleneck rate, as determined by simply examining the k_{ij} values for the assignment, determines the output rate for the line where each worker only works at one station.

3.2.2 Sharing is allowed but the second worker in the order is never idle

Workers are assigned either in W1 - W2 or W2 - W1 order, and the second worker in the order is not allowed to be idle. If he/she finishes at S2 while the other worker is still completing work at S1, he/she takes work from the first worker in the order at S1. Once the first worker in the order is interrupted he/she waits until S1 is empty. Once the station empties he/she starts working on a new part. If the first worker in the order finishes first at his/her station, he/she is blocked and waits until the second worker in the order finishes his/her work at S2. This worker movement is the same as presented in the bucket brigade literature, except for the fact that the first worker is

idle until the first station becomes available as we are looking at discrete tasks. Note that in the case of two stations, the first worker in the order only works at the first station or is blocked by the second worker.

3.2.3 Sharing is allowed and the second worker can be idle

Workers are assigned either in W1 - W2 or W2 - W1 order, and if beneficial, either worker is allowed to be idle. If the second worker in the order is faster than the first worker at both stations, the assumptions are the same as those presented in 4.1.2 and the results from that section apply. However, if the first worker in the order finishes work at S1 prior to the second worker finishing at S2 and is faster at S2 than the second worker, he/she will interrupt the second worker and take his/her work at S2, instead of being blocked. The second worker will be idle, or wait, until the first worker finishes the part. After this part is finished, the first worker in the order starts a new part at S1. The second worker finishes work at S2. Handoff and walking times are zero as the assumption is that these operations are performed instantaneously.

3.3 TWO WORKER TWO STATION CASE ANALYSIS

Based on our assumptions there are six options, or total of six possible ways of assigning W1 and W2 to S1 and S2. All of the possible options are presented in Table 2. We studied the above described assignment options for the 12 cases presented in Table 1 and determined output rates for all of the options and case combinations. The optimal assignment of workers, from the six options given above, minimizes the total production time per part and thus results in the maximum output rate. Detailed analysis for Case 1 of Table 1 is given in the following section.

Analysis of Cases 2 to 12 is presented in Appendix A. Optimal assignments for all 12 cases are presented in Table 3 and the optimal output rates for all 12 cases are presented in Table 4.

Table 2: All assignment options

	Starting Assignment	**Assignment approach**
Option 1	W1 assigned to S1 W2 assigned to S2	No sharing
Option 2	W2 assigned to S1 W1 assigned to S2	No sharing
Option 3	W1 assigned to S1 W2 assigned to S2	Sharing is allowed and the second worker in the order cannot be idle
Option 4	W2 assigned to S1 W1 assigned to S2	Sharing is allowed but the second worker in the order cannot be idle
Option 5	W1 assigned to S1 W2 assigned to S2	Sharing is allowed and the second worker in the order can be idle
Option 6	W2 assigned to S1 W1 assigned to S2	Sharing is allowed and the second worker in the order can be idle

The optimal solution to the problem must satisfy the following equation:

$$x_{11}*k_{11} + x_{21}*k_{21} = x_{12}*k_{12} + x_{22}*k_{22} \quad (1)$$

which states that the output from the first station equals the output from the second station. Recall that x_{ij} is defined as the portion of time that worker i spends working at station j during the production horizon.

Case 1 analysis

Recall that for Case 1, presented in Table 1, $k_{11}=a$, $k_{12}=b$, $k_{21}=c$ and $k_{22}= d$ and that $a \leq b \leq c \leq d$. We now determine the maximum throughput for each of the worker assignment options given in Table 2.

Option *1*: $Output^1 = min\ (k_{11}, k_{22}) = min\ (a, d) = \mathbf{a}$.

Option *2*: $Output^2 = min\ (k_{21}, k_{12}) = min\ (c, b) = \mathbf{b}$.

Option *3*: Under the assumption that W1 is first in the order and W2 is second, then (1) becomes $x_{11}* k_{11} + x_{21}* k_{21} = x_{22}* k_{22}$ because $x_{12} = 0$. We know that $x_{12} = 0$ because the second worker, or *W2*, in this case, cannot be idle which implies that W1 can never be assigned to S2, thus $x_{12} = 0$. We also know that $x_{21}+x_{22} = 1.0$ because *W2* is always working and that the first station is always occupied, thus $x_{11}+ x_{21} = 1.0$. Solving these three equations with three unknowns results in $x_{21} = (k_{22} – k_{11})/(k_{22} + k_{21} – k_{11})$ and therefore $Output^3 = (1-x_{21})*k_{22} = x_{22}*k_{22} = k_{21}*k_{22}/(k_{22} + k_{21} – k_{11})$ $\mathbf{= c*d/(d + c – a)}$.

Option *4*: Under the assumption that W2 is first in the order and W1 is second (and cannot be idle) equation (1) reduces to $x_{21}* k_{21} = x_{12}* k_{12}$ *or* $x_{21}* c = x_{12}* b$. Because $\boldsymbol{c \geq b}$ *(or* $k_{21} \geq k_{12}$), W1 would never work at S1 and since W2 is faster, W2 will be idle a portion of the time. Thus, the output is: $Output^4 = min\ (k_{21}, k_{12}) = min\ (c, b) = k_{12} = \mathbf{b}$.

Option *5*: W1 is first in the order, and $\boldsymbol{a \leq d}$ *(*or $k_{11} \leq k_{22}$), thus, this situation is equivalent to Option 3, as W2 is never idle. W2 is faster on both S1 and S2, so once he/she finishes at S2, it is beneficial to interrupt W1 on S1, which results in the same output rate as Option 3.Thus $Output^5 = Output^3 = k_{21}*k_{22}/(k_{22} + k_{21} – k_{11}) = \mathbf{c*d/(d + c – a)}$.

Option *6*: W2 is first in the order, and the second worker in the order can be idle, thus W2 is assigned to both stations and W1 is assigned only to the second station. This is implied by the relationship of the production rates given. Then (1) reduces to: $x_{21}* k_{21} = x_{22}* k_{22} + x_{12}* k_{12}$, since we know that $x_{11}= 0$ and W2 is always busy thus, $x_{21}+x_{22} = 1.0$ and S2 is always occupied thus $x_{22} + x_{12} = 1.0$. Solving these equations, we have the following: $x_{22} = (k_{21} – k_{12})/(k_{22} + k_{21} – k_{12})$. The output from the line under this assignment option is: $Output^6 = (1-x_{22})*k_{21} = x_{21}*k_{21} = k_{22}*k_{21}/(k_{22} + k_{21} – k_{12})$ $\mathbf{= c*d/(d + c – b).}$

Under the assumption that $a \leq b \leq c \leq d$ (or $k_{11} \leq k_{12} \leq k_{21} \leq k_{22}$) we have to determine which option results in the maximum throughput. We can prove that for Case 1, the optimal assignment is Option 6. First, we note that $Output^2 = Output^4$ and $Output^3 = Output^5$. Next, we show that $Output^6 \geq Output^2$ (or $Output^4$). To do this, we have to demonstrate that $\mathbf{c*d/(d + c - b) \geq b.}$ After rearranging the terms, this inequality is equivalent to $\mathbf{(c - b)*(d - b)/ (d + c - b) \geq 0,}$ which is true given that $a \leq b \leq c \leq d$. This also implies that $Output^6 \geq Output^1$ *as* $\mathbf{b \geq a}$. It is also clear that $Output^3 \leq Output^6$ and $Output^5 \leq Output^6$, because $\mathbf{(c + d - a) \leq (c + d - b).}$ Thus the optimal assignment is Option 6, or to assign workers in W2 - W1 order and allow the second worker to be idle. Cases 2 to 12 are analyzed in a similar manner in Appendix A.

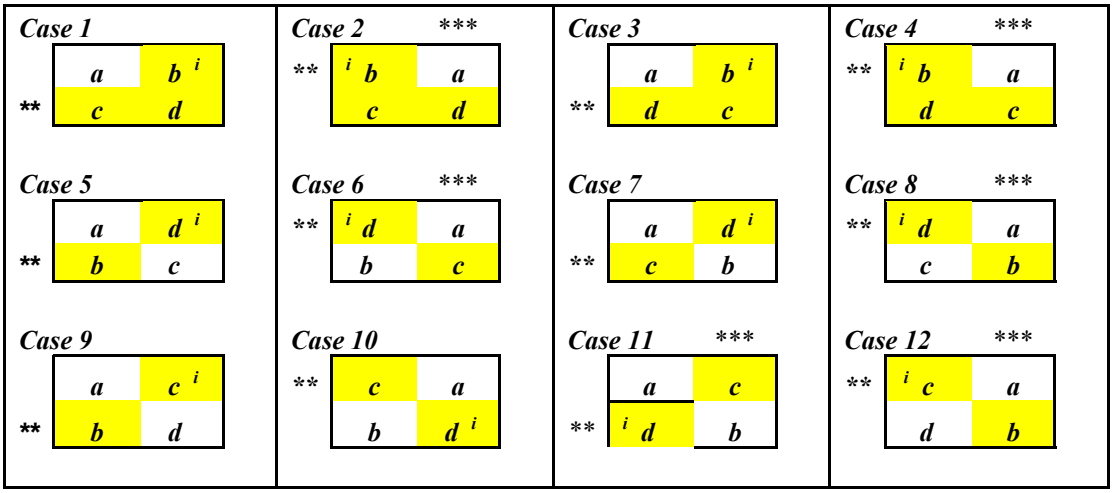
The optimal assignments for all 12 cases are presented in Table 3. Shaded cells are used to represent the assignment of the workers, and *i* represents idle time. For example, for Case 1, W2 is first in the order and is assigned to both stations, while W1 works at S2 and is idle part of the time. For Case 2, W1 is assigned to S1 and has idle time, while W2 is assigned to both stations and is never idle.

We now summarize our results. We start by considering Cases 1 to 4 where worksharing is beneficial. For Cases 1 and 3, the second worker is faster on both tasks but assigned at the beginning of the line. The optimal throughput will be achieved if we allow the second worker in the order to have idle time. This differs from Cases 2 and 4, when we have the faster worker second in the order.

Also, note that for the first four cases the worker-task combination with the smallest (or bottleneck) production rate is avoided. For Cases 5 to 12, it is optimal to assign workers in such a manner as to avoid the minimum production rate and not to allow sharing. Thus, when assignment approaches that consider sharing stations are applied to a two worker, two station

line, these can be beneficial only if one worker is faster on both tasks, otherwise we should assign workers under the fixed assignment no-sharing options.

Table 3: Optimal Assignments



Notes:** first worker in the order;*** same as bucket brigade; $a \le b \le c \le d$.

Currently, there are many practical implementations of the bucket brigade concept in manufacturing systems, assembly lines in particular, and warehousing and distribution (Bratcu and Dolgui, 2004). As application areas are continuously growing, we want to look at our results and determine what the output and optimal assignment would be if we run the system as a traditional bucket brigade system. In order to determine what would be the optimal assignment and optimal output rate if the system is restricted to operating as a traditional bucket brigade only, we can compare Options 3 and 4 and determine if the W1 - W2 or W2 - W1 order is better for the Case 1 production rates. When $\boldsymbol{b \ge c*d/(d + c - a)}$ then $Output^4 \ge Output^3$ and workers will be sequenced W2 - W1 or, for Case 1, the faster worker will be the first in the order. Example 1 presented below, shows that when $\boldsymbol{b \ge c*d/(d + c - a)}$, it is best to put W2 first in the order while W1 is assigned to S2 and W2 is idle due to blocking part of the time. The resulting throughput rate is 7 parts / hour. However, for this same data, if we consider all of the

worksharing options (including permitting idle time for the second worker), the best possible throughput rate is 7.2 parts/hour and the optimal order of the workers stays the same. In this situation, W1 is idle part of the time. This is presented in Example 1a where shaded cells are used to represent the assignment of the workers, and *i* represents idle time, so we can see that W1 is idle part of the time. One can see that W2 is assigned to both S1 and S2, while W1 is assigned to S2 only.

Example 1: b ≥ c*d/(d + c – a)

kij	*S1*	*S2*
W1	*6*	*7*
W2	*8* *i*	*9*

Throughput: 7 parts/hour

Example 1a: b ≤ c*d/(d + c – b)

kij	*S1*	*S2*
W1	*6*	*7* *i*
W2	*8*	*9*

Throughput: 7.2 parts/hour

When ***b ≤ c*d/(d + c – a)***, and we consider Options 3 and 4 only, the optimal assignment will be W1 - W2, or in this case the slower worker will be the first in the order. This is illustrated in Example 2 where the throughput rate is 11.2 parts / hour. However, if we consider all of the worksharing options and allow the second worker to be idle, the optimal worker order will change and W1 will be idle part of the time at S2 as presented in Example 2a. The optimal throughput will be 11.79 parts / hour. Note that in this case the faster worker is first in the order. These examples illustrate that the worker ordering can be faster to slower or vice versa and that there can be benefits to permitting the second worker in the order to have idle time (which does not occur in a traditional bucket brigade operation).

Example 2: b ≤ c*d/(d + c – a)

kij	*S1*	*S2*
W1	*i* *10*	*11*
W2	*14*	*16*

Throughput: 11.2 parts/hour

Example 2a: b ≤ c*d/(d + c – a)

kij	*S1*	*S2*
W1	*10*	*11* *i*
W2	*14*	*16*

Throughput: 11.79 parts/hour

Optimal output rates for all cases are summarized in Table 4. The optimal output rates under the six proposed options are presented in Column 6. The corresponding optimal worker order is presented in the last column.

Recall that Options 3 and 4 assume that the second worker in the order is not allowed to be idle. Under Options 3 and 4 when the second worker is always busy, for Cases 1 and 3 the optimal assignment changes based on the relationship of ***b*** and ***c*d/(d + c – a)***. For Cases 5, 7 and 9 the optimal assignment and the optimal output rate depend on the relationship of the terms presented in columns 2 and 4 in Table 4**.**

Table 4: Optimal output rates

1	*2*	*3*	*4*	*5*	*6*	*7*
	Output rate Options 3,and 4	**Optimal Order**	**Output rate Options 3 and 4**	**Optimal Order**	**Output rate All options**	**Optimal Order**
Case 1	***b***	***W2-W1***	***c*d/(d+c-a)***	***W1-W2***	***c*d/(d+c-b)***	***W2-W1***
Case 2	***c*d/(d+c-b)***	***W1-W2***			***c*d/(d+c-b)***	***W1-W2***
Case 3	***b***	***W2-W1***	***c*d/(d+c–a)***	***W1-W2***	***c*d/(d+c-b)***	***W2-W1***
Case 4	***c*d/(d+c-b)***	***W1-W2***			***c*d/(d+c-b)***	***W1-W2***
Case 5	***a*d/(a+d-b)***	***W2-W1***	***c*b/(b +c–a)***	***W1-W2***	***b***	***W2-W1***
Case 6	***c***	***W1-W2***			***c***	***W1-W2***
Case 7	***a*d/(a+d-c)***	***W2-W1***	***c*b/(b+c –a)***	***W1-W2***	***c***	***W2-W1***
Case 8	***b***	***W1-W2***			***b***	***W1-W2***
Case 9	***a*c/(a+c-b)***	***W2-W1***	***b*d/(b+d-a)***	***W1-W2***	***b***	***W2-W1***
Case 10	***b*d/(b+d-c)***	***W1-W2***			***c***	***W1-W2***
Case 11	***c***	***W2-W1***			***c***	***W2-W1***
Case 12	***b***	***W1-W2***			***b***	***W1-W2***

For example, for Case 5, if ***a*d/(a+d-b) ≥ c*b/(b +c–a)*** the optimal assignment will be W2 – W1, otherwise the optimal assignment would be W1 - W2 and the optimal output rate would be ***c*b/(b +c–a)***. Therefore, alternate output rates and assignments are presented in

columns 2 and 4 for Cases 1, 3, 5, 7 and 9. For Cases 2, 4, 6, 8, 10, 11 and 12 the optimal output and order would be the same as that presented in Column 6. It is interesting to note that if one is going to utilize bucket brigade types of rules where upon completion of a part the second worker in the line comes back to takeover work from the first worker that even if one worker's speed dominates another worker's that the optimal worker order depends on the values of *a, b, c* and *d* and is not necessarily as simple as placing the faster worker second.

Based on our findings, and considering task dependent worker production rates we can propose operational rules to managers in order to maximize productivity:

- In cases where one worker is faster than the other worker (the complete dominance case) the faster worker can be assigned at the beginning of the line, rather than the end of the line (Cases 1, 3 and 9). This result is contrary to the main bucket brigade result which states that the optimal production rate is achieved when workers are ordered slowest to fastest (Bartholdi and Eisenstein (1996, 1999)) and the work is spread evenly and continuously over the line, as well as McClain's result (McClain et al., 2000) that sequencing workers from slowest to fastest when preemption is allowed is a good starting point as it keeps the fastest worker busy. This result is similar to that found by Lim and Yang (2006).
- In all cases, the minimum production rate is avoided (when all six options are considered). This result is intuitive since we are trying to minimize the total time per part.

- When only Options 3 and 4 are considered (so that the line works as a traditional bucket brigade system), the optimal ordering assignment will change depending on the relationship of the workers' production rates (Cases 1, 3, 5, 7 and 9).
- If a worker is faster on one task but slower on the other task and we allow workers to be idle, sharing assignment approaches result in suboptimal throughput. Thus, in these cases worksharing is not beneficial.
- In some cases, as given in Example 3 below, the optimal assignment is not intuitive. In the Example the optimal order is W1-W2. At first glance it might seem that W2-W1 would be better because W2 is extremely fast on S1. However, the optimal production rate in this case is *b* (or 8) and the optimal assignment is W1 on S1 and W2 on S2. To see this, consider the output rates for all six assignment options.

Example 3.

k_{ij}	*S1*	*S2*	k_{ij}	*S1*	*S2*
W1	*c* (*i*)	*a*	*W1*	*9* (*i*)	*7*
W2	*d*	*b*	*W2*	*100*	*8*

Option 1: $Output^1 = min\ (k_{11}, k_{22}) = min\ (c, b) = b$.

Option 2: $Output^2 = min\ (k_{21}, k_{12}) = min\ (d, a) = a$.

Option 3: Under this Option, the second worker in the order is allowed to work at the first station, but as $k_{22} \leq k_{11}$ this never occurs, and the output rate is equivalent to Option 1.

***Option 4*:** Using similar reasoning as for Option *3*, $Output^4 = a$.

***Option 5*:** Under this option, the second worker in the order can be idle if beneficial. Because $k_{22} \leq k_{11}$, this never occurs, and thus $Output^5 = Output^1 = b$.

***Option 6*:** The second worker in the order can be idle, and because $k_{12} \leq k_{22}$, this can be beneficial, and $Output^{6} = k_{22}*k_{21}/(k_{22} + k_{21} - k_{12}) = b*d/(d + b - a)$. One can readily show that Option 1 is optimal with the assignment W1 - W2. Thus, in this case the faster worker will be the second in the order but never utilize his/her extremely high production rate at S1.

We presented detailed analysis of the two worker two station production line with discrete stations and station dependent workers' production rates. Our analysis suggests that under our assumptions worksharing is only helpful when one worker's slowest rate is faster than the second worker's fastest rate. Otherwise, fixed assignment options result in better throughput. Additionally, our analysis indicates that ordering workers from slowest to fastest in cases where this is possible, may not produce optimal throughput. In some Cases, especially for options where sharing is allowed, it is beneficial to have the second worker in the order have idle time. For all Cases, the minimum production rate should be avoided.

3.4 WORKSHARING DYNAMICS FOR TWO WORKERS AND DUPLICATE STATIONS

In this section we look at possible benefits of duplicating tooling at one or both stations. Based on our observations and results for two worker two station production lines we wanted to study possible advantages of duplicating stations. The modeling assumptions are the same as presented in Section 3.1 for the two worker two station production line. The additional assumption is that if the station has duplicate tooling both workers can be assigned to the same station at the same time.

We also calculate the output when the system performs as two parallel lines, in the case of duplicating both stations. We compare the results for duplicating one station with the option of creating a parallel line.

Note that for this problem variation workers need not ever be idle due to the presence of the duplicate station (assuming that the correct station is duplicated and that the workers are ordered optimally). Our goal is to sequence the workers in a way that utilizes the duplicate tooling for the maximum benefit.

3.4.1 Two workers and duplicate station lines

We define the objective function as minimizing the total time required per part produced. We analyze Cases 1 to 12 presented in Table 1 when we duplicate either S1 or S2 or both S1 and S2 and determine optimal output rates for each case. Based on our assumptions there are four options for assigning workers W1 and W2 to the two stations. All options assume that sharing is allowed (when we have duplicate tooling), so that both workers can be at the same station at the same time. For example, if W2 is assigned to S2, W1 to S1, and S1 has duplicate tooling, then if W2 finishes first he/she walks back and takes the part from W1, at the same time W1 starts a new part and thus there is no idle time. All of the assignment options are presented in Table 5.

In order to determine the output rate for each Option in Table 5, we first calculated the time needed to finish each part. For example, for Option 1, if we assume that at time zero W2 starts working at S2 (assuming that one part was already processed at S1), W1 starts working on a new part at S1 and S1 has duplicate tooling, the total time required to finish a part is the time W2 takes at S2, t_{22}, plus the time W2 needs to finish the part at S1. Note that in order for this option to be logical, we need to assume that $t_{22} < t_{11}$. Otherwise W2 will never work at S1 and there is no reason to have duplicated the tooling at S1. The time that W2 spends at S1 is

calculated as $(1- t_{22}/t_{11})/(1/t_{21}+1/t_{11})$ where the numerator represents the portion of the part that is not finished by W1 after t_{22} time units and the denominator represents the total production rate of the two workers at S1. We applied the same reasoning for all four options.

Table 5: Duplicate Tooling Assignment Options

	Starting Assignment	Duplicate tooling at station
Option 1	W1 assigned to S1 W2 assigned to S2	*S1*
Option 2	W1 assigned to S1 W2 assigned to S2	*S2*
Option 3	W2 assigned to S1 W1 assigned to S2	*S1*
Option 4	W2 assigned to S1 W1 assigned to S2	*S2*

The output rates are calculated as *1/time per part* and are given below.

Option 1: *Time per part* $= t_{22} + (1-t_{22}/t_{11})/(1/t_{21}+1/t_{11}) = (t_{22}t_{11}+t_{21}t_{11})/(t_{11}+t_{21})$. Thus, $Output^1$ = *1/time per part* = $\boldsymbol{(k_{21}+k_{11})/(k_{21}/k_{22}+1)}$.

Option 2: *Time per part* $= t_{11} + (1-t_{11}/t_{22})/(1/t_{12}+1/t_{22}) = (t_{11}t_{22}+t_{22}t_{12})/(t_{22}+t_{12})$. Thus, $Output^2$ = *1/time per part* = $\boldsymbol{(k_{12}+k_{22})/(k_{12}/k_{11}+1)}$.

Option 3: *Time per part* $= t_{12} + (1-t_{12}/t_{21})/(1/t_{11}+1/t_{21}) = \boldsymbol{(t_{12}t_{21}+t_{11}t_{21})/(t_{21}+t_{11})}$. Thus, $Output^3$ = *1/time per part* = $\boldsymbol{(k_{11}+k_{21})/(k_{11}/k_{12}+1)}$.

Option 4: *Time per part* $= t_{21} + (1-t_{21}/t_{12})/(1/t_{22}+1/t_{12}) = \boldsymbol{(t_{21}t_{12}+t_{22}t_{12})/(t_{22}+t_{12})}$. Thus, $Output^4$ = *1/time per part* = $\boldsymbol{(k_{22}+k_{12})/(k_{22}/k_{21}+1)}$.

For some cases, depending on the relationship of the workers' production rates, it is not advantageous to duplicate either S1 or S2, as this station would not be used. For example, if we look at Case 4 presented in Table 1, it is not beneficial to duplicate S1 if the workers are ordered W2-W1 as the duplicate station would never be used since $a \leq d$. For the 12 cases, output rates

were determined for feasible (practical, logically possible) options, where feasible means that sharing could actually occur at the duplicated station based on the values of *a, b, c* and *d*.

Duplicate Tooling Case 1 analysis

Recall that for Case 1 we have the production rates given in Table 1 and that $a \leq b \leq c \leq d$. Given the data for Case 1, Options 1 and 4 are analyzed because they are the only feasible ones. If we look at Option 2, we can conclude that it is not feasible because it is not beneficial to duplicate S2 when the workers are ordered W1- W2 since $k_{11} \leq k_{22}$ (based on our assumption that $k_{11}=a \leq k_{22} = d$). The same reasoning applies for Option 3. For Case 1, duplicate tooling in Options 1 and 4 is beneficial because output will be higher if we have duplicate tooling. Thus, we need to compare the output rates for Options 1 and 4 to determine the optimal output.

We determine the output rates based on the calculations presented in 5.1., thus:

Option 1: $Output^1 = (k_{21}+k_{11})/(k_{21}/k_{22}+1) =$ ***(c+a)/(c/d+1)***.

Option 4: $Output^4 = (k_{22}+k_{12})/(k_{22}/k_{21}+1) =$ ***(d+b)/(d/c+1).***

When we compare these output rates, it reduces to the relationship between *a*d* and *b*c*. If *a*d* > *b*c* it is optimal to order the workers W1 - W2 and duplicate S1. Otherwise, it is optimal to order the workers W2 - W1 and duplicate S2.

Duplicate Tooling Case 2 analysis

The workers' production rates for Case 2 are presented in Table 1. For Case 2, it is reasonable to only analyze Options 1 and 4 because output will be higher if we have duplicate tooling. If we look at Option 3, we can conclude that it is not beneficial to duplicate S1 when the workers are

ordered W2 -W1 since $k_{21} \geq k_{12}$ (based on our assumption that $k_{21}=c \geq k_{12} = a$). The same reasoning applies for Option 2.

We determine the output rates based on the calculations presented in 5.1, thus:

***Option 1**: Output*^{1} $= (k_{21}+k_{11})/(k_{21}/k_{22}+1) =$ **$(b+c)/(c/d+1)$.**

***Option 4:** Output*^{4} $= (k_{22}+k_{12})/(k_{22}/k_{21}+1) =$ **$(d+a)/(d/c+1)$.**

When we compare ***(b+c)/(c/d+1)*** and ***(d+a)/(d/c+1)***, it reduces to the relationship between ***d*b*** and ***a*c***. Thus, Option 1 is optimal because ***d*b*** is always greater than ***a*c***.

For the two worker line with duplicate stations we looked at situations where it is beneficial to duplicate either S1 or S2, based on the workers' production rates. Similar to the analysis presented for Case 1 and Case 2 we determined the feasible options for all cases and calculated output rates for these option and case combinations. In order to determine the optimal assignment option for each case, the output rates for the feasible worker assignments were compared for each case where feasible means that sharing could actually occur at the duplicated station based on the values of *a, b, c* and *d*. If sharing could not occur the production rate was not calculated and the case/option combination was not considered (not applicable). For some cases, such as Case 2, one Option dominated the other (such as Option 1 for Case 2). In other cases, such as Case 1, the optimal option depends on the relative values of the workers' production rates. Table 6 presents a summary of the feasible options and indicates which worker ordering option is optimal for each case.

Table 7 shows the corresponding optimal throughput rates for each case. In the event that neither worker ordering Option dominates the other, the expressions that determine the breakpoint for choosing one Option versus the other are given. Detailed calculations for all option and case combination are presented in Appendix A.

Table 6: Feasible Options and Break Points

	Option 1 O1	***Option 2 O2***	***Option 3 O3***	***Option 4 O4***	***Optimal Option***
Case 1	***a*d***	***NA***	***NA***	***b*c***	***O1 or O4***
Case 2	***b*d***	***NA***	***NA***	***a*c***	***O1***
Case 3	***a*c***	***NA***	***NA***	***b*d***	***O4***
Case 4	***b*c***	***NA***	***NA***	***a*d***	***O1 or O4***
Case 5	***1/(b/c+1)***	***NA***	***1/(a/d+1)***	***NA***	***O3***
Case 6	***NA***	***1/(a/d+1)***	***NA***	***1/(c/b+1)***	***O2***
Case 7	***1/(c/b+1)***	***NA***	***1/(a/d+1)***	***NA***	***O3***
Case 8	***NA***	***1/(a/d+1)***	***NA***	***1/(b/c+1)***	***O2***
Case 9	***1/(b/d+1)***	***NA***	***1/(a/c+1)***	***NA***	***O1 or O3***
Case 10	***c*d***	***NA***	***NA***	***a*b***	***O1***
Case 11	***a*b***	***NA***	***NA***	***c*d***	***O4***
Case 12	***NA***	***1/(a/c+1)***	***NA***	***1/(b/d+1)***	***O2 or O4***

Note: NA not applicable

Assuming that tooling costs are the same for duplicating either S1 or S2, we can summarize the optimal assignments and scenarios for all cases. For Case 1, it is likely that either Option (*O1* or *O4*) may be optimal as it is possible to have workers' production rates such that *a*d* is greater than, equal to or smaller than *b*c*. Cases 4, 9, and 12 are similar in that the optimal assignment Option depends on the workers' production rates. For Case 2, it is clear that *O1* (Option 1) is always the best Option, thus W1 would be first in the order and we would have a line with duplicate tooling at S1. Similarly, Cases, 3, 5, 6, 7, 8, 10, and 11 always have one Option that dominates the other and thus the choice of which tool to duplicate and how to order the workers is clear. If the costs of duplicating the different stations are different than one could conduct an economic analysis comparing the benefits of the increased throughput with the cost of duplicating the stations.

3.4.2 Two parallel lines

We now consider the benefit of duplicating both stations. If we assume that both stations have duplicate tooling, we can look at the system as two parallel production lines. By two parallel lines we mean that each worker works at both stations in sequential order and finishes parts independently of the other worker.

Table 7: Optimal output rates

	One station with duplicate tooling	**Two parallel lines**
Case 1	*(c+a)/(c/d+1) or (d+b)/(d/c+1)*	*a*b/(a+b)+c*d/(c+d)*
Case 2	*(b+c)/(c/d+1)*	*a*b/(a+b)+c*d/(c+d)*
Case 3	*(b+c)/(c/d+1)*	*a*b/(a+b)+c*d/(c+d)*
Case 4	*(b+d)/(d/c+1) or (a+c)/(c/d+1)*	*a*b/(a+b)+c*d/(c+d)*
Case 5	*(a+b)/(a/d+1)*	*a*d/(a+d)+c*b/(c+b)*
Case 6	*(a+c)/(a/d+1)*	*a*d/(a+d)+c*b/(c+b)*
Case 7	*(a+c)/(a/d+1)*	*a*d/(a+d)+c*b/(c+b)*
Case 8	*(a+b)/(a/d+1)*	*a*d/(a+d)+c*b/(c+b)*
Case 9	*(a+b)/(b/d+1)or (a+b)/(a/c+1)*	*a*c/(a+c)+d*b/(d+b)*
Case10	*(b+c)/(b/d+1)*	*a*c/(a+c)+d*b/(d+b)*
Case11	*(b+c)/(b/d+1)*	*a*c/(a+c)+d*b/(d+b)*
Case12	*(a+b)/(a/c+1)or (a+b)/(b/d+1)*	*a*c/(a+c)+d*b/(d+b)*

Optimal throughput rates are presented in Table 7 for two parallel lines (both stations have duplicate tooling) assuming the same worker production rates as presented in Table 1. It can be shown that the throughput obtained from two parallel lines is at most as good as the output produced when we have optimally ordered the workers and have chosen the one correct station to duplicate. For example, for Cases 6 and 7, it can be shown that $a*d/(a+d)+ c*b/(c+b) \leq (a+c)/(a/d+1)$. Applying simple algebra, the expression reduces to $a*b \leq c*d$. Based on the

assumed relationships of ***a, b, c, d*** we can say that the output rates from two parallel lines for Cases 6 and 7 are always lower than the optimal worker ordering combined with the best choice of duplicate tooling. The analysis for all other cases is presented in Appendix A.

3.4.3. Summary

For the two worker line with duplicate stations we focused on duplicating tooling on the stations where it would be best utilized, depending on the workers' production rates. When we duplicate tooling the best possible way, the optimal output rate can be achieved and we can show which station to duplicate for the different worker production rate scenarios. We also analyzed how the system performs if we duplicate tooling at both stations and run the system as two parallel lines. Our main result is that the output obtained from two parallel lines is never greater than the output obtained from a line with two stations when one of those stations has duplicate tooling and the workers are properly sequenced.

3.5 SUMMARY

In this chapter we concentrated on a two worker two station serial production line and analyzed different worker assignment policies as well as different levels of worksharing to determine their impact on system performance. We determined conditions for which worksharing is most effective, as well as optimal throughput rates. We also considered the value of duplicating workstations.

4.0 DYNAMIC ASSIGNMENT: ONE-CYCLE FORMULATION

In Chapter Three we studied two worker two station production lines and compared different assignment approaches and policies. A logical extension is to consider and study larger serial production lines with more workers and stations. In this chapter we discuss and analyze larger production lines and settings. We study a production line on its own and we look at a set of production lines operating together as an independent or dependent system. An example of this approach is when we have parts or work pieces from two or more input production lines combined to form a new line for further processing. Our goal is to determine the optimal worker selection, order and assignment based on workers' production rates. Thus, we solve the worker assignment problem for a single production line and later we solve the problem of worker selection and assignment from a larger pool of workers to a set of production lines running as an independent or dependent production system.

4.1 MODELING ASSUMPTIONS

We develop a mixed integer programming formulation that models constant one-cycle behavior with the objective to maximize the throughput and optimally assign workers. For a one-cycle formulation, workers exchange parts at the same point (i.e., for each part each worker performs exactly the same portion of the work). We want to determine the optimal percentage of work that

workers perform on each part in the case of a balanced line. We assume that a line is balanced if workers exchange parts at the same point so that each worker repeats the same work content on each successive part (Bartholdi 1996). Under traditional bucket brigade rules workers are only idle when waiting in front of a busy station and the last worker in the order (the worker who finishes the product) is never idle. Our formulation allows for the last worker in the order to be idle if his/her idle time benefits the total production or total throughput.

In the previous chapter we presented a two worker two station production line analysis and we now extend our study to a n worker m station line where $n < m$. The modeling assumptions are the same as presented in Chapter Three. Work is concentrated in various proportions among discrete stations. Workers perform at steady-state production rates, processing times are deterministic and each part needs processing on the same sequence of stations. As mentioned in Chapter Three, workers may block each other especially in situations when the worker to station ratio is high (close to 1).

An illustrative example of steady-state productivity rates, k_{ij} values, for a line with six workers and twelve stations is presented in Table 8. This is an incomplete dominance case where it is not possible to order workers from slowest to fastest.

Table 8: Incomplete dominance case: steady-state production rates

k_{ij}	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12
W1	25	38.2	29.2	29.1	28.2	35.7	33.1	35.7	37.4	30.5	27.4	37.9
W2	25.9	32.8	34.8	26.1	37.9	32.3	28.8	34	35.2	30.8	25.5	31.5
W3	37.5	25.7	37.4	33.6	26.1	37.7	30.2	39.3	29.5	38.8	27	32.9
W4	38.6	34.5	36.9	37.7	37.3	36.4	27.1	35.2	35.3	27.5	34.6	37.1
W5	26.8	33.8	38.1	36	28.6	27.6	30.1	31.9	25.6	31.8	30.6	38.4
W6	36.6	31.2	30.8	26.5	35.5	28.6	36.1	26.7	31.3	33.4	39.7	30.5

In the table above, lightly shaded cells represent the fastest rates or the fastest worker for a specific task/station, where cells with a darker shade represent the slowest worker for a specific task/station. For example, worker 1 (W1) is slowest at stations 1 (S1) and 3, and fastest on S2 and S9. If we examine the table we can conclude that the situation is similar for other workers (rows represent workers and columns represent stations), thus it is not simple to assign workers based on their highest production rates.

4.2 ONE-CYCLE FORMULATION

Our goal is to determine the optimal worker assignment and optimal level of worksharing, or the exact percentage of work performed at each station. Under worksharing we assume that more than one worker is assigned to a particular station during the production horizon (not at the same moment as only one worker is allowed to work at the station at one time). The input buffer has infinite inventory and there are no other buffer inventories. Two workers share at most one station (not at the same time) as exchange of the part occurs always at the same position. Each worker works only at adjacent stations. The starting original order of the workers is preserved. The notation used in this section can be found in the Nomenclature section.

Objective function

Maximize *Output*

Constraints

$$w_j = \sum_{i=1}^{n} x_{i,j}\, k_{i,j} \qquad \forall j \qquad (1)$$

$$Output \leq w_j \qquad \forall j \qquad (2)$$

$$\sum_{i=1}^{n} x_{i,j} \leq 1 \qquad \forall j \qquad (3)$$

$$\sum_{j=1}^{m} x_{i,j} + idle_i \leq 1 \qquad \forall i \qquad (4)$$

$$x_{i,j} \leq z_{i,j} \qquad \forall i, \forall j \qquad (5)$$

$$M\, x_{i,j} \geq z_{i,j} \qquad \forall i, \forall j \qquad (6)$$

$$-abs_{i,j} \leq z_{i,j} - z_{i,j+1} \qquad \forall i \qquad (7)$$

$$abs_{i,j} \geq z_{i,j} - z_{i,j+1} \qquad \forall i \qquad (8)$$

$$z_{i,1} + \sum_{j=1}^{|J|-1} abs_{i,j} + z_{i|J|} \leq 2 \qquad \forall i \qquad (9)$$

$$z_{i,j} + z_{i,j+1} + z_{i',j} + z_{i',j+1} \leq 3 \qquad \forall i, \forall i'(i' \neq i),\ j=1,\ldots,m-1 \qquad (10)$$

$$z_{i,j-1} + z_{i,j} + z_{i,j+1} + \sum_{i' \neq i} z_{i',j} \leq 3 \qquad \forall i, , \forall i'(i' \neq i), \forall j=2,\ldots,m-1 \qquad (11)$$

$x_{ij} \geq 0$, $w_j \geq 0$, $Output \geq 0$, $idle_i \geq 0$, $abs_{i,j} \geq 0$ and z_{ij} binary.

The objective is to maximize throughput from the serial production line during the production horizon. Constraint set *(1)* determines the output from each station. Constraint set *(2)* models the idea that the output from the last station can be at most the output of the bottleneck station.

Constraint set *(3)* provides that the maximum work assigned to a station cannot be more than one, as we assumed that the duration of the production horizon is either one hour, one day, etc. Consequently constraint set *(4)* provides that each worker works at most the duration of the production horizon, including the time that worker is idle. Variable x_{ij} indicates the portion of the production horizon that worker i spends doing task j. If x_{ij} is positive, meaning that worker i

performs task *j,* an indicator variable $z_{i,j}$ will equal 1. Otherwise $z_{i,j}$ will be zero as given in constraints *(5)* and *(6).* In order to model work at adjacent stations or model the idea that if a worker works at station *j-1* and *j+1*, he/she needs to work at station *j,* the following constraints were introduced for each worker

$$z_{i,1} + \sum_{j=1}^{|J|-1} \left| z_{i,j} - z_{i,j+1} \right| + z_{i|J|} \leq 2.$$

In order to model this idea and to model the absolute values, constraints *(7), (8)* and *(9)* were introduced.

To model the idea that each worker covers a portion of a line and that two workers share at most one station, we introduced constraints *(10)* and *(11)*. These constraints also provide that if a worker is assigned to station *j-1* and station *j+1*, he/she will be the only worker assigned to station *j*.

4.2.1 Two workers three stations numerical examples

First we start with a production line of two workers and three stations. For a general case, the final throughput is calculated for a line with *n* workers and *m* stations. Worker productivity levels represent steady-state production rates and $k_{i,j}$ values were randomly generated based on the steady state levels given in Shafer et al. (2001). For each set of k_{ij} values we present x_{ij} as the portion of time that worker *i* spends working at station *j* during the production horizon. We also present the final throughput obtained.

Several examples for a line with two workers and three stations are presented. In Table **9** an example is given where W2 is faster at each station, for example k_{21} is higher than k_{11}, but also lower than k_{12}, etc. The optimal solution obtained is also presented in the table and W2 is optimally assigned on S1 and S2 and W1 on S2 and S3. W2 spends 90% of his/her time on S1

and 10% on S2. W1 spends 55% of his/her time on S2 and the rest on S3. The objective function value is 2.7 parts / hour. We assumed that the production horizon length is one hour. Interestingly, though W2 is faster than W1, W2 is not at the end of the line as in a "traditional" bucket brigade case.

Table 9: Data set with two workers and three stations

k_{ij}	S1	S2	S3
W1	2	4	6
W2	3	5	7

x_{ij}	S1	S2	S3
W1		0.55	0.45
W2	0.9	0.1	

Objective **2.7 parts/hour**

In Table 10 we present a complete dominance case. W2 is faster than W1 at any station. An optimal solution is also presented where the faster worker, W2 in this case, is first in the order. W2 spends 61% of his/her time at S1 and the remaining 39% at S2. There is no idle time.

Table 10: Slower worker at the end of the line

k_{ij}	S1	S2	S3
W1	5	7	9
W2	11	13	15

x_{ij}	S1	S2	S3
W1		0.25	0.75
W2	0.61	0.39	

Objective **6.75 parts/hour**

An example of the last worker being idle is presented in Table 11. Once W2 finishes at the last station (or the part is finished), it is better for him/her to wait for W1 to finish his/her work at S1 and then exchange a part rather than exchange earlier in which case W1 would wait and it would take longer for W2 to finish the part at S1, as k_{32} is smaller than k_{31}. The objective function value or the total throughput in this case is 6 parts per hour.

Table 11: An example of idle time

k_{ij}	S1	S2	S3	x_{ij}	S1	S2	S3	idle
W1	2	12	14	**W1**		0.5	0.42	0.08
W2	6	6	10	**W2**	1			
Objective	**6 parts/hour**							

4.2.2 N workers M stations numerical examples

In this section we present the application to larger production lines. An example of a line with six workers and twelve stations is presented. The optimal solution obtained for the production rates given in Table 8 is presented in Table 12. W4 is assigned to S1 and S2, and he/she shares S2 with W5 (not at the same moment), etc. The last row represents utilization of each station and the last column sums the total proportion of time that the workers are busy. The order of the workers presented here is as follows: W4 is at S1 and S2, W5 is at S2 to S4, W6 is at S4 to S7, etc. The objective function value is 18.13 parts/hour. We assume that the production horizon length is one hour.

Table 12: Math formulation solution: Example

	S1	*S2*	*S3*	*S4*	*S5*	*S6*	*S7*	*S8*	*S9*	*S10*	*S11*	*S12*	*SUM*
W4	***0.57***	***0.43***											***1.0***
W5		***0.07***	***0.49***	***0.44***									***1.0***
W6				***0.06***	***0.46***	***0.49***							***1.0***
W1						***0.01***	***0.51***	***0.46***	***0.01***				***1.0***
W3									***0.49***	***0.49***	***0.01***		***1.0***
W2											***0.46***	***0.54***	***1.0***
U	***0.57***	***0.49***	***0.49***	***0.50***	***0.46***	***0.50***	***0.51***	***0.46***	***0.50***	***0.49***	***0.48***	***0.54***	

We also tested data sets for production lines with two to eight workers and three to sixteen stations. Selected examples are given in Appendix B.

4.2.3 Summary

Our primary focus was to optimally solve the assignment problem and determine exact portions of work performed by each worker under the assumptions presented. We developed a mixed integer programming formulation that models one-cycle balanced line performance where workers exchange parts at exactly one position. We are able to obtain optimal positioning of workers and the optimal amount of work performed by each worker. Based on workers' production rates we can determine the optimal throughput and exact workers' assignments. This represents the maximum throughput with worksharing under the given assumptions.

4.3 MULTIPLE PRODUCTION LINES

In situations where several production lines are used simultaneously or parts assembled or produced at one line are used as a beginning inventory for another line, we need to look at that production scenario as one whole problem. Analyzing each line independently is not accurate, as in order to produce at line B workers need finished parts from line A. Sewing garments, or manufacturing cars are possible examples. In these cases we can state that lines A and B are not independent.

Now we extend our study to a set of production lines. We developed a mixed integer programming formulation for selecting workers to different lines as well as finding an optimal assignment of workers. This concept can be applied when worker groups are formed from a larger pool of assembly workers.

The formulation presented in section 4.2 is modified in order to solve a different assignment problem. The problem defined is the selection and assignment of workers to several production lines based on the defined production requirements and the given workers' production rates. We can either look at the system as a set of l independent lines or a set of l linked or dependent lines.

We have a total of n workers and m stations where parts need to be processed on consecutive stations and the sum of all the stations of all lines equals m. Our goal is to determine how to optimally select workers for each line, as well as how to assign workers to stations. Modeling assumptions are the same as presented in section 3.1.

An example of ten workers and fifteen stations is presented in Table 13. In this case we assumed that these fifteen stations actually represent five independent production lines, each line consisting of three stations. Workers' production rates are assumed to be deterministic and known for each station. Based on workers' production rates and line break-points (beginning and end of each line), we can select workers for each line as well as assign them in the proper order in order to achieve maximum throughput.

We formulate the problem as follows: from a group of n workers with given production rates on m tasks, assign workers to each line in order to satisfy given constraints and to maximize the objective. In the first case we define the objective function as an equally weighted sum of throughput obtained from each line. Also, the notation used in this section can be found in the Nomenclature section.

We also need to define the allocation of stations to lines, so for the example given above we'll assign S1, S2 and S3 to line 1 (L1), S4, S5 and S6 to L2, etc. The assignment of stations to lines is given in Table 13. The formulation is presented below.

Table 13: Data set 1: Worker/station production rates

k_{ij}	Line 1			Line 2			Line 3			Line 4			Line 5		
W/S	**S1**	**S2**	**S3**	**S4**	**S5**	**S6**	**S7**	**S8**	**S9**	**S10**	**S11**	**S12**	**S13**	**S14**	**S15**
W1	11	9.4	10	9.2	9.5	10	11	10	9	12	11	10	10	9.9	12
W2	7	8.8	5.6	6.2	7.7	9	7	8.4	6.8	10	8.7	7	9.5	8.3	10.2
W3	9	7.7	6	5.7	5.1	6.1	5.4	9	6.1	7.5	8.1	9.2	9.9	7.1	8.3
W4	6	8.1	9	9	8	7.9	7	8	6.8	8.1	8.7	9.3	7.4	6.4	8
W5	10	6	7.6	6.1	5.7	5.8	6.2	5.4	6.7	7.6	8	9	6.6	7.7	7.2
W6	8	7.3	6	9	6.2	5	7.1	7	6.1	6.9	8.1	9.2	7.7	8.8	9.9
W7	7.2	6.9	7	7	8	6	7	8	6.8	8.1	8.7	9.3	6.2	6.6	8
W8	6.7	7.1	9.4	7.9	5.6	6.6	8.8	6.8	6.2	9	10.6	6.2	8.3	8.5	7.2
W9	7	8.9	6	8.8	5.7	7.2	9	5.2	6.1	7.5	8.1	9.2	7.4	5.5	10
W10	4.5	4.2	4.1	5.4	5	4	4.9	4.9	5	5.2	6.7	3.2	5.5	4.6	5.9

Objective function

$$\textbf{\textit{Maximize}} \quad \sum_{o=1}^{l} Output_o$$

Constraints

$$w_j = \sum_{i=1}^{n} x_{i,j}\, k_{i,j} \qquad \forall j \qquad (1)$$

$$Output_o \leq w_j \qquad \forall j \in l, \forall l \qquad (2)$$

$$\sum_{i=1}^{n} x_{i,j} \leq 1 \qquad \forall j \qquad (3)$$

$$\sum_{j=1}^{m} x_{i,j} + idle_i \leq 1 \qquad \forall i \qquad (4)$$

$$x_{i,j} \leq z_{i,j} \qquad \forall i, \forall j \qquad (5)$$

$$M\, x_{i,j} \geq z_{i,j} \qquad \forall i, \forall j \qquad (6)$$

$$-abs_{ij} \leq z_{i,j} - z_{i,j+1} \qquad \forall i \qquad (7)$$

$$abs_{ij} \geq z_{i,j} - z_{i,j+1} \qquad \forall i \qquad (8)$$

$$z_{i,1} + \sum_{j=1}^{|J|-1} abs_{i,j} + z_{i|J|} \leq 2 \qquad \forall i \qquad (9)$$

$$z_{i,j} + z_{i,j+1} + z_{i',j} + z_{i',j+1} \leq 3 \qquad \forall i, \forall i'(i' \neq i), j=1,\ldots,m-1 \qquad (10)$$

$$z_{i,j-1} + z_{i,j} + z_{i,j+1} + \sum_{i' \neq i} z_{i',j} \leq 3 \qquad \forall i, \forall i'(i' \neq i), \forall j=2,\ldots,m-1 \qquad (11)$$

$$\sum_{j \in Sp} z_{i,j} \leq 3\, g_{il} \qquad \forall i, \forall l \qquad (12)$$

$$10 \sum_{j \in Sp} z_{i,j} \geq g_{il} \qquad \forall i, \forall l \qquad (13)$$

$$\sum_{l=1}^{o} g_{il} \leq 1 \qquad \forall i \qquad (14)$$

$$\sum_{i=1}^{n} g_{il} \leq v \qquad \forall l \qquad (15)$$

$x_{i,j} \geq 0$, $w_j \geq 0$, $Output \geq 0$, $idle_i \geq 0$, $abs_{i,j} \geq 0$, $z_{i,j}$ binary and $g_{i,l}$ binary.

In order for a worker to be assigned to one line only, we introduced another binary variable g_{io}. If worker i is assigned to line o, then g_{io} is one, otherwise it is zero. The objective given above is to maximize total throughput obtained from the serial production lines during the production horizon. Constraints *(1)* to *(11)* are identical to the constraints presented in section 4.2. If worker i works at station j, then z_{ij} is one as described above. Also, if at least one z_{ij}, for stations that belong to line l, is one then the corresponding g_{il} is also one. Constraints *(12)* and *(13)* capture that if the z_{ij} values corresponding to line l are all zero then the variable g_{il} is zero. Constraints *(14)* provide that a worker is assigned to at most one line. The fact that at most v workers are assigned to one line is modeled by constraints *(15)*.

4.3.1. Optimality Criteria

The objective given in section 4.3 can be modified in order to maximize the total weighted throughput from the considered serial production lines during the production horizon. We can assign different weights to each line as given by $weight_o$ in order to construct more realistic scenarios. Thus $weight_o$ is defined as the coefficient assigned to each line. The modified objective can be written as *Maximize* $\sum_{o=1}^{l} weight_o Output_o$.

In situations when higher throughput is needed from several lines we can assign more workers accordingly. Thus, instead of having the same number of workers and/or stations at each line, we can modify the formulation and satisfy requirements such as having different numbers of workers and/or stations at different production lines. For the examples tested $v=2$, which implies that at most two workers work at each line. Also, we tested lines with three stations. We could expand the formulation and examine production lines with different numbers of stations as well as allowing different numbers of workers at each production line. Another extension is defining a minimum necessary throughput from each line based on production requirements.

Another possible assumption is that we have l connected cells or l linked (dependent) production lines where production at line l depends on finished products from line l-1, etc. The product or an item is finished if it was processed on all l successive lines.

The modified objective can be written as ***Maximize*** *Output* where the variable *Output* is defined as the output from the last line (or finished product). The new set of constraints introduced is $Output \leq Output_o, \ \forall o$. New constraints introduced would insure that the output from the last line is at most the output of the line with the lowest output.

4.3.2 10 workers 15 stations (5 production lines)

We primarily tested production lines with two workers and three stations. However, examples with six, eight and ten workers and nine, twelve and fifteen stations respectively were also tested and solved optimally. CPLEX solution times varied from 20 seconds to 450 seconds for independent lines, and were much higher for five dependent (or connected) lines. The solution times for the data sets tested ranged from 200 to 10000 seconds.

For data set 1, given in Table 13, based on the production rates given, it can be seen that W1 is faster than other workers at all stations, W10 is slowest at all stations and task S9 is defined as a hard task (the lowest production rates for all workers). The optimal solution obtained is presented in Table 14. W4 and W5 share work at line 1, and W5 works 56% of his/her time at S1 and 44% of the time at S2. W4 works 37% of his/her time at S2 and 63% of the time at S3. W10 is idle 3% of time. Total throughput is 28.4 units per hour. The utilization for each station as well as throughput obtained from each line are presented in rows U and T in Table 14 respectively.

If we allow more than two workers per line/cell, and also require at least one worker per each line/cell while having the same objective function, the optimal throughput and assignment may change. For example, for data set 1 the optimal throughput changed from 28.4/parts per hour to 29.4 parts per hour. The solution with modified constraints is presented in Table 15 and only one worker was assigned to line 3 and line 2, while three workers were assigned to lines 1 and 5. We can also see from the table that W5 and W8 are idle 11% and 5% of the time respectively. By allowing more than two workers per cell (line) the number of workers is not strictly less than the number of stations. In this case we have the same number of workers as the number of stations in some cells (lines).

Table 14: Two workers per line (independent lines)

	Line 1			Line 2			Line 3			Line 4			Line 5			
W/S	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	
1							0.46	0.51	0.03							**1**
2					0.4	0.6										**1**
3													0.62	0.38		**1**
4		0.37	0.63													**1**
5	0.56	0.44														**1**
6														0.38	0.62	**1**
7											0.33	0.67				**1**
8										0.69	0.31					**1**
9				0.61	0.39											**1**
10									0.97							**0.97**
U	**0.56**	**0.81**	**0.63**	**0.61**	**0.79**	**0.6**	**0.46**	**0.51**	**1**	**0.69**	**0.64**	**0.67**	**0.62**	**0.76**	**0.62**	
T	**5.64**			**5.35**			**5.10**			**6.2**			**6.10**			**28.4**

Table 15: Solution with more than two workers per line

	Line 1			Line 2			Line 3			Line 4			Line 5			
W/S	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	
1														1		**1**
2										0.62	0.38					**1**
3													1			**1**
4				0.31	0.34	0.35										**1**
5	0.89															**0.89**
6															1	**1**
7											0.33	0.67				**1**
8			0.95													**0.95**
9		1														**1**
10							0.34	0.34	0.33							**1**
U	**0.89**	**1**	**0.95**	**0.31**	**0.34**	**0.35**	**0.34**	**0.34**	**0.33**	**0.62**	**0.71**	**0.67**	**1**	**1**	**1**	
T	**8.9**			**2.76**			**1.64**			**6.2**			**9.9**			**29.4**

As we discussed earlier, when finished products from several production lines, or manufacturing cells are used simultaneously or products assembled at one line are being used as beginning inventory for another line, we study the system as a set of dependent lines or cells. So, if we run the system as five linked cells or five dependent lines we would obtain the solution presented in Table 16. The final throughput is 5.1 parts/hour if we assume that a part finished at line p is instantaneously available at line $p+1$.

Table 16: Five linked (dependent) lines

	Line 1			Line 2			Line 3			Line 4			Line 5			
W/S	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	
0							0.5	0.5	0.03							**1**
1														0.5	0.5	**1**
2										0.7	0.2					**0.9**
3					0.3	0.6										**0.9**
4	0.5	0.5														**1**
5				0.6	0.4											**1**
6													0.8	0.2		**1**
7		0.3	0.5													**0.8**
8											0.4	0.6				**1**
9									0.97							**0.97**
U	**0.5**	**0.8**	**0.5**	**0.6**	**0.7**	**0.6**	**0.5**	**0.5**	**1**	**0.7**	**0.6**	**0.6**	**0.8**	**0.7**	**0.5**	
T	**5.1**			**5.1**			**5.1**			**5.1**			**5.1**			

4.3.3 Summary

We considered multiple production lines and were able to assign workers to a set of production lines based on the optimality criteria defined. We analyzed two cases. The first is when production lines can be viewed as a set of *l* independent lines, and second when we have *l* dependent lines and we can produce at line *l* at most the output from line *l-1*. This extended formulation which assigns workers to a set of production lines can be helpful when workers are selected to work in different manufacturing cells. We are able to determine the optimal throughput as well as the information regarding the amount of busy/idle time for each worker (this is beneficial in cases where a worker can be assigned to perform some additional tasks during that time).

4.4 SUMMARY

In this chapter we studied a production line consisting of *n* workers and *m* stations. We were able to determine the optimal worker assignment to stations as well as exact portions of work

performed by each worker. We extended our formulation to a set of production lines functioning as an independent or dependent production system. We were able to determine the optimal throughput as well as the optimal assignment of workers to stations.

5.0 DYNAMIC ASSIGNMENT: TWO-CYCLE FORMULATION

In order to model two-cycle behavior, where workers exchange parts at exactly two fixed positions that they visit periodically, we developed a two-cycle math formulation. The modeling assumptions are the same as presented in Chapter Four. We are able to show that depending on the workers' production rates two-cycle behavior may result in optimal throughput.

5.1 TWO-CYCLE FORMULATION

We develop a mixed integer programming formulation that models two-cycle behavior with the objective to maximize throughput and optimally assign workers. Each worker covers a portion of a line during each cycle, and the assumption is that he/she repeats the same portion of work in each cycle for each part produced. Workers perform repetitive work in each cycle and they exchange parts at two locations. There is no blocking and each worker performs the same portion of work on each part. Two workers may share only one station during each cycle (not at the same time) and each worker can only work at adjacent stations. We assume that the raw material (or input) buffer is constantly replenished.

During the first part of the cycle the portion of the time that the workers spend at each station is defined by the x' variables, the second part of the cycle is defined using x'' variables. Thus, in order to model two cycles we introduce variables x'_{ij}, w'_j, z'_{ij} and $idle'_i$ that correspond

to the first cycle and x''_{ij}, w''_j, z''_{ij} and $idle''_i$ that correspond to the second cycle. Detailed definitions regarding these variables are given in the Nomenclature section. Workers movement during the two cycles for a line with two workers and four stations is presented in Figure 4. In this example workers alternate exchanging parts at S2 and S3.

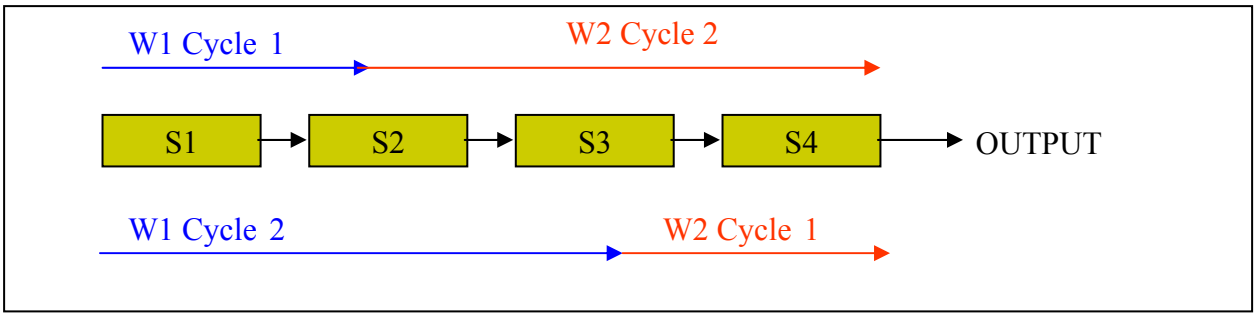


Figure 4: Two-cycle movement for two workers

The two-cycle math formulation is presented below.

Objective function

Maximize *Output*

Constraints

$$w'_j = \sum_{i=1}^{n} x'_{i,j} k_{i,j} \qquad \forall j \qquad (1)$$

$$w''_j = \sum_{i=1}^{n} x''_{i,j} k_{i,j} \qquad \forall j \qquad (2)$$

$$Output \leq w'_j + w''_j \qquad \forall j \qquad (3)$$

$$Output \leq w'_0 \qquad (4)$$

$$Output \leq w''_0 \qquad (5)$$

$$\sum_{i=1}^{n} x'_{i,j} + \sum_{i=1}^{n} x''_{i,j} \leq 1 \qquad \forall j \qquad (6)$$

$$\sum_{j=1}^{m} x'_{i,j} + idle'_i + \sum_{j=1}^{m} x''_{i,j} + idle''_i \leq 1 \qquad \forall i \qquad (7)$$

$$x'_{i,j} \leq z'_{i,j} \qquad \forall i, \forall j \qquad (8)$$

$$x''_{i,j} \leq z''_{i,j} \qquad \forall i, \forall j \qquad (9)$$

$$M x'_{i,j} \geq z'_{i,j} \qquad \forall i, \forall j \qquad (10)$$

$$M x''_{i,j} \geq z''_{i,j} \qquad \forall i, \forall j \qquad (11)$$

$$- abs'_{i,j} \leq z'_{i,j} - z'_{i,j+1} \qquad \forall i \qquad (12)$$

$$abs'_{i,j} \geq z'_{i,j} - z'_{i,j+1} \qquad \forall i \qquad (13)$$

$$z'_{i,1} + \sum_{j=1}^{|J|-1} abs'_{i,j} + z'_{i|J|} \leq 2 \qquad \forall i \qquad (14)$$

$$- abs''_{i,j} \leq z''_{i,j} - z''_{i,j+1} \qquad \forall i \qquad (15)$$

$$abs''_{i,j} \geq z''_{i,j} - z''_{i,j+1} \qquad \forall i \qquad (16)$$

$$z''_{i,1} + \sum_{j=1}^{|J|-1} abs''_{i,j} + z''_{i|J|} \leq 2 \qquad \forall i \qquad (17)$$

$$z'_{i,j} + z'_{i,j+1} + z'_{i',j} + z'_{i',j+1} \leq 3 \qquad \forall i, \forall i' (i' \neq i), j=1,\ldots,m-1 \qquad (18)$$

$$z''_{i,j} + z''_{i,j+1} + z''_{i',j} + z''_{i',j+1} \leq 3 \qquad \forall i, \forall i' (i' \neq i), j=1,\ldots,m-1 \qquad (19)$$

$$z'_{i,j-1} + z'_{i,j} + z'_{i,j+1} + \sum_{i' \neq i} z'_{i',j} \leq 3 \qquad \forall i, \forall i' (i' \neq i), j=1,\ldots,m-1 \qquad (20)$$

$$z''_{i,j-1} + z''_{i,j} + z''_{i,j+1} + \sum_{i' \neq i} z''_{i',j} \leq 3 \qquad \forall i, \forall i' (i' \neq i), j=1,\ldots,m-1 \qquad (21)$$

$$z'_{i,0} - z''_{i,0} = 0 \qquad \forall i \qquad (22)$$

$$z'_{i,m} - z''_{i,m} = 0 \qquad \forall i \qquad (23)$$

$$\sum_{j=1}^{m} x'_{i,j} + idle'_i - \sum_{j=1}^{m} x''_{i+1,,j} - idle''_{i+1} = 0 \qquad \forall i, i+1 \qquad (24)$$

$$\sum_{j=1}^{m} x''_{i,j} + idle''_i - \sum_{j=1}^{m} x'_{i+1,,j} - idle'_{i+1} = 0 \qquad \forall i, i+1 \qquad (25)$$

$$\sum_{i=1}^{n} x'_{i,j} k_{i,j} - \sum_{i=1}^{n} x'_{i,(j+1)} k_{i,(j+1)} = 0 \qquad \forall j, j+1 \qquad (26)$$

$$\sum_{i=1}^{n} x''_{i,j} k_{i,j} - \sum_{i=1}^{n} x''_{i,(j+1)} k_{i,(j+1)} = 0 \qquad \forall j, j+1 \qquad (27)$$

$idle'_i \geq 0$, $idle''_i \geq 0$, $Output \geq 0$, $x'_{ij} \geq 0$, $x''_{ij} \geq 0$, $w'_j \geq 0$, $w''_j \geq 0$, , $abs'_{ij} \geq 0$, $abs''_{ij} \geq 0$, z'_{ij} binary and z''_{ij} binary.

The objective is to maximize throughput from the serial production line during the production horizon. Constraint sets *(1)* and *(2)* determine the output from each station during the first and second cycle respectively. Constraint set *(3)* models the fact that the total output cannot be greater than the output from bottleneck stations during both cycles. Constraint sets *(4)* and *(5)* ensure that the output from the two cycles is the same.

Constraint set *(6)* provides that the maximum work assigned to a station during both cycles cannot be more than one, as we assumed that the duration of the production horizon is either one hour, one day, etc. Consequently constraint set *(7)* provides that each worker works at most the duration of the production horizon, including the time that the worker is idle.

Constraint sets *(8)* to *(21)* are equivalent to the constraints sets *(5)* to *(11)* discussed in Chapter Four. Constraint sets *(8)* to *(21)* include constraints for both cycles. Constraint sets *(22)* and *(23)* ensure that the order of the workers is preserved in the two cycles, so if a worker is first in the order during the first cycle, he/she should be first in the order during the second cycle. Constraint set *(22)* conveys that idea. The equivalent reasoning applies for the last worker in the

order and is expressed by constraint set *(23)*. These constrains are sufficient to preserve the order of two or three workers in two cycles. However, for more than three workers, additional constraints should be included in order to preserve the order of workers.

Constraint sets *(24)* and *(25)* provide that the time of one worker in one cycle equals the time of the other worker in the other cycle in the case of two workers. Constraint sets *(26)* and *(27)* ensure that the output form each station is the same (serial line constraints).

This formulation has one limitation. The possibility of workers exchanging at the first or the last station, which would necessarily induce a worker to be idle, is not modeled. In the future, the formulation could be extended to account for this possibility.

5.2 NUMERICAL EXAMPLES

In this section we present several examples for two worker and four station production lines where the one-cycle solution results in a lower throughput than the two-cycle solution.

Steady-state production rates for data set 1 with two workers and four stations are presented in Table 17. The one-cycle solution is presented in Table 18. The total throughput obtained is 2 parts/hour. We can see from Table 18 that W1 is assigned to S1 and S2, while W2 works at S3 and S4. The two-cycle solution for data set 1 is presented in Table 19.

Table 17: Data set 1 workers production rates

k_{ij}	**S1**	**S2**	**S3**	**S4**
W1	6	3	6	2
W2	2	6	3	6

The total throughput obtained is 2.4 parts/hour which is 20% better than the optimal one-cycle solution. We can see from the tables that W1 works at stations S1, S2 and S3 during the first cycle and only S1 during the second cycle. W2 works at S2, S3 and S4 during the first cycle and at S4 during the second cycle.

Table 18: Data set 1: One-cycle solution

x_{ij}	**S1**	**S2**	**S3**	**S4**
W1	.33	.67		
W2			.67	.33

Table 19: Data set 1: Two-cycle solution

x'_{ij}	**S1**	**S2**	**S3**	**S4**
W1	.2	.4	.2	
W2				.2
x''_{ij}	**S1**	**S2**	**S3**	**S4**
W1	.2			
W2		.2	.4	.2

Steady-state production rates for data set 2 with two workers and four stations are presented in Table 20. The one cycle solution is presented in Table 21. The total throughput obtained is 2.4 parts/hour. We can see from Table 21 that W1 is assigned to S1 and S2, while W2 works at S3 and S4.

Table 20: Data set 2 workers production rates

k_{ij}	**S1**	**S2**	**S3**	**S4**
W1	6	4	4.6	3
W2	3	4.6	4	6

The two-cycle solution for data set 2 is presented in Table 22. The total throughput obtained is 2.5 parts/hour which is higher than the one-cycle solution obtained. We can see from the tables that W1 works at station S1 during the first cycle and S1, S2 and S3 during the second cycle. W2 works at S4 during the first cycle and at S2, S3 and S4 during the second cycle.

Table 21: Data set 2: One-cycle solution

x_{ij}	**S1**	**S2**	**S3**	**S4**
W1	0.4	0.6		
W2			0.6	0.4

Table 22: Data set 2: Two-cycle solution

x'_{ij}	**S1**	**S2**	**S3**	**S4**
W1	0.21			
W2		0.27	0.31	0.21
x''_{ij}	**S1**	**S2**	**S3**	**S4**
W1	0.21	0.31	0.27	
W2				0.21

5.3 SUMMARY

We presented a two-cycle mixed integer formulation that models two-cycle behavior in a dynamic assignment environment. We were able to present examples where a two-cycle solution results in the optimal throughput while the one-cycle solution, for a given set of workers production rates, results in a lower throughout.

This is a significant result as Bartholdi et al. (1999) recognize the existence of two-cycle behavior, but also state that for their problem setting two-cycle behavior results in a suboptimal production rate. Also, for three of more workers other asymptotic modes of behavior or balance

are possible with a suboptimal production rate (such as k-cycles, Bartholdi et al. 1999). As we discussed in Chapter Two, Lim and Yang analyzed systems with discrete stations and determined the average throughput levels obtained based on the order of workers and workers' velocities. However, they assume that workers' velocities are constant along the line and thus if one worker is faster than another then the faster worker dominates the other worker throughout the entire line. They define regions where the system converges to one-cycle exchange positions or two-cycle exchange positions as well as average throughput levels for these regions. They state that the maximum attainable throughput is obtained when workers are ordered from slowest to fastest, the system converges to a fixed point. Our result is unique as we assume that workers' production rates are station dependent and that complete dominance may not be possible.

The two-cycle solution can result in an optimal throughput that is significantly better than the throughput obtained from the one-cycle solution. However, there are situations when a two-cycle solution results in only a slightly better optimal throughput. For example, for data set 1, there is a 20% increase in throughput but for data set 2 the increase in throughput is only 4.1%.

6.0 DYNAMIC ASSIGNMENT WITH LEARNING AND FORGETTING

In Chapters Three, Four and Five we studied a dynamic assignment environment which considers a serial production setting where workers walk to adjacent stations and carry the work towards completion. Workers' production rates are deterministic, and our assumption is that work is divided among the finite number of discrete stations. An example of this dynamic environment is presented in Chapter One (Figure 1). Previously, we studied different assignment policies and their impact on worker utilization, station utilization and portions of the work performed by each worker. We also presented one-cycle and two-cycle mixed integer formulations for serial production lines with n workers and m stations. We were able to obtain optimal worker assignments and determine the optimal throughput. We also looked at the benefits of duplicating stations.

In our previous analysis, we assumed that workers perform at a steady state level, and workers' productivity does not change during the given production horizon. As stated in the Introduction, one of the aims of this research is to study dynamic systems when workers' learning and forgetting is present. In this chapter we discuss situations when workers' productivity changes due to workers' learning and forgetting. In Chapter Two we presented a detailed literature review on the topic of workers' individual learning and forgetting characteristics. We study how changes in the production rates due to learning and forgetting effects impact overall productivity. We are still looking at an environment where the workforce is heterogeneous.

6.1 MOTIVATION TO INTRODUCE LEARNING AND FORGETTING

As we emphasized earlier, a significant amount of research has been done in recent years studying the individual and organizational learning in manufacturing environments. Researchers have discussed the importance of introducing individual learning and forgetting characteristics in situations when short product life cycles and faster product changes are present. Many organizations are restructuring and reorganizing work activities and workers learn new tasks frequently. Workers must learn new skills and processes often in order to keep up with shorter production runs and product cycle times. As a result, worker learning and retention is becoming an increasingly important factor in manufacturing productivity (Nembhard and Uzumeri, 2000b). However, almost all previous work regarding bucket brigade systems assumes that worker speeds are constant over time (Armbruster et al., 2007).

Due to shorter production and frequent process changes worker production rates also change over time in many production environments where bucket brigade system can be a practical alternative. Armbruster et al. (2007) studied the dynamics of the bucket brigade system with worker learning. The authors assumed that n workers are ordered from slowest to fastest with constant speeds along the production line. They considered a production line with fully cross-trained workers and continuous tasks. The authors primarily used an exponential learning model introduced by Mazur and Hastie (1978). One of their conclusions considering situations with all workers learning is that this system will lead to a self balanced production. In situations with one worker learning (it is assumed that the line is already self balanced with two workers working and that the new worker is introduced) the authors define conditions when managerial guidance is needed.

Our assumptions are significantly different than those presented in Armbruster et al. (2007) in two important details. First, we assume that tasks are discrete (or work is divided among discrete stations) and second, workers production rates differ from station to station. We are also looking at how both learning and forgetting impact the total throughput and optimal assignment. We believe that in environments where there are fewer workers than stations, such as bucket brigade systems, forgetting could have significant impacts and should be considered.

6.2 PRODUCTIVITY BASED ON THE LEARNING AND FORGETTING MODEL

Worker productivity levels are determined based on a hyperbolic recency learning and forgetting model. The learning model used in our research was introduced by Mazur and Hastie (1978) and was modified to include the effects of forgetting by Nembhard and Uzumeri (2000). The workers' productivity changes over time and is dependent on the number of times the workers were assigned to each task, and the recency of their experience. Now we present the hyperbolic recency learning and forgetting model. The notation used in this section can be found in the Nomenclature section.

The recency term, R_u, provides a relative measure of how recently an individual's experience was obtained. For each unit of cumulative work u, R_u is determined from the ratio of the average elapsed time to the elapsed time of the most recent unit produced (Shafer et al. 2001), as given in *(1)*.

$$R_{u_{i,j}} = \frac{\sum_{i=1}^{u_{i,j}} \left(st_{u_{i,j}} - st_0 \right)}{u_{i,j} \left(st_{u_{i,j}} - st_0 \right)} \qquad (1)$$

The hyperbolic-recency learning and forgetting model is presented in *(2)* where $p_{i,j}$ represents an approximation of the initial experience of worker *i* doing task *j*, $r_{i,j}$ is a learning or shape parameter based on the same units as *u* and $\alpha_{i,j}$ is the forgetting rate of worker *i* doing task *j*. The learning rate parameter $r_{i,j}$ in a hyperbolic-recency model represents the cumulative production required to get halfway to $k_{i,j}$ Thus, slower learning is represented by a larger value of $r_{i,j}$. Parameters $p_{i,j}$ and $\alpha_{i,j}$ are nonnegative and the learning rate r_{ij} can be negative. This learning model is capable of describing both positive and negative learning episodes (Nembhard and Uzumeri 2000), and for the negative learning case, $r_{ij} < 0$, in order to avoid division by 0, it is necessary that $p_{i,j} + r_{i,j} > 0$. However, this research considers positive learning only.

$$y_u = k_{i,j}\left(\frac{u_{i,j}R_{u_{i,j}}^{\alpha_{i,j}} + p_{i,j}}{u_{i,j}R_{u_{i,j}}^{\alpha} + p_{i,j} + r_{i,j}}\right) \qquad (2)$$

More complex tasks or hard tasks result in slower learning because workers need additional practice to become proficient (Nembhard, 2000a). The forgetting rate, $\alpha_{i,j}$, represents the extent to which worker *i* forgets task *j* following a break. The literature on the effects of task complexity on individual forgetting rates concludes that as task complexity increases, the individual forgetting rate increases (Nembhard, 2000b). Initial expertise is estimated based on initial performance level. The literature on the effects of task complexity on initial performance level, or the fitted initial expertise $p_{i,j}$, suggests that a higher task complexity results in a lower mean initial expertise (Nembhard, 2000b). In constraint *(1)* recency is based on cumulative units of work.

Modeling assumptions are the same as presented in section 3.1. As discussed in section 3.1, the tasks performed at each station are different. Consequently, each worker has distinctive learning-forgetting parameters and learning-forgetting curves for each station at which he/she performs. A worker's productivity rate on each station varies over time due to the individual's learning-forgetting characteristics.

6.3 SIMULATION: COMPARISONS WITH STEADY STATE ASSIGNMENTS

A C++ code for the bucket brigade application has been written in order to simulate a production line that operates as a bucket brigade system. The code's algorithm and flow chart are presented in Appendix F. The final throughput is calculated for a line with m stations and n workers. Different order and initial positioning of workers were tested. No passing or switching of workers is allowed. We compared outputs when workers perform at steady-state production levels with the outputs obtained when productivity changes with each part/unit produced due to learning and forgetting effects.

The data used in this work is based on 124 workers that were observed at three serial production lines and 24 test stations doing an industrial manufacturing task. Data were collected from the final test and inspection station of an assembly line that produces car radios. The learning and forgetting parameters were individually fit from the empirical data and for detailed information see Shafer et al. (2001). For some instances tested (when workers produce at their steady state levels) $k_{i,j}$ values were randomly generated.

We tested instances with two to eight workers and two to sixteen stations. The total throughput calculated was based on the number of completed parts (units). We looked at different durations of 1, 2, 3, 4, 10, 20, 30, 40, 50, 100 and 1000 time units. We enumerated all possible assignments for both steady state production rates and production rates when learning and forgetting is present. Our objective is to determine the position of workers which yields the highest throughput.

For a line with six workers and twelve stations, steady state production rates are given in Table 23. Learning and forgetting parameters p, r and α, for the same set of workers/stations, are given in Table 71 (Appendix F).

Table 23: Data Set: Steady-state production values

k_{ij}	1	2	3	4	5	6	7	8	9	10	11	12
1	25.02	28.29	29.25	29.16	28.22	35.79	33.1	35.76	37.46	30.57	27.48	37.92
2	25.93	32.85	34.88	26.1	35.92	32.34	28.89	34.03	35.28	30.86	25.53	31.49
3	38.5	25.79	37.48	33.65	28.18	37.77	30.24	39.3	29.52	38.83	27.09	32.9
4	38.63	34.56	36.99	37.79	37.38	36.4	27.18	35.28	35.34	27.56	34.61	37.11
5	26.86	33.81	38.1	36.04	28.61	27.62	30.19	31.99	25.69	31.81	30.67	38.43
6	36.67	31.23	30.82	26.52	35.57	28.65	30.13	26.78	31.33	33.42	34.79	30.59

We present the optimal throughput for a line with six workers and twelve stations when learning and forgetting is present. We can see from Table 24 that the optimal assignment of workers, based on the highest throughput obtained, changes with the duration of the total production time or the total production horizon. If we looked at the highest levels of throughput obtained for the production horizon length of 100 time units we would assume that the optimal assignment is Position 3 (W1 at S9, W2 at S5, W3 at S7, W4 at S11, W5 at S3 and W6 at S1). However, if the total duration of production or the total production time is 20 time units, the optimal starting order or assignment would be Position 2.

Table 24: Throughput obtained with learning and forgetting

Time	Position 4: W1S7 W2S3 W3S11 W4S9 W5S1 W6S5	Position 2: W1S7 W2S1 W3S11 W4S8 W5S3 W6S5	Position 3: W1S9 W2S5 W3S7 W4S11 W5S3 W6S1
1	3	**12**	3
2	10	**25**	8
3	23	**39**	18
4	36	**54**	32
5	50	**69**	45
6	65	**84**	60
7	80	**100**	76
8	96	**115**	91
9	112	**130**	107
10	128	**145**	124
20	292	**300**	292
30	458	456	**464**
40	626	613	**636**
50	795	771	**808**
100	1641	1565	**1672**

The results obtained for the six worker twelve station production line with workers producing at steady state levels are given in Table 25. The optimal solution or the highest levels of throughput are obtained when workers are assigned according to Position 3 (W1 at S9, W2 at S5, W3 at S7, W4 at S11, W5 at S3 and W6 at S1). We also presented throughput levels for Position 2 (W1 at S7, W2 at S1, W3 at S11, W4 at S8, W5 at S3 and W6 at S5) and Position 4 (W1 at S7, W2 at S3, W3 at S11, W4 at S9, W5 at S1 and W6 at S5). When workers produce at steady state levels Positions 4 and 2 result in lower throughput.

The average throughput (the total throughput obtained divided by the number of time units) for Positions 2, 3, and 4 when learning and forgetting is included is presented in Figure 5. It can be seen that Position 2 yields the highest throughput for the shorter production horizons. We also wanted to note that Position 4 yields higher throughput than Position 3 for production horizons shorter than 20 time units.

Table 25: Throughput for three different workers positions: Steady state

Time	Position 4: W1S7 W2S3 W3S11 W4S9 W5S1 W6S5	Position 2: W1S7 W2S1 W3S11 W4S8 W5S3 W6S5	Position 3: W1S9 W2S5 W3S7 W4S11 W5S3 W6S1
1	17	16	**17**
2	34	32	**35**
3	50	48	**52**
4	66	64	**69**
5	83	80	**87**
6	98	96	**104**
7	116	112	**121**
8	131	128	**139**
9	148	144	**156**
10	164	160	**173**
20	326	320	**346**
30	489	480	**520**
40	653	639	**693**
50	815	799	**866**
100	1629	1598	**1732**

However, ordering workers according to Position 3 would result in the highest throughput for longer production runs. It is interesting to look at the average throughput obtained by each assignment or position of workers as the break point when throughput obtained by steady state productivity becomes higher is easily determined. In, the break point is around twenty time units.

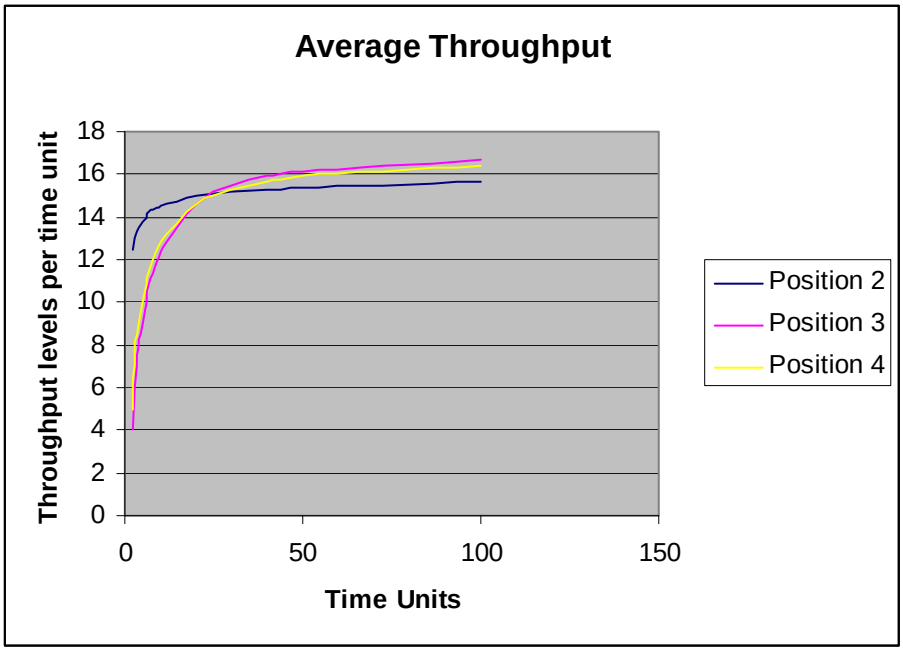


Figure 5: Average throughput with learning and forgetting

6.4 SUMMARY

The example presented confirms our intuition that for each set of workers where individual workers' learning and forgetting characteristics are incorporated, the optimal assignment depends on the total production time. Based on the example presented, as well as other examples studied, we can conclude that for each set of workers we should focus on determining a break point where the optimal assignment changes due to the maximum throughput possible. We can see from Figure 5 that the break point regarding average throughput occurs around twenty time units for the data set presented. For a production horizon that is less than twenty time units, Position 2 yields the optimal throughout. After twenty time units Position 3 is optimal.

Thus, when production runs are shorter and depending on workers' previous experience, steady state production rate, and learning and forgetting parameters, the optimal throughput can be significantly different than throughput obtained based on steady state production only. Sensitivity analysis can be performed as well as more testing in order to determine the magnitude of importance, but when tasks are harder and it takes longer to achieve steady state the impact can be significant. It is very applicable to include forgetting in environments when there are less workers than tasks, like environments considered in this research.

7.0 FIXED ASSIGNMENT: MIP AND MINLP

This part of our research focuses on a fixed assignment environment as discussed in Chapter One. This is an extension of the work presented in Nembhard and Norman (2002). The authors developed a worker-task assignment model where work is performed on sequential stations and there are intermediate buffers.

7.1 MODELING ASSUMPTIONS

We now present our modeling assumptions for a fixed assignment environment. Due to unique machinery only one task is performed at the station and the production horizon is divided into time periods. There is no collaboration among workers during the given time period. The authors considered individual worker learning and forgetting. The formulation was based on log-linear learning and forgetting. Leopairote (2004) analyzed the same model where learning and forgetting was modeled using a hyperbolic-recency learning-forgetting model.

We consider a serial-production line with n workers, l time periods and m stations. Workers perform at steady-state production rates, processing times are deterministic and each part needs processing on the same sequence of stations (stations 1 to m). We are looking at a

discrete part production system where completed parts are put into a buffer which supplies the next station. This production environment is presented in Figure 6.

There are several major differences between the production environment presented in previous chapters and the production environment considered in this analysis. The production horizon is now divided into time periods, thus we look at the assignments per worker/per station /per time period. In the previous analysis we looked at worker/station assignments and different assignment options, so we only considered one worker order as the assumption was that the order of the workers is preserved. In this case workers are assigned every period so the order of workers may change several times during the production horizon. Another important difference is the presence of buffers between stations.

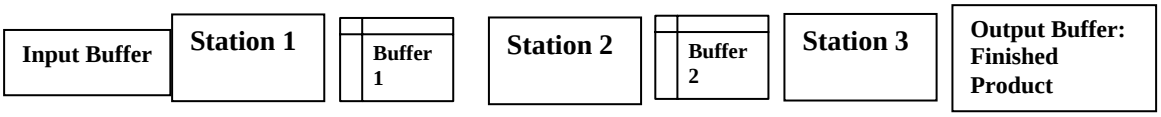


Figure 6: Fixed Assignment Environment

There are no restrictions on the assignments and the work content at the stations need not be identical. At most one worker can work at a station at any given time due to the characteristics of the tasks or equipment requirements. Only one part can be processed at the station. Tasks at stations have different complexity levels and worker production rates are task dependent. One worker can be faster at one station than another worker, but slower at another station. Buffer inventory is unconstrained and permitted between the stations. We looked at situations when the starting intermediate buffer inventory is zero, 10, 20, or 50 parts. We also looked at some situations (see Chapter Eight) when starting and ending buffer inventory is required to be identical. The raw material input to the first station is constantly restocked. A worker at a

downstream station can produce the minimum of the buffer inventory plus the output from the previous station and his/her productivity rate during the given time period. We assume that units produced at station *j* during time period *l* are instantaneously available at station *j+1*. Also, the times that workers need to set aside the finished part and obtain the new part are significantly smaller than production times thus it is safe to assume that these occur instantaneously. This also implies that the positions of workers on the line change instantaneously at the end of each time period. The total number of workers assigned to a production line remains unchanged over time.

7.2 MODEL FORMULATION: MIP

A worker assignment problem is considered where workers are assigned on a serial production line. This formulation was developed by Nembhard and Norman (2002). Initially, it is assumed that workers produce at their steady-state productivity rate. A mixed integer programming (MIP) formulation of the problem is now presented. The objective is to assign *n* workers on a serial production line consisting of *m* sequential stations (tasks) in order to maximize throughput during the total production time which is divided into *l* time periods. The notation used in this section can be found in the Nomenclature section.

Objective function

$$\textbf{\textit{Maximize}} \sum_{i=1}^{n} \sum_{t=1}^{p} O_{i,m,t}$$

Constraints

$$\sum_{i=1}^{n} q_{i,j,t} \leq 1 \qquad \forall j, \ \forall t \qquad (1)$$

$$\sum_{j=1}^{m} q_{i,j,t} \leq 1 \qquad \forall i,\ \forall t \qquad (2)$$

$$O_{i,j,t} \leq q_{i,j,t}\, k_{i,j} \qquad \forall i,\ \forall j,\ \forall t \qquad (3)$$

$$B_{1,1} = BI_1 - \sum_{i=1}^{n} O_{i,1,1} \qquad (4)$$

$$B_{j,1} = BI_j + \sum_{i=1}^{n} O_{i,j-1,1} - \sum_{i=1}^{n} O_{i,j,1} \qquad j=2,\ldots,m \qquad (5)$$

$$B_{1,t} = B_{1,t-1} - \sum_{i=1}^{n} O_{i,1,t} \qquad t=2,\ldots,l \qquad (6)$$

$$B_{j,t} = B_{j,t-1} + \sum_{i=1}^{n} O_{i,j-1,1} - \sum_{i=1}^{n} O_{i,j,t} \qquad j=2,\ldots,m \;\; t=2,\ldots l \qquad (7)$$

q_{ijt} binary, $O_{ijt} \geq 0$, $B_{j,t} \geq 0$ *and* $pr_{i,j,t} \geq 0$.

The objective function maximizes the total number of finished units from the last workstation on the serial production line. Constraints *(1)* and *(2)* provide that each workstation has at most one worker assigned to it and that each worker is assigned to at most one workstation. Constraint set *(3)* insures that the output from the station can be at most the productivity of the assigned worker. Constraint sets *(4) to (7)* are inventory balance constraints which determine the number of available units at each workstation. For example a worker can produce the minimum of his/her production rate and the number of available units. Note that B_{jt} represents the number of units in the buffer immediately preceding station *j* at the end of period *t*.

7.3 MODEL FORMULATION: MINLP

The assignment of workers to tasks based on individual learning and forgetting characteristics has received very little attention in the literature. Research in the area of worker-task assignment generally assumes steady-state productivity. One of the goals in this research is to show the importance of introducing learning and forgetting in these types of problems. As we discussed in previous chapters, due to frequent changes in product demand, workers need to learn and master new tasks more often than before. The assumption that workers produce at steady state is not as justified as the length of total production for some products becomes shorter.

We now present the extended MIP formulation. Leopairote (2004) modified the MIP formulation developed by Nembhard and Norman (2002) to include the effects of workers learning and forgetting. Worker productivity levels are determined based on the hyperbolic recency learning and forgetting model introduced in 6.2. Workers' productivity is now included in the MIP formulation. The worker-assignment model is formulated as a mixed integer nonlinear programming model (MINLP).

Modeling assumptions are the same as presented in section 7.1. As discussed in section 7.1, the operations performed at each station are station or task dependent. Consequently, each worker has distinctive learning-forgetting parameters and learning-forgetting curves for each station at which he/she performs. The notation used in this section can be found in the Nomenclature section.

The recency term, $R_{u,}$, introduced in *(1)* in 6.2, is again presented in *(8)*.

$$R_{u_{i,j}} = \frac{\sum_{i=1}^{u_{i,j}} \left(st_{u_{i,j}} - st_0 \right)}{u_{i,j} \left(st_{u_{i,j}} - st_0 \right)} \qquad (8)$$

The hyperbolic-recency learning and forgetting model is presented in *(9)*.

$$y_u = k_{i,j} \left(\frac{u_{i,j} R_{u_{i,j}}^{\alpha_{i,j}} + p_{i,j}}{u_{i,j} R_{u_{i,j}}^{\alpha} + p_{i,j} + r_{i,j}} \right) \qquad (9)$$

In order to incorporate the recency measure given in *(8)* and the hyperbolic-recency learning and forgetting model given in *(9)* into the math programming formulation, the following constraints were added into the formulation (Leopairote 2004):

$$pr_{i,j,t} = k_{i,j} \left[\frac{\left(\sum_{s=1}^{t-1} O_{i,j,s} \right) R_{i,j,t}^{\alpha_{i,j}} + p_{i,j}}{\left(\sum_{s=1}^{t-1} O_{i,j,s} \right) R_{i,j,t}^{\alpha_{i,j}} + p_{i,j} + r_{i,j}} \right] \qquad (10)$$

and

$$R_{i,j,t} = \frac{\sum_{s=1}^{t-1} (s - t_0) q_{i,j,t}}{\left(\sum_{s=1}^{t-1} x_{i,j,s} \right) (t - t_0)} \qquad (11).$$

In constraints *(10)* and *(11)* recency is based on time periods rather than cumulative units of work, because it would be extremely difficult to incorporate the unit based constraints into the math programming formulation. Our assumption is that $t_0=0$.

Constraint (3) is modified to include the variable productivity rate:

$$O_{i,j,t} \le q_{i,j,t}\, pr_{i,,j,t} \qquad \forall i,\ \forall j,\ \forall t \qquad (3)$$

All other constraints, *(1)* to *(7)*, and the objective function are assumed to be the same as presented in the MIP formulation (see Section 7.1).

7.4 SOLUTION METHODOLOGIES

7.4.1 Branch and bound: MIP

The worker assignment problem was solved to optimality for smaller and moderate size instances by the "branch and bound" of CPLEX. Branch and bound is a general algorithm for finding optimal solutions of various optimization problems, particularly in discrete and combinatorial optimization. The algorithm is nonheuristic in nature and consists of implicit enumeration of all possible solutions where large subsets of not promising candidates are discarded, by using upper and lower estimated bounds. Branch and bound algorithms can be slow in some cases. In the worst case they require effort that grows exponentially with problem size, but in some cases the algorithm converges much faster.

Several situations occurred when running "harder" and "larger" instances in CPLEX that were not solved in "reasonable" times. We either stopped the runs after recording the best obtained (feasible) solution or error termination occurred due to insufficient memory. We tried to set the CPLEX parameters so that memory is conserved by selecting strong branching or the node selection strategy to best estimate, or to do depth-first searches. Strong branching requires substantial computational effort at each node to determine the best branching variable (ILOG Reference Manual (2003)). As a result, it generates fewer nodes and thus makes less overall demand on memory. However, for "hard" instances we didn't see much improvement (runs were

still terminated due to insufficient memory). We also tried tuning the parameters with the help of the tool STOP (Selection Tool for Optimization Parameters) developed by Brady Hunsaker, Mustafa Baz, Paul Brooks, and Abhijit Gosavi (Baz et al. (2007)). STOP is a tool to help find good parameter settings for a program for a set of instances. After using STOP we reduced running times on easier instances but we didn't resolve the memory issues.

Thus, due to the structure and complexity of the defined problem and the size of the search space, even moderate size instances are difficult to solve. A "hard" instance with two workers, four stations and twenty periods or eight workers, eight tasks and twenty periods could not be solved. We define a "hard" instance as an instance that cannot be straightforwardly solved. Also, an "easy" instance is defined as an instance that can be easily solved. However, we could solve an "easy" instance with eight workers, eight tasks and twenty periods.

7.4.2 MINLP Solvers

The General Algebraic Modeling System (GAMS) is specifically designed for modeling linear, nonlinear and mixed integer optimization problems. MINLP small instances with two workers and up to four stations were solved in GAMS. We used GAMS solvers: SBB and DICOPT. SBB is a GAMS solver for MINLP models. It is based on a combination of the standard Branch and Bound method known from Mixed Integer Linear Programming and some of the standard Non Linear Programming solvers already supported by GAMS. During the solution process SBB solves a number of relaxed MINLP models with tighter and tighter bounds on some of the integer variables. The solutions to these sub-models are assumed to provide valid bounds on the objective function. SBB will find the global optimum if the underlying RMINLP model is convex. If the sub-models are not convex then some sub-models may be solved to a local

optimum that is not global, and they may terminate in a locally infeasible point even if feasible solutions exist.

DICOPT (DIscrete and Continuous OPTimizer) is a framework for solving MINLP models. DICOPT is an extension of the outer-approximation algorithm with equality relaxation strategies. DICOPT solves a series of NLP and MIP sub-problems using any solver supported by GAMS. Although, the algorithm has provisions to handle non-convexities, it does not always find the global solution.

7.4.3 Pairwise-Exchange Heuristic: MIP and MINLP

In order to solve larger and "harder" problem instances, we looked at different heuristic algorithms. We developed a heuristic algorithm that is based on an improvement search concept. The main idea of the proposed local improvement algorithm is based upon a pairwise exchange of worker-station assignments. Based on the mechanics of the algorithm, it can be considered an improvement algorithm. The algorithm starts from a random feasible solution. The algorithm searches for the local optimum, and the solution obtained is dependent on the initial solution because the search proceeds by examining adjacent feasible solutions. Random assignments were used as initial assignments. A pseudocode of the algorithm is presented in Appendix C. Selected results are given in Appendix E.

7.4.4 Simulated Annealing Algorithm: MIP and MINLP

Due to the difficulty of using math programming to obtain solutions to the MIP metaheuristics methods are investigated. Both, Simulated Annealing and Tabu Search are very suitable for combinatorial optimization problems (Reeves 1993) and have been widely applied. We will

concentrate on the application of a Simulated Annealing algorithm. Some of the recent relevant literature is presented in Chapter Two.

Simulated Annealing is a probabilistic heuristic approach for global optimization. The objective is to locate a good approximation to the global optimum of a given function in a large search space. It was independently invented by S. Kirkpatrick, C. D. Gelatt and M. P. Vecchi in 1983, and by V. Cerny in 1985. Simulated Annealing is very appropriate for problems with a large feasible region and many locally optimal solutions (Reeves 1993). The algorithm's advantage over many other improvement methods is a capability to avoid becoming trapped at local optima. The algorithm uses a random search which not only accepts changes that improve the objective function, but also some changes that worsen it.

As its name implies, Simulated Annealing uses an analogy between the annealing process and the search for an optimum in a more general system. The current state of the thermodynamic system corresponds to the current solution of the combinatorial problem, the energy equation for the thermodynamic system corresponds to the objective function, and ground state is analogous to the global minimum (or maximum) (Reeves 1993). The major difficulty in implementation of the algorithm is that there is no obvious analogy for the temperature T with respect to a free parameter in the combinatorial problem. Furthermore, avoiding entrapment in local minima is dependent on the "annealing schedule", the choice of initial temperature, the number of iterations performed at each temperature, and how much the temperature is decremented at each step as cooling proceeds (Reeves 1993).

An initial solution is defined as a feasible assignment of workers to tasks. The move operator is defined as the exchange of two workers' assignments during a given time period. The neighborhood of a solution is defined as all pairwise exchanges of workers assignments. The

annealing schedule is determined empirically. Several temperature reduction functions were tested. Notation, together with the assumptions and the pseudocode of the algorithm are presented in Appendix D.

7.5 NUMERICAL RESULTS

7.5.1 Data sources

The data used in this work is based on 124 workers that were observed at three serial production lines and 24 test stations doing an industrial manufacturing task. The learning and forgetting parameters were individually fit from the empirical data. For detailed information see Shafer et al. (2001). We also randomly generated workers' production rates for some instances. Results obtained by the branch and bound of CPLEX for the MIP and the simulated annealing algorithm for both the MIP and the MINLP are presented below.

7.5.2 Numerical Results: MIP

The Simulated Annealing algorithm was tested on different sets of randomly generated initial assignments. We tested instances with two to eight workers and four to eight stations. The length of the production horizons varied from four to twenty four time periods. Ten random seeds were generated and all combinations were tested for these seed values. Initial temperature levels were determined based on initial testing and the average improvement of accepted solutions or moves. Changes in the objective function were between 4 and 12, thus these values were tested as the initial temperature levels. Two different annealing schedules were tested with α=0.95 and α=0.99. The number of iterations at each temperature level was either 1000, 1500 or 2000. The stopping condition at lower temperature levels was 800, 900 or 1000 iterations without

improvement. A summary of the results is presented below for selected instances. Results for additional instances are presented in Appendix F.

We present results obtained for the data sets tested with two workers and four stations.

Table 26: Data set 1: Workers' Production Rates

k_{ij}	S1	S2	S3	S4
W1	7.01	6.3	5.05	7.55
W2	9.75	8.94	6.59	9.75

We compared the results obtained and the solution times for eight, twelve, and sixteen time periods for data set 1 (production rates given in Table 26). The results obtained by CPLEX for data set 1 are summarized in Table 27.

Table 27: Data set 1: CPLEX Results

Data set 1	Periods	Output	Solution times (seconds)
1-8	8	29.25	5.13
1-12	12	43.48	40938.22
1-16	16	59.74	11491.48*

*Error termination

For data set 1 and 8 time periods, the simulated annealing algorithm reached the optimal objective value, or optimal throughput of 29.25 parts, 100% of the time. We tested four initial random assignments for ten randomly generated seeds. The average solution time was 3.312 seconds (CPU time). The following settings were used:

- *tpr* (starting temperature) = 4;
- α (a temperature reduction function parameter) = 0.99; and annealing schedule *tpr* = α*tpr*;

- maximum number of iterations at each temperature level is 1000 and number of moves without improvement is 800;
- *tpr* > 0.01 stopping condition and the last temperature level.

Results for data set 1 and 12 time periods are presented in Table 28. The optimal solution was obtained every time and the average solution times are around seven seconds which is much shorter than CPLEX's solution time of 40938 seconds.

The solution time for data set 1 and 16 time periods solved by the branch and bound of CPLEX was 11491.48* seconds even though we applied parameter settings that we believed were the best possible. The best feasible solution obtained resulted in the objective function value of 59.74 parts. We were able to obtain this solution by Simulated Annealing in 6.5 seconds on the average. Other solutions were within 0.4 percent of the optimal objective value.

Table 28: Data set 1-12 time periods: Simulated Annealing Results

	Assignment 1		**Assignment 2**		**Assignment 3**	
Seed	**Solution**	**CPU Time**	**Solution**	**CPU Time**	**Solution**	**CPU Time**
222	**43.48**	7.45	**43.48**	7.73	**43.48**	7.97
44	**43.48**	7.44	**43.48**	7.53	**43.48**	7.56
777	**43.48**	7.25	**43.48**	7.47	**43.48**	7.61
653	**43.48**	7.44	**43.48**	7.55	**43.48**	7.38
21	**43.48**	7.73	**43.48**	7.53	**43.48**	7.45
269	**43.48**	7.75	**43.48**	7.89	**43.48**	7.50
17	**43.48**	7.69	**43.48**	7.33	**43.48**	7.22
140	**43.48**	7.75	**43.48**	7.36	**43.48**	7.38
33	**43.48**	7.52	**43.48**	7.39	**43.48**	7.34
178	**43.48**	7.70	**43.48**	7.25	**43.48**	7.53
MAX	43.48	7.75	43.48	7.89	43.48	7.97
MIN	43.48	7.25	43.48	7.25	43.48	7.22
AVERAGE	**43.48**	**7.57**	**43.48**	**7.50**	**43.48**	**7.49**

Table 29: Data set 1-16 time periods: Simulated Annealing Results

	Assignment 1		Assignment 2		Assignment 3	
Seed	**Solution**	**CPU Time**	**Solution**	**CPU Time**	**Solution**	**CPU Time**
222	59.45	6.28	59.45	6.16	59.45	6.19
44	59.45	6.59	**59.74**	6.22	59.45	6.23
777	59.45	6.36	59.45	6.19	**59.74**	6.22
653	59.45	6.38	59.45	6.19	59.45	11.45
21	59.45	6.42	59.45	6.19	59.45	6.20
269	**59.74**	6.33	59.45	6.17	59.45	6.20
17	59.45	6.24	**59.74**	6.19	59.45	6.19
140	59.45	6.73	59.45	7.36	59.45	6.28
33	59.45	6.17	59.45	6.22	59.45	6.75
178	**59.74**	6.20	**59.74**	6.22	59.45	6.41
MAX	59.74	6.73	59.74	7.36	59.45	11.45
MIN	59.45	6.17	59.45	6.16	59.45	6.19
AVERAGE	**59.508**	**6.37**	**59.54**	**6.3095**	**59.48**	**6.81**

Workers' production rates for data set 2 are given in Table 30. CPLEX results for data set 2 and 8, 12 or 16 time periods are given in Table 31.

Table 30: Data set 2: Workers' production rates

k_{ij}	S1	S2	S3	S4
W1	**8**	**9**	**5**	**7**
W2	**5**	**6**	**7**	**8**

Table 31: Data set 2: CPLEX Results

Data set 2	**Periods**	**Output**	**Solution times (seconds)**
2-8	8	28	2.02
2-12	12	42	4389
2-16	16	60	13846

We obtained an optimal solution 100% of the time for data set 2 for both 8 and 12 time periods. Simulated Annealing results for data set 2 and 16 time periods are presented in Table 32. The best feasible solution obtained resulted in an objective function value of 60 parts. We were

able to obtain this solution by Simulated Annealing in 6.45 seconds on the average. Other solutions were within 3.3 percent of the optimal objective value.

Table 32: Data set 2 – 16 time periods: Simulated Annealing Results

	Assignment 1		**Assignment 2**		**Assignment 3**	
Seed	**Solution**	**CPU Time**	**Solution**	**CPU Time**	**Solution**	**CPU Time**
222	**60**	6.5	58	6.31	58	6.28
44	**60**	6.58	**60**	6.66	**60**	6.31
777	59	6.31	59	6.30	58	6.28
653	58	6.33	**60**	6.31	**60**	6.28
21	59	6.30	59	6.31	**60**	6.28
269	58	6.33	59	6.34	59	6.22
17	58	6.30	**60**	6.28	**60**	6.34
140	58	6.36	59	6.30	59	6.28
33	58	6.33	**60**	6.30	58	6.27
178	**60**	6.33	59	6.30	**60**	6.27
MAX	60	6.58	60	6.66	**60**	6.34
MIN	58	6.30	58	6.28	58	6.22
AVERAGE	**58.8**	**6.37**	**59.3**	**6.34**	**59.2**	**6.28**

We also tested larger instances with 2 to 8 workers and 3 to 16 stations. Results for several larger instances are summarized in Table 33, together with CPLEX results. The following three instances are presented: Instance L1 (large 1) with eight workers, eight stations and twelve time periods, Instance L2 with eight workers, eight stations and twelve time periods and Instance L3 with eight workers, eight stations and twenty time periods. For instance L1 $k_{i,j}$ values were randomly generated. The optimal solution for this instance can be obtained by CPLEX in approximately 3 seconds, thus it is considered an "easy" instance. Instances L2 and L3 are considered as "hard" instances. For instance L3, the upper bound is 604.4 (LP Relaxation), thus the solutions obtained are close to the upper bound (1.5%).

Table 33: Larger Instances: Comparison of Results

CPLEX			Simulated Annealing		
Instance	**Best Solution**	**CPU time**	**Instance**	**Best Solution**	**CPU time**
1	**408**	**3.2**	**1**	**408**	**2.7**
2	**356**	**3775**	**2**	**356**	**3.1**
3	**595****	**34023**	**3**	**595**	**31.2**

**Error termination

Simulated Annealing provided solutions faster than CPLEX for all instances presented and based on the quality of the solutions found seems to be an appropriate solution methodology for this problem. For instance L1, simulated annealing reached the best solution 100% of the time. For Instance L2, Simulated Annealing reached the best solution 30 to 50% of the time and the other solutions were within 1.1% of the best solution. For Instance L3, Simulated Annealing reached the best solution 10 to 30% of the time and the other solutions were within 0.3% of the best solution. As the problem size increases, the search space is very large and Simulated Annealing has the potential of performing better than the "branch and bound" of CPLEX. Based on our study, solutions times needed for Simulated Annealing Algorithms are significantly lower. It is important to note that for Instance L1 $k_{i,j}$ values are randomly generated based on the steady state levels given in Shafer et al. (2001) and Instance 1 is considered an "easy" instance. Also, Instances L2 and L3 are considered "hard" instances (as defined in 7.4.1). All runs and data sets are attached in Appendix F.

7.5.3 Pairwise-Exchange Heuristic: MINLP

The algorithm was tested on problems with four different sets of k, p, r and α values. These were small problems with three workers, three stations and eight time periods, and they were solved to optimality. The optimal solution was compared with the solution obtained from the heuristic

algorithm. For buffer inventory levels of 0, 10, 20, 50 and 100 units and all data sets tested, the optimal solution was obtained with the heuristic algorithm. The results are presented in Table 34.

In order to show the importance of introducing learning and forgetting into these types of problems two additional values were calculated for all data sets. The first is the optimal throughput provided by the optimal assignment based on the steady state productivity rates. The second value represents the throughput of the steady state optimal assignments evaluated considering learning and forgetting. The optimal throughput obtained with the steady state productivity rates is much higher than the actual output that is found when learning and forgetting are considered. Across the problems the actual output is about 50% less than that found by simply using the steady state productivity values. This is important because if planning was based on attaining production assuming steady state productivity then there would be significant errors in the planning process. A second important point to notice is that the output found by properly evaluating the assignment that is found only considering the steady state productivity values is significantly less than the optimal assignment that is found when learning and forgetting are considered. The differences range from only a few percent to almost 25 percent. Thus, if workers are assigned "optimally" based on their steady state productivity this, in fact, turns out to be a suboptimal assignment. These values are presented in Table 34 for four data sets.

For larger problems such as ones containing eight workers, eight stations and twenty time periods we compared the heuristic solution with that found by an enumeration algorithm (lower bound) and an LP relaxation solution (upper bound). The enumeration algorithm exhaustively searches through all fixed assignments (no rotation) and calculates the total throughput. These

solutions are considered as lower bounds as the flow of the line could improve under different rotation polices.

Table 34: Pairwise-Exchange Heuristic for small instances

	inv 0	**inv 10**	**inv 20**	**inv 50**	**inv 100**
Data set 1: Optimal solution with steady-state productivity	226	238	248	278	304
Assignment evaluated with learning and forgetting	132	135	143	155	162
Heuristic solution	144	156	162	168	168
Optimal solution	**144**	156	162	168	168
Data set 2: Optimal solution with steady-state productivity	224	224	224	224	224
Assignment evaluated with learning and forgetting	112	112	112	112	112
Heuristic solution	150	160	160	160	160
Optimal solution	**150**	160	160	160	160
Data set 3: Optimal solution with steady-state productivity	200	215	216	216	216
Assignment evaluated with learning and forgetting	74	94	107	134	134
Heuristic solution	96	104	124	152	152
Optimal solution	**96**	104	124	152	152
Data set 4: Optimal solution with steady-state productivity	242	246	248	248	248
Assignment evaluated with learning and forgetting	135	152	144	144	144
Heuristic solution	144	164	176	176	176
Optimal solution	**144**	164	176	176	176

Unfortunately, due to the complexity of the MINLP formulation, simply relaxing the integrality constraints of the MINLP results in a problem that is still nontrivial to solve. Thus, we created an alternative formulation of the problem. First, we calculated the total number of units produced by assigning each worker to one station (no rotation) during the total production period, using the learning and forgetting curves. Then, the average individual productivity per period was calculated as the total number of units produced divided by the number of production periods. The resulting data was used in a formulation that assigned one worker to one task during

each of the twenty time periods. Thus, the non-linear constraints were removed because only the average productivity was used. Unfortunately, due to the difficulty of solving this integer program we had to look at the solution of the relaxation for an upper bound on total production. The solution obtained by this LP relaxation can only give us an insight into the possible throughput through the serial line and represents a high, and in most cases, non-attainable upper bound on the problem. The LP is formulated and solved using the GAMS optimization software. The results for three data sets with eight workers, eight stations and twenty time periods are presented in Table 36. The k_{ij}, r_{ij}, p_{ij} and α_{ij} for the three data sets are given in Appendix F.

7.5.4 Simulated Annealing Algorithm: MINLP

The algorithm was tested on problems with four different sets of k, p, r and α values. These were small problems with three workers, three stations and eight time periods, and they were solved to optimality. The optimal solution was compared with the solution obtained from the Simulated Annealing algorithm. The algorithm was tested for five randomly generated initial assignments and 10 random seed values, as well as different sets of initial temperature values, temperature reduction function parameters and number of iterations. The Simulated Annealing algorithm found an optimal solution 100% of time for each initial assignment. The results are presented in Table 35. Detailed results as well as data sets are presented in Appendix F.

Table 35: Simulated Annealing Results for "small" instances

	Data set 1	**Data set 2**	**Data set 3**	**Data set 4**
Optimal Solution	**144**	**150**	**96**	**144**
Simulated Annealing	**144**	**150**	**96**	**144**

For larger data sets with eight workers, eight tasks and twenty time periods, the lower and upper bounds were obtained as discussed in section 7.5.3. The Simulated Annealing algorithm performed considerably better than the Pairwise-Exchange heuristic. The best solutions obtained are given in Table 36. The following settings were used:

- *tpr* (starting temperature) = 4
- α (a temperature reduction function parameter) = 0.99; and annealing schedule *tpr* = α**tpr*;
- maximum number of iterations at each temperature level is 1000 and number of moves without improvement is 800;
- *tpr* > 0.01 stopping condition and the last temperature level.

Table 36: Simulated Annealing Results for larger instances

	Data set 1	**Data set 2**	**Data set 3**
No Rotation Heuristic	**320**	**375**	**383**
Upper Bound	**428.5**	**428.85**	**435**
Pairwise-Exchange Heuristic Solution	**329**	**339**	**346**
Simulated Annealing Solution	**410**	**412**	**426**

Simulated Annealing obtained the best solution 20 to 50% of the time for the large instances presented. Other solutions were within 0.5 to 3.2% of the best solutions obtained. Data for specific runs is presented in Appendix F.

7.6 SUMMARY

We studied a worker assignment problem where workers are assigned on a serial production line and workers produce at either steady-state productivity rates or productivity rates that change with time due to the presence of learning and forgetting. We implemented a Pairwise Heuristic Algorithm as well as a Simulated Annealing Algorithm for both the MIP and MINLP formulations. The Simulated Annealing Algorithm was able to reach the optimal solutions significantly faster than CPLEX for the MIP instances tested. For the smaller MINLP instances tested, the Simulated Annealing Algorithm reached the optimum for every instance tested. We were able to obtain reasonably good solutions for larger instances in a very reasonable time. Based on our analysis, Simulated Annealing seems appropriate for this problem.

As the problem size increases, the search space is very large and Simulated Annealing has the potential of performing better than the "branch and bound" of CPLEX for the MIP instances. Based on our study, solutions times needed for Simulated Annealing Algorithm are significantly lower than solution times for CPLEX. Based on our results for smaller MINLP instances and comparisons with bounds for larger MINLP instances we strongly believe that Simulated Annealing Algorithm is suitable for these types of problems.

8.0 DYNAMIC VS. FIXED ASSIGNMENT: COMPARISON OF RESULTS

We presented the analysis of two different methods of assigning workers on a serial production line. We looked at a dynamic assignment environment where workers work at adjacent stations and there are no buffers between stations. We also looked at a fixed assignment environment where workers work at one station during a given time period. Our goal was to maximize throughput as well as to determine the optimal assignment of workers for both assignment environments.

There are several major differences in the assignment methods of these two environments. For a dynamic assignment environment workers carry a part to successive stations and the starting order of the workers is preserved. There is no intermediate buffer inventory and WIP is minimal. For the fixed assignment environment workers place a part in a downstream buffer when finished, or obtain a part from an upstream buffer when starting to work. Also, total production time is divided into l time periods, where a worker performs only one task, or works at one station, during each time period. Workers may change stations after each time period thus no worker order is preserved. Also, for the fixed assignment environment there are buffers between the stations and the levels of WIP are significantly higher when compared with the dynamic assignment environment. The number of parts stored in each buffer can be constrained or not constrained.

For both assignment methods we compared the optimal throughput obtained. We also looked at the impact of different levels of WIP on the optimal throughput in the case of the fixed

assignment. For the dynamic assignment production line each worker carries a part or starts with the part thus the WIP is at most the number of workers. For the fixed assignment production line there are several starting/ending WIP scenarios that are important to consider. The optimal solution obtained for the dynamic assignment environment considers steady state behavior of the system. Thus, when analyzing a fixed assignment environment we have to take into account that during early production (warm up or first few time periods) intermediate buffers are empty and some workers may be idle due to empty buffers or 'starvation'. In order to prevent workers being idle we may assume that there is some starting inventory in the intermediate buffers. So, for the fixed assignment environment we impose the requirement that the ending intermediate buffer inventory be at least equal to the beginning intermediate buffer inventory.

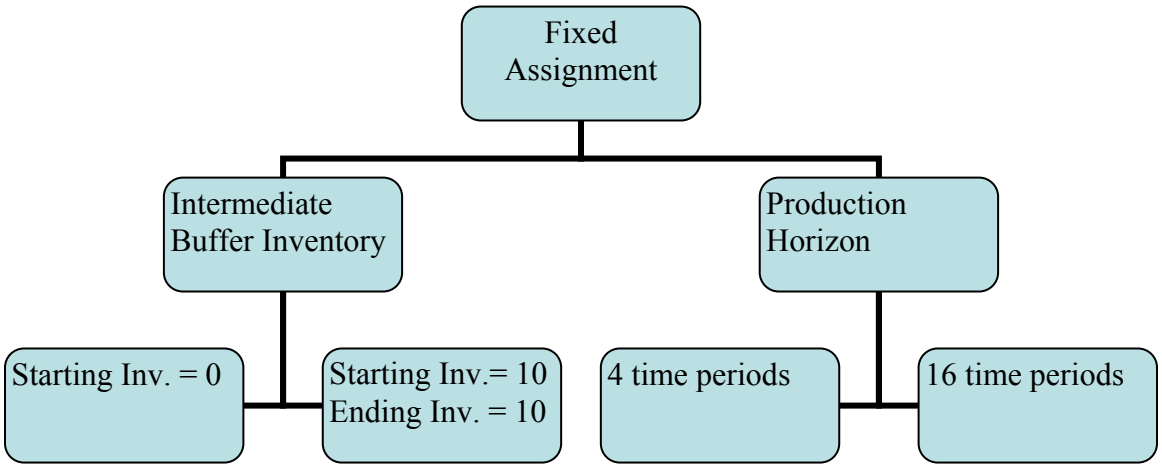


Figure 7: Fixed Assignment combinations tested

Thus, in order to have a "fair comparison", we calculated the total throughput when the beginning as well as ending intermediate buffer inventory is not less than the maximum

production rate of all workers. For example, if the maximum production rate of all workers at all stations is 10 parts / time period we assumed that the beginning/ending inventory should be 10 parts in all intermediate buffers. All possible combinations tested for a fixed assignment are presented in Figure 7. For a dynamic assignment there is no additional inventory or WIP in the system.

8.1 COMPARISON OF RESULTS

Our assumption is that workers' production rates are deterministic and known and modeling assumptions are the same as presented in Sections 3.1 and 4.1 for a dynamic assignment environment and 7.1 for a fixed assignment environment. The optimal solutions obtained for both assignment environments and the optimal throughput as well as different levels of WIP were compared for all data sets tested.

We present results obtained for several data sets in the case of two worker and four station lines. These data sets were selected from a large group of data sets tested in order to present different scenarios that we observed when these two environments were compared. Data sets 1 and 2 were selected in order to illustrate situations when a dynamic assignment results in a higher throughput. Data set 3 illustrates a situation when a fixed assignment results in a higher throughput. Also, these two methods of assignment can result in the same optimal throughput for some data sets. However, in the long run, a fixed assignment will always converge to the maximum attainable throughput as illustrated with data set 4.

Table 37: Dynamic assignment one-cycle solution

Dynamic Assignment: Data set 1

Solution

k_{ij}	1	2	3	4	k_{ij}	1	2	3	4
1	7.01	6.3	5.05	7.55	1			0.50	0.50
2	9.75	8.94	6.59	9.75	2	0.39	0.42	0.19	

Output 3.78

Dynamic Assignment: Data set 2

Solution

k_{ij}	1	2	3	4	k_{ij}	1	2	3	4
1	7.5	8	4.5	7	1	0.47	0.45	0.08	
2	7	5	8	6	2			0.40	0.60

Output 3.58

Dynamic Assignment: Data set 3

Solution

k_{ij}	1	2	3	4	k_{ij}	1	2	3	4
1	8	10	8	10	1			0.56	0.44
2	10	8	10	8	2	0.44	0.56		

Output 4.448

Dynamic Assignment: Data set 4

Solution

k_{ij}	1	2	3	4	k_{ij}	1	2	3	4
1	8	9	5	7	1	0.49	0.44	0.07	
2	5	6	7	8	2			0.51	0.49

Output 3.93

Workers' production rates as well as the optimal solutions for the dynamic assignment environment are presented in Table 37 for data sets 1, 2, 3 and 4. For data set 1 the optimal throughput is 3.78 parts/hour. The optimal throughputs obtained for data sets 2, 3 and 4 are 3.58 parts/hour, 4.44 parts/hour and 3.93 parts/hour, respectively. For data sets 2 and 4 the first worker works at the first segment of the line, and the second worker works at the second segment of the line and only station three is shared. For data sets 1 and 3, W2 is the first in the order, and there is no worksharing for data set 3.

For a fixed assignment environment, we looked at several different scenarios such as zero beginning intermediate buffer inventory or ten parts in each intermediate buffer. However if we would start the production with ten parts in each intermediate buffer we also impose the requirement that the production horizon would end with at least ten parts in each intermediate buffer. A size of ten parts was selected because for all worker/station production rates tested the following inequality is true: $k_{ij} \leq 10$.

It is important to note that the effectiveness of using a fixed versus dynamic assignment varies depending on the length of the time horizon. If we calculate the optimal throughput for a fixed assignment and consider only a few time periods, it is highly possible that the number of completed parts is zero or very low. The reasoning behind this is that at the start of the production horizon workers are mostly assigned at the beginning of the line as there are less workers than stations. For this reason, different time horizons were investigated. The number of total time periods or the length of the production horizon is given in the first column of Table 38. Column 3 represents average output information when the beginning intermediate buffer inventory is zero and there is no requirement regarding the final buffer inventory. The average output presented in Column 5 reflects the requirement that the ending intermediate buffer inventory is at least as large as the beginning intermediate buffer inventory (in this case 10 parts).

For data set 1 and a production horizon of 16 time periods, the optimal average throughput obtained is 3.73 parts/period (see Column 5), which is lower than the 3.78 parts/period obtained when workers are assigned dynamically.

Table 38: Data set 1: Average Output

1 Time Periods	2 Output Inv 0	3 Average Output	4 Output; Inv 10 Start = End	5 Average Output
4	13.18	3.30	13.18	3.30
8	29.25	3.66	29.25	3.66
12	43.48	3.63	44.7	3.72
16	59.74	3.73	59.74	3.73

Results for data set 2 are presented in Table 39. For 16 time periods, the optimal average throughput obtained is 3.5 parts/period (see Column 7), which is lower than the 3.5757 parts/period obtained when workers are assigned dynamically.

Table 39: Data set 2: Average Output

1 Time Periods	2 Output Inv 0	3 Average Output	4 Output; Inv 10 Start = End	5 Average Output
4	14	3.5	14	3.5
8	28	3.5	28	3.5
12	42	3.5	42	3.5
16	56	3.5	56	3.5

Results for data set 3 are given in Table 40. For data set 3 and 16 time periods, the optimal average throughput obtained is 5 parts/period (see Column 5), which is higher than the 4.444 parts/period obtained when workers are assigned dynamically.

Table 40: Data set 3: Average Output

1 Time Periods	2 Output Inv 0	3 Average Output	4 Output; Inv 10 Start = End	5 Average Output
4	20	5	20	5
8	40	5	40	5
12	60	5	60	5
16	80	5	80	5

Results for data set 4 are presented in Table 41. The optimal average throughput obtained for 16 time periods and 10 parts beginning/ending inventory is 3.94 parts/hour which is higher than the 3.93 parts/hour obtained when workers are assigned dynamically.

Table 41: Data set 4: Average Output

1 Time Periods	2 Output Inv 0	3 Average Output	4 Output; Inv 10 Start = End	5 Average Output
4	13	3.25	14	3.5
8	28	3.5	28	3.5
12	42	3.5	45	3.75
16	60	3.75	63	3.94

Based on the workers' production rates, we can conclude that for data sets 1 and 2 it is better to assign workers dynamically even when intermediate buffer inventory is present. However, for data sets 3 and 4, it is better to assign workers according to a fixed assignment policy. A summary of the results for the four data sets presented is given in Table 42. In order to better compare the obtained optimal throughputs for both assignment environments we also looked at the results of the LP relaxation for the fixed assignment model because it is an upper bound on the optimal throughput. We compared the throughputs obtained with the upper bound.

The optimal throughput obtained for data set 1 through dynamic assignment is the same as the LP relaxation and the optimal throughput obtained for data set 3 through the fixed assignment is the same as the LP relaxation. The solution presented for the fixed assignment is the average output obtained for 4 and 16 time periods with either the beginning intermediate buffer inventory zero, or with a beginning/ending intermediate buffer inventory of 10 parts. We can see from the data sets presented, that there are cases when dynamic assignment performs better than a fixed assignment.

Table 42: Comparison of results

OUTPUT	**Data set 1**	**Data set 2**	**Data set 3**	**Data set 4**
LP Relaxation	**3.78**	3.73	**5.00**	3.95
Dynamic Solution (MIP)	**3.78**	3.58	4.44	3.93
Fixed Assignment Solution (4 periods 0 Inventory)	3.29	3.5	**5.00**	3.25
Fixed Assignment Solution (4 periods 10 Inventory Start = End)	3.29	3.5	**5.00**	3.5
Fixed Assignment Solution (16 periods 0 Inventory)	3.73	3.5	**5.00**	3.75
Fixed Assignment Solution (16 periods 10 Inventory Start = End)	3.73	3.5	**5.00**	**3.94**

Another way to compare the assignments is to look at a comparison of the average throughput for a fixed assignment (throughput obtained per period) with the LP relaxation and the dynamic assignment. We want to see how fast the output would converge to the upper bound for different lengths of a production horizon. For data set 4 we can see from Figure 8 that the dynamic assignment one-cycle solution is very close to the upper bound obtained by the LP relaxation. Also, the fixed assignment solution when we have 10 parts of beginning/ending

inventory is quickly converging to the upper bound. Depending on the data sets we could plot similar graphs and determine the optimal assignment approach.

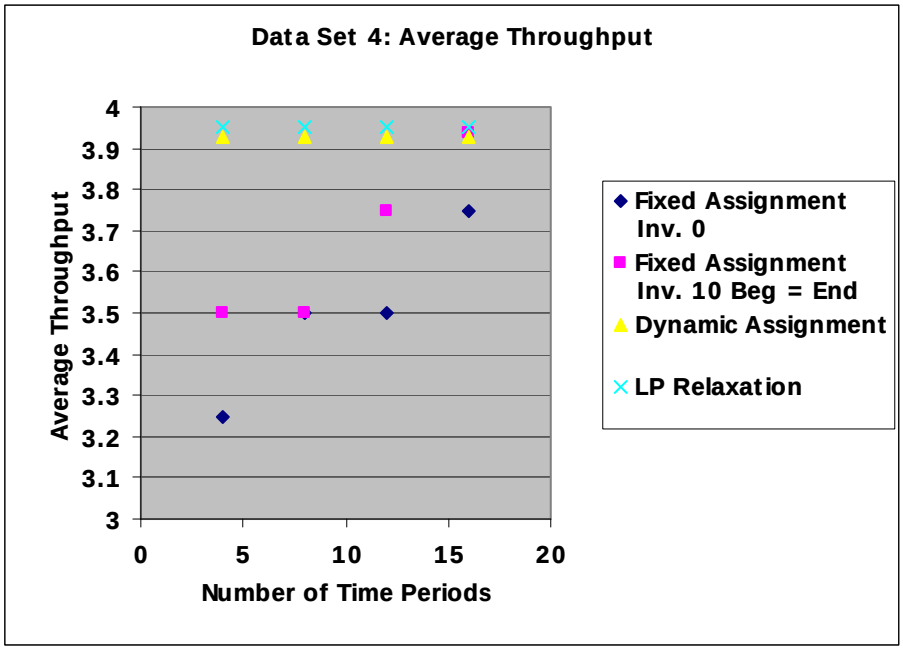


Figure 8: Data set 4: Average Throughput Convergence

We also discussed that one of the main differences between these two environments is the level of WIP. We can see from Table 42 that a fixed assignment approach results in a higher throughput for data sets 3 and 4. However, we didn't take into account WIP levels during the production as we assume that intermediate buffer levels during the production horizon are unconstrained. Thus, in the case of the fixed assignment environment we need to look how the intermediate buffer inventory changes during the production run. Our primary objective is to maximize throughput but at the same time to minimize (or control) the number of parts in each buffer during the each time period. When solving a mixed integer model formulation for the fixed assignment, we observed that there are many alternative solutions (depending on the data

set or given production rates). For data set 4, we recorded buffer inventory levels for 4, 8, 12 and 16 time periods and these levels are presented in Table 43.

Table 43: Maximum Buffer Inventory for Data set 4

	Beginning Inv 0			**Beginning Inv 10, End Inv 10**		
Periods\Buffers	**B2**	**B3**	**B4**	**B2**	**B3**	**B4**
4	10	9	6	24	12	11
8	8	14	14	19	17	19
12	**19**	14	9	19	15	**38**
16	12	13	**19**	24	13	**31**

We can see from the table that when the beginning intermediate buffer inventory is zero the maximum buffer level is 19 parts. If we look at the last three columns in Table 43, the highest number of parts in the intermediate buffers is 38. This is the maximum number of parts when we require the beginning /ending inventory to be no less then 10 parts.

Thus, we needed to consider different ways to minimize the intermediate buffer inventory in order to better compare our assignment environments. In order to determine the best solution based on the throughput obtained, but also with the minimum possible buffer inventory we modified the objective function to include the term $-coef\sum_{j=2}^{m}\sum_{t=1}^{l} b_{jt}$. This modified objective function takes into account the total intermediate buffer inventory. The maximum buffer inventory obtained when *coef = 0.01* is presented in Table 44. The optimal throughput obtained remained unchanged. We can see that the maximum buffer inventory given in

Table 44 is significantly lower than the maximum buffer given in Table 43. Thus, when studying the maximum levels of buffer inventory we should be aware of the existence of alternative solutions. Also, the value of *coef* is dependent on the given data set. In conclusion,

intermediate buffer inventory levels should be taken into consideration when comparing the different assignment environments.

Table 44: Maximum Buffer Inventory for Data set 4 with the modified objective

	Beginning Inv 0			**Beginning Inv 10, End Inv 10**		
Periods\Buffers	**B2**	**B3**	**B4**	**B2**	**B3**	**B4**
4	2	7	0	10	18	10
8	8	14	7	10	19	10
12	8	14	7	16	19	10
16	11	14	13	16	19	14

8.2 SUMMARY

In this chapter we compared dynamic versus fixed assignment environments under the same conditions such as number of workers, number of stations and workers' production rates. There are several advantages when assigning workers dynamically:

- Work-in-process (WIP) is controlled and is minimal.
- Throughput is continuous as there is one finished part in constant intervals, so it's easier to plan further production or storage of finished products.
- The layout is simpler as well as starting investments needed for the production line.
- The space requirement for a production line without intermediate buffers is smaller.
- Training workers for specific tasks is easier to accomplish in the case of dynamic assignment. Workers are assigned to a portion of the line so they need to be trained for one region or segment of the line only. For the fixed assignment environment, it can be

seen from the examples tested that a worker can be assigned to any station on the line requiring him/her to be trained for all tasks.

However, when assigning workers dynamically workers exchange parts at any point in time, thus this environment cannot be applied in situations when tasks are not preemptive. In situations when inventory holding cost is minimal and space requirements require low investments, then a fixed assignment environment may be preferable assuming that the production horizon is sufficiently long.

9.0 CONCLUSIONS AND FUTURE WORK

This chapter summarizes conclusions from the first 9 chapters. It also discusses potential extensions of this research.

9.1 CONCLUSIONS

This research focused on developing methods to solve a worker assignment problem on a serial production line. Both dynamic and fixed assignment environments were considered with the objective of maximizing throughput. The problems that were discussed and analyzed assumed a heterogeneous workforce and also considered the impacts of individual learning and forgetting on system throughput.

For a dynamic assignment environment we first considered how the assignment of a fully cross-trained workforce deployed on a serial production line consisting of two workers and two stations affects throughput. We analyzed different worker assignment policies as well as different levels of worksharing and determined their impact on system performance. For the dynamic assignment environment, we were able to define possible benefits achieved from worksharing and the impact of different assignment approaches and policies on system performance. We defined conditions when idle time is beneficial, as well as how to optimally assign workers based on their production rates. For the two worker two station line, in the

complete dominance case, the faster worker may be assigned at the beginning of the line, rather than the end of the line in order to achieve the highest throughput. This is counter to what is found in normative bucket brigade systems which order workers from slowest to fastest to maximize throughput.

For the two worker two station line, we also investigated the value of duplicating workstations as well as how the system performs if the system performs as two parallel lines due to duplicate tooling at both stations. One of our main results is that the maximum possible output attained from two parallel lines is never greater than the output obtained from a line with two stations when one of those stations has duplicate tooling and the workers are optimally ordered.

Another important focus of this research was to optimally solve a dynamic assignment problem for any number of workers and stations and determine exact proportions of work performed by each worker under the assumptions presented. Thus, another main contribution is the development of one-cycle mixed integer programming formulation for n workers and m stations. We were able to obtain optimal assignments of workers and determine optimal levels of throughput. We also extended the one-cycle formulation to incorporate multiple production lines. We were able to determine optimal assignments of workers to a set of production lines based on the optimality criteria defined. We analyzed two different types of production lines: one where the production lines can be viewed as a set of l independent lines and a second where the production lines are l dependent lines.

Another significant contribution is the development of a two-cycle formulation that models the case where workers exchange parts at exactly two positions in a periodic manner with the objective to maximize throughput. We were able to achieve the optimal throughput and

determine the optimal assignment of workers. This is an important result as the previous literature recognizes the existence of a two-cycle behavior which results in a suboptimal throughput. We were able to show examples where a two-cycle solution results in the optimal throughput under the modeling assumptions presented.

We also investigated dynamic assignment when worker learning and forgetting is present, or workers do not perform at steady state levels. We considered a dynamic worksharing system with n workers and m stations and compared optimal assignments and throughputs when workers produce at their steady state levels versus when learning and forgetting is considered.

For a fixed assignment environment we considered a worker assignment problem on a serial production line where workers either perform at steady state production levels or where workers' productivity changes due to the presence of learning and forgetting. Two formulations were considered: the MIP formulation with steady state production rates, and the MINLP formulation which includes the individual worker learning and forgetting effects.

A main contribution is the development and implementation of heuristic methods to solve both MIP and MINLP problems and determine the optimal throughput levels and optimal worker assignment. A Simulated Annealing Algorithm was implemented and was able to attain the optimal solutions considerably faster than CPLEX for large MIP instances. For smaller MINLP instances tested, the Simulated Annealing Algorithm reached the optimum for every instance tested. For larger MINLP instances, we were able to obtain good solutions in a very reasonable time. Based on our study, Simulated Annealing seems suitable for this problem.

A final contribution of this research is the comparison of two different worker assignment environments. We compared dynamic and fixed assignment methods given the same number of workers, number of stations, workers' production rates and the length of the production horizon

under the assumptions presented and were able to determine advantages and disadvantages when applying these two assignment methods. For short production horizons, dynamic assignment has many advantages such as minimal and controlled WIP and constant throughput. For long production horizons the fixed assignment environment can be preferable as the maximum possible throughput converges to the maximum attainable throughput.

9.2 FUTURE RESEARCH

For both dynamic assignment and fixed assignment environments our future research has many directions. For a dynamic assignment environment we are interested in determining the value of duplicating workstations for larger production lines, as well as the value of duplicating stations in the multiple line production settings.

We want to expand our focus on dynamic systems with workers learning and forgetting and introduce varying productivity into the one-cycle and two-cycle math formulations The additional research would then focus on the development and implementation of suitable heuristic methods in order to solve these problems.

Another focus is to introduce blocking into the two-cycle formulation. At present, the formulation does not allow for an exchange of parts at the first station and we would consider alternative formulations to allow for this situation. In our experience, the exchange of parts never occurs at the first station for a line of more than two stations that has realistic worker production rates. In addition, in the future we may focus on extending the two-cycle formulation to a k-cycle math formulation for cases of three or more workers.

For a fixed assignment environment we may focus on the development of additional heuristic rules to determine a better "starting assignment" such as selecting the assignment based on the highest steady-state values. Another direction would be to implement additional heuristic algorithms such as a Tabu Search Algorithm.

Also, for a fixed assignment environment we can focus on solving models with different objectives such as to maximize the output given n workers, m stations and a production horizon T; to minimize the number of workers needed given a demand D to satisfy, m stations and a production horizon T; or to minimize the makespan given n workers, m stations and a demand D to satisfy.

For a dynamic as well as a fixed assignment environment we may consider the use of other learning models. For a fixed assignment environment it may also be constructive to conduct a larger computational study to be able to make stronger generalized conclusions concerning learning and forgetting effects. For example, it would be possible to determine the impacts of having bottleneck stations located at different portions of the production line.

APPENDIX A

TWO WORKER TWO STATION CASE ANALYSIS

Case 2 **analysis:** Recall that for Case *2($k_{12} \le k_{11} \le k_{21} \le k_{22}$)* we have the production rates given in Table 1 and that $a \le b \le c \le d$.

Table 45: Case *2* workers' production rates

k_{ij}	*S1*	*S2*
W1	*b*	*a*
W2	*c*	*d*

In this case we have that $k_{11}=b$, $k_{12}=a$, $k_{21}=c$ and $k_{22}=d$.

Option *1*: $Output^1 = min\ (k_{11}, k_{22}) = min\ (b, d) = b.$

Option *2*: $Output^2 = min\ (k_{21}, k_{12}) = min\ (c, a) = a.$

Option *3*: Under the assumption that *W1* is the first in the order and *W2* is the second, the following equations can be written: $x_{11}* k_{11} + x_{21}* k_{21} = x_{22}* k_{22}$ and $x_{12} = 0$, as the second worker, or *W2* in this case, cannot be idle which implies that *W1* can never be assigned to *S2*, thus we have that $x_{12}=0$. *Also,* $x_{21}+x_{22} = 1.0$ (*W*2 is always working) and $x_{11}+x_{21} = 1.0$ (*S*1 is always occupied). By solving these equations and assuming that $x_{12} = 0$, we have the following: $x_{21} = (k_{22} – k_{11})/(k_{22} + k_{21} – k_{11})$ and $Output^3 = (1-x_{21})*k_{22} = x_{22}*k_{22} = (k_{21} / (k_{22} + k_{21} – k_{11}))*k_{22} = k_{21}*k_{22}/(k_{22} + k_{21} – k_{11}) = c*d/(d + c – b).$

Option *4*: Under the assumption that *W2* is the first and *W1* is the second (and cannot be idle) we have the following equation: $x_{21}* k_{21} = x_{12}* k_{12}$ or $x_{21}* c = x_{12}* a.$

As $c \ge a$ (*or* $k_{21} \ge k_{12}$) thus, *W1* would never work at *S1* as *W2* is faster and he/she will wait portion of the time. So, the output is: $Output^4 = min\ (c, a) = min\ (k_{21}, k_{12}) = a$ (*or* k_{12}).

Option *5*: For these production rates this Option is equal to *Option 3* as $b \leq d$ *or* $(k_{11} \leq k_{22})$, thus if workers are ordered in this manner, *W2* will never be idle. Thus $Output^5 = k_{21}*k_{22}/(k_{22} + k_{21} - k_{11}) = c*d/(d + c - b)$.

Option *6*: *W2* is the first in the order, and the second worker in the order, or *W1*, can be idle, thus *W2* is assigned to both stations and *W1* is assigned only to *S2* (this is implied by the relationship of the production rates given). Then the following is true: $x_{21}* k_{21} = x_{22}* k_{22} + x_{12}* k_{12}$ *and* $x_{11} = 0$ *or* $x_{21}* c = x_{22}* d + x_{12}* a$. Also, $x_{21}+x_{22} = 1.0$ as worker *2* is always working and the second station is always occupied so $x_{22} + x_{12} = 1.0$. By solving these equations, we have the following: $x_{21} = 1.0 - x_{22}$ *and* $x_{12} = 1.0 - x_{22}$ and $x_{22} = (k_{21} - k_{12})/(k_{22} + k_{21} - k_{12})$. The output from the line under these assumptions (worker *2* is the first and worker *1* is the second) is: $Output^6 = (1-x_{22})*k_{21} = x_{21}*k_{21} = (k_{22} / (k_{22} + k_{21} - k_{12}))*k_{21} = k_{22}*k_{21}/(k_{22} + k_{21} - k_{12}) = c*d/(d + c - a)$.

Summary: Under the assumption that $k_{12} \leq k_{11} \leq k_{21} \leq k_{22}$ it is clear that $Output^2 \leq Output^1$ and $Output^4 \leq Output1^1$. Also, it is clear that $Output^6 \leq Output^3$ or $Output^5$ as they are equal. We can also show that $Output^4 \leq Output^3$. We have to show that $a \leq c*d/(d + c - b)$. After rearranging the terms we have $d * (c-a) + a* (b-c) \geq 0$. Based on our assumption $a \leq b \leq c \leq d$, we know that $c- a \geq 0$ and $b-c \leq 0$. However, as $d \geq a$, we can assume that $d*(c-a) + a*(b-c) \geq 0$ will always be positive. We also have to show that $Output^1 \leq Output^3$ or $b \leq c*d/(d + c - b)$. After rearranging the terms we have $c(d-b)-b(d-b) \geq 0$ or $(d-b)(c-b) \geq 0$. Based on our assumption $a \leq b \leq c \leq d$, we know that $c- b \geq 0$ and $d-b \geq 0$ thus $b \leq c*d/(d + c - b)$. The optimal assignment is *W1-W2* or Option *5 (or* Option *3)*.

Table 46: Case 2 output rates

	Output rate	*Optimal*
Option 1	*b*	
Option 2	*a*	
Option 3	*c*d/(d + c – b)*	*c*d/(d + c – b)*
Option 4	*a*	
Option 5	*c*d/(d + c – b)*	*c*d/(d + c – b)*
Option 6	*c*d/(d + c – a)*	

Examples:

k_{ij}	*S1*	*S2*
W1	*i* 19	18
W2	20	21

k_{ij}	*S1*	*S2*
W1	*i* 19	18
W2	20	60

Case 3 analysis: Recall that for Case 3 $(k_{11} \leq k_{12} \leq k_{22} \leq k_{21})$ we have the production rates given in Table 1 and that $a \leq b \leq c \leq d$.

Table 47: Case 3 workers' production rates

k_{ij}	*S1*	*S2*
W1	*a*	*b*
W2	*d*	*c*

In this case we have that $k_{11}=a$, $k_{12}=b$, $k_{21}=d$ and $k_{22}=c$.

Option *1*: $Output^1 = min\ (k_{11}, k_{22}) = min\ (a, c) = a$.

Option *2*: $Output^2 = min\ (k_{12}, k_{21}) = min\ (b, d) = b$.

Option *3*: Similarly to the Case *1* analysis: $x_{11}* k_{11} + x_{21}* k_{21} = x_{22}* k_{22}$ and $x_{12} = 0$ or $x_{11}* a + x_{21}* d = x_{22}* c$ and $x_{12} = 0$. We also have: $x_{21}+x_{22} = 1.0$ (*W2* is always working) and $x_{11}+x_{21} = 1.0$ (*S1* is always occupied).By solving these equations and assuming that $x_{12} = 0$, we have the following: $Output^3 = (1-x_{21})*k_{22} = x_{22}*k_{22} = (k_{21} / (k_{22} + k_{21} – k_{11}))*k_{22} = k_{21}*k_{22}/(k_{22} + k_{21} – k_{11}) = c*d/(d + c – a)$.

Option *4*: Under the assumption that *W2* is the first and *W1* is the second (and cannot be idle) we have the following equation: $x_{21}* k_{21} = x_{12}* k_{12}$ *or* $x_{21}* d = x_{12}* b$.

As $d \geq b$ *(or* $k_{21} \geq k_{12}$) thus, *W1* would never work at *S1* as *W2* is faster and he/she will wait portion of the time. So, the output is: $Output^4 = min\ (d, b) = min\ (k_{21}, k_{12}) = b$ *(or* k_{12}*)*.

Option *5*: This Option is equal to Option *3* as $a \leq c$ or $(k_{11} \leq k_{22})$, thus if workers are ordered in this manner, *W2* will never be idle. Thus $Output^5 = k_{21}*k_{22}/(k_{22} + k_{21} – k_{11}) = c*d/(d + c – a)$.

Option *6*: *W2* is the first in the order, and the second worker in the order can be idle, thus *W2* is assigned to both tasks and *W1* is assigned only to the second task (this is implied by the relationship of the production rates given). Then the following is true: $x_{21}* k_{21} = x_{22}* k_{22} + x_{12}* k_{12}$ *and* $x_{11}= 0$ *or* $x_{21}* d = x_{22}* c + x_{12}* b$ and $x_{21}+x_{22} = 1.0$ (*W2* is always working). Also, $x_{22} + x_{12} = 1.0$ (*S2* is always occupied). Solving these equations, we have the following: $x_{21} = 1.0 - x_{22}$ *and* $x_{12} = 1.0 - x_{22}$ and $x_{22} = (k_{21} – k_{12})/(k_{22} + k_{21} – k_{12})$. The output is: $Output^6 = (1-x_{22})*k_{21} = x_{21}*k_{21} = (k_{22} / (k_{22} + k_{21} – k_{12}))*k_{21} = k_{22}*k_{21}/(k_{22} + k_{21} – k_{12}) = c*d/(d + c – b)$.

Similarly as presented for Case *1* and Case *2*, we can show that *Option 6* is optimal or *W2-W1* assignment, as $c*d/(d + c – b) \geq b \geq a$ and $c*d/(d + c – b) \geq c*d/(d + c – a)$.

Table 48: Case 3 output rates

	Output rate	*Optimal*
Option 1	*a*	
Option 2	*b*	
Option 3	*c*d/(d + c – a)*	
Option 4	*b*	
Option 5	*c*d/(d + c – a)*	
Option 6	*c*d/(d + c – b)*	*c*d/(d + c – b)*

Also, as discussed for *Case 1*, if we not allow the second worker to be idle we would have to choose between *Option 3* and *Option 4*. When $b \geq c*d/(d + c – a)$ or $Output^4 \geq Output^3$ workers will be sequenced *W2-W1* or faster worker will be first in the order. When this condition does not hold the assignment will be *W1-W2* or the slower worker in this worker will be first in the order.

Example: $b \geq c*d/(d + c – a)$ *Example:* $b \geq c*d/(d + c – a)$

k_{ij}	*S1*	*S2*
W1	*18*	*19* ^{i}
W2	*21*	*20*

kij	*S1*	*S2*
W1	*18*	*19* ^{i}
W2	*100*	*20*

Case 4 analysis: Recall that for Case 4 $(k_{12} \leq k_{11} \leq k_{22} \leq k_{21})$ we have the production rates given in Table 1 and that $a \leq b \leq c \leq d$.

Table 49: Case 4 workers' production rates

k_{ij}	*S1*	*S2*
W1	*b*	*a*
W2	*d*	*c*

In this case we have that $k_{11}=b$, $k_{12}=a$, $k_{21}=d$ and $k_{22}=c$.

Option 1: $Output^1 = min\ (k_{11}, k_{22}) = min\ (b, c) = b.$

Option 2: $Output^2 = min\ (k_{21}, k_{12}) = min\ (d, a) = a.$

Option 3: $Output^3 = k_{21}*k_{22}/(k_{22} + k_{21} - k_{11}) = c*d/(d + c - b)$.

Option 4: $Output^4 = min\ (d, a) = min\ (k_{21}, k_{12}) = a\ (or\ k_{12})$.

Option 5: As $c \geq b$, *W1* would never be idle thus $Output^5 = Output^3 = c*d/(d + c - b)$.

Option 6: In this case it is beneficial for W1 to be idle, as $c \geq a$. Thus, $Output^6 = k_{21}*k_{22}/(k_{22} + k_{21} - k_{12}) = c*d/(c + d - a)$.

It is clear (see Case *1* and Case *2* proofs) that $Output^4 \leq Output^6 \leq Output^3$. The optimal workers order is *W1 - W2,* or *Option 3 (Option 5).*

Table 50: Case 4 output rates

	Output rate	*Optimal*
Option 1	*b*	
Option 2	*a*	
Option 3	*c*d/(d + c – b)*	*c*d/(d + c – b)*
Option 4	*a*	
Option 5	*c*d/(d + c – b)*	*c*d/(d + c – b)*
Option 6	*c*d/(d + c – a)*	

Examples

k_{ij}	*S1*	*S2*
W1	*8*	*7*
W2	*10*	*9*

k_{ij}	*S1*	*S2*
W1	*8*	*7*
W2	*100*	*9*

Case 5 analysis: Recall that for Case 5 $(k_{11} \leq k_{21} \leq k_{22} \leq k_{12})$ we have the production rates given in Table 1 and that $a \leq b \leq c \leq d$.

Table 51: Case 5 workers' production rates

k_{ij}	*S1*	*S2*
W1	*a*	*d*
W2	*b*	*c*

Option 1: $Output^1 = min\ (k_{11}, k_{22}) = min\ (a, d) = a$.

Option 2: $Output^2 = min\ (k_{21}, k_{12}) = min\ (c, b) = b$.

Option 3: $Output^3 = (k_{21}/(k_{22} + k_{21} - k_{11}))*k_{22} = k_{21}*k_{22}/(k_{22} + k_{21} - k_{11}) = c*b/(b + c - a)$.

Option 4: $Output^4 = k_{11}*k_{12}/(k_{11} + k_{12} - k_{21}) = a*d/(a + d - b)$.

Option 5: In this case Option *5* is equal to Option *3* as $a \le c$ *or* $(k_{11} \le k_{22})$, thus if workers are ordered in this manner, *W2* will never be idle. Thus, $Output^5 = k_{21}*k_{22}/(k_{22} + k_{21} - k_{11}) = c*b/(b + c - a)$.

Option 6: As $b \le d$, the second worker in the order will never be idle thus $Output^6 = k_{11}*k_{12}/(k_{11} + k_{12} - k_{21}) = a*d/(a + d - b)$.

Summary: Output rates for all options are presented in Table 52. We can show that *Option 2* is optimal, or that *W2-W1* is the optimal order. It is clear that $Output^2 \ge Output^4$ *(or* $Output^6$*),* or $b \ge a*d/(a + d - b)$, as it reduces to positive terms $(d - b)(b - a)/(a + d - b) \ge 0$. Also, $Output^2 \ge Output^3$ *(or* $Output^5$*),* $b \ge c*b/(b + c - a)$ as it reduces to $b*(b - a)/(b + c - a) \ge 0$.

Table 52: Case 5 output rates

	Output rate	*Optimal*
Option 1	*a*	
Option 2	*b*	*b*
Option 3	*c*b/(b + c – a)*	
Option 4	*a*d/(a + d - b)*	
Option 5	*c*b/(b + c – a)*	
Option 6	*a*d/(a + d - b)*	

If we want to apply only bucket brigade rules, we can look at *Options 3* and *4*. Under bucket brigade rules, it is not feasible to have the second worker idle, thus we have to decide if it is better to assign workers according to *Option 3* or *Option 4*. If $c*b/(b + c - a) \ge a*d/(a + d - b)$ then *Option 3* is optimal and order is *W1-W2,* otherwise *Option 4* is optimal, and order is *W2-W1*.

Examples:

k_{ij}	*S1*	*S2*
W1	*7*	*10*
W2	*8*	*9*

k_{ij}	*S1*	*S2*
W1	*7*	*100*
W2	*9*	*10*

Case 6 analysis: Recall that for Case 6 $(k_{12} \le k_{21} \le k_{22} \le k_{11})$ we have the production rates given in Table 1 and that $a \le b \le c \le d$.

Table 53: Case 6 workers' production rates

k_{ij}	*S1*	*S2*
W1	*d*	*a*
W2	*b*	*c*

In this case we have that $k_{11}=d$, $k_{12}=a$, $k_{21}=b$ and $k_{22}=c$.

Option 1: $Output^1 = min\ (k_{11}, k_{22}) = min\ (d, c) = c$.

Option 2: $Output^2 = min\ (k_{21}, k_{12}) = min\ (b, a) = a$.

Option 3: As $d \geq c$, $Output^3 = min\ (d, c) = min\ (k_{11}, k_{22}) = c$ (or k_{22}).

Option 4: As $b \geq a$, $Output^4 = min\ (b, a) = min\ (k_{21}, k_{12}) = a$ (or k_{12}).

Option 5: There is no benefit of the second worker on the order to be idle, as the first workers' rate at *S2* is lower then the second workers' rate at *S2,* thus $Output^5 = Output^3 = c$. If we allow *W2*

Option 6: *As* $c \geq a$, it is beneficial for the second worker to be idle, thus $Output^6 = k_{21}*k_{22}/(k_{22} + k_{21} - k_{12}) = c*b/(c + b - a)$.

Summary: It is clear that $Output^2 = Output^4 \leq Output^1$. Also, we can show that $Output^6 \leq Output^1$ or $c*b/(c + b - a) \leq c$, which reduces to $c(c-a) \geq 0$ which is true as $c \geq a$. The optimal order is *W1-W2,* or *Option 1*.

Table 54: Case 6 output rates

	Output rate	*Optimal*
Option 1	*c*	
Option 2	*a*	
Option 3	*c*	
Option 4	*a*	
Option 5	*c*	
Option 6	*c*b/(c + b - a)*	

Examples:

k_{ij}	*S1*	*S2*
W1	*i* 10	7
W2	8	9

k_{ij}	*S1*	*S2*
W1	*i* 20	17
W2	18	19

Case 7 analysis: Recall that for Case *5* $(k_{11} \leq k_{22} \leq k_{21} \leq k_{12})$ we have the production rates given in Table 1 and that $a \leq b \leq c \leq d$.

Table 55: Case 7 workers' production rates

k_{ij}	*S1*	*S2*
W1	*a*	*d*
W2	*c*	*b*

Option 1: $Output^1 = min\ (k_{11}, k_{22}) = min\ (a, b) = a.$

Option 2: $Output^2 = min\ (k_{21}, k_{12}) = min\ (c, d) = c.$

Option 3: $Output^3 = (k_{21}/(k_{22} + k_{21} - k_{11}))*k_{22} = k_{21}*k_{22}/(k_{22} + k_{21} - k_{11}) = c*b/(b + c - a).$

Option 4: $Output^4 = k_{11}*k_{12}/(k_{11} + k_{12} - k_{21}) = a*d/(a + d - c).$

Option 5: *Option 5* is equal to *Option 3* as $a \leq b$ *or* $(k_{11} \leq k_{22})$, thus if workers are ordered in this manner, *W2* will never be idle. Thus $Output^5 = c*b/(b + c - a).$

Option 6: *Option 6* is equal to *Option 4* as $c \leq d$ *or* $(k_{21} \leq k_{12})$, thus if workers are ordered in this manner, *W1* will never be idle. $Output^6 = a*d/(a + d - c).$

Summary: We can show that *Option 2* is optimal, or that *W2-W1* is the optimal assignment. It is clear that $c \geq a*d/(a + d - c)$, as it reduces to positive terms $(c-a)*(d-c)/(a + d - c) \geq 0$. Also, $c \geq c*b/(b + c - a)$ *as it reduces to* $c*(c-a)/(b + c - a) \geq 0$. Also, we know that $c \geq a$.

If it is not feasible to have the second worker idle, or just apply bucket brigade rules, we have to decide is it better to assign workers according to *Options 3* or *4*. If $c*b/(b + c - a) \geq a*d/(a + d - c)$ then the optimal assignment is *W1-W2,* otherwise *W2-W1*.

Table 56: Case 7 output rates

	Output rate	*Optimal*
Option 1	*a*	
Option 2	*c*	*c*
Option 3	*c*b/(b + c – a)*	
Option 4	*a*d/(a + d - c)*	
Option 5	*c*b/(b + c – a)*	
Option 6	*a*d/(a + d - c)*	

Examples:

k_{ij}	*S1*	*S2*
W1	*7*	*10* ^{i}
W2	*9*	*8*

k_{ij}	*S1*	*S2*
W1	*7*	*100* ^{i}
W2	*9*	*8*

Case 8 analysis: Recall that for Case 8 *($k_{12} \le k_{22} \le k_{21} \le k_{11}$)* we have the production rates given in Table 1 and that $a \le b \le c \le d$.

Table 57: Case 8 workers' production rates

k_{ij}	*S1*	*S2*
W1	*d*	*a*
W2	*c*	*b*

Option 1: Output^1 $= min\ (k_{11}, k_{22}) = min\ (d, b) = b.$

Option 2: Output^2 $= min\ (k_{21}, k_{12}) = min\ (c, a) = a.$

Option 3: As $d \ge b$, *W1* finishes first, thus *Output*^{3} $= min\ (k_{11}, k_{22}) = min\ (d, b) = b.$

Option 4: As $c \ge a$, *W2* finishes first, thus *Output*^{4} $= min\ (k_{21}, k_{12}) = min\ (c, a) = a.$

Option 5: As $d \ge b$ but $a \le b$, it is not beneficial for *W2* to be idle, thus *Output*^{5} $=$ *Output*^{3} $= b.$

Option 6: As $b \ge a$, or $k_{22} \ge k_{12}$, it is beneficial for *W1* to be idle. Thus, *Output*^{6} $= k_{21}*k_{22}/(k_{22} + k_{21} - k_{12}) = c*b/(c + b-a).$

Summary: It is clear that *Option 3* is optimal as $b \ge a$, and we can show that $b \ge c*b/(c + b-a)$. If we rearrange the terms the inequality reduces to $b(b - a) \ge 0$, which is true.

Table 58: Case 8 output rates

	Output rate	*Optimal*
Option 1	*b*	*b*
Option 2	*a*	
Option 3	*b*	*b*
Option 4	*a*	
Option 5	*b*	*b*
Option 6	*c*b/(c + b-a)*	

Examples:

k_{ij}	*S1*	*S2*
W1	*10*	*7*
W2	*9*	*8*

k_{ij}	*S1*	*S2*
W1	*100*	*7*
W2	*29*	*25*

Case 9 analysis: Recall that for Case 9 $(k_{11} \le k_{21} \le k_{12} \le k_{22})$ we have the production rates given in Table 1and that $a \le b \le c \le d$.

Table 59: Case 9 workers' production rates

k_{ij}	*S1*	*S2*
W1	*a*	*c*
W2	*b*	*d*

Option 1: $Output^1 = min\ (k_{11}, k_{22}) = min\ (a, d) = a.$

Option 2: $Output^2 = min\ (k_{11}, k_{22}) = min\ (b, c) = b.$

Option 3: $Output^3 = k_{22}*k_{21}/(k_{22} + k_{21} - k_{11}) = b*d/(b + d - a).$

Option 4: $Output^4 = k_{11}*k_{12}/(k_{11} + k_{12} - k_{21}) = a*c/(a + c - b).$

Option 5: As $d \ge a$, the second worker will not be idle, thus $Output^5 = Output^3 = b*d/(b + d - a).$

Option 6: It is beneficial for the second worker to be idle as he/she is slower on the first task. Thus, $Output^6 = Output^2 = min\ (k_{21}, k_{12}) = min\ (b, c) = b.$

Summary: We can show that $Output^5 = Output^3 \le Output^6$. The optimal output is *Option 6* as $b \ge a*c/(a+c-b)$. After rearranging the terms we have $(c-b)*(b-a)/(a + c - b) \ge 0$. This inequality is true as all the terms are positive by our assumption. We also have to show that $Output^4 \le Output^6$, or that $a*c/(a + c - b) \le b$. This inequality reduces to $(c-b)*(b-a) \ge 0$, which is true. Thus the optimal order is *W2-W1,* and optimal output rate is *b*.

Table 60: Case 9 output rates

	Output rate	*Optimal*
Option 1	*a*	
Option 2	*b*	*b*
Option 3	*b*d/(b + d - a)*	
Option 4	*a*c/(a + c - b)*	
Option 5	*b*d/(b + d - a)*	
Option 6	*b*	*b*

However, if we want the system to behave as bucket brigade, we would look at either *Option 4* or *Option 3*. If $a*c/(a + c - b) \ge b*d/(b + d - a)$ then the assignment is *W2-W1*, otherwise it is *W1-W2*.

Examples:

k_{ij}	S1	S2
W1	15	20 ^{i}
W2	18	25

k_{ij}	S1	S2
W1	7	9 ^{i}
W2	8	100

Case 10 analysis: Recall that for Case 10 $(k_{12} \le k_{21} \le k_{11} \le k_{22})$ we have the production rates given in Table 1 and that $a \le b \le c \le d$.

Table 61: Case 10 workers' production rates

k_{ij}	S1	S2
W1	c	a
W2	b	d

Option 1: $Output^1 = min\ (k_{11}, k_{22}) = min\ (c, d) = c.$

Option 2: $Output^2 = min\ (k_{21}, k_{12}) = min\ (b, a) = a.$

Option 3: As it is not allowed for the second worker to be idle, in this case *W2,* the output can be calculated as $Output^3 = k_{22}*k_{21}/(k_{22} + k_{21} - k_{11}) = b*d/(b+ d - c).$

Option 4: $Output^4 = min\ (b, a) = min\ (k_{21}, k_{12}) = a.$

Option 5: As $c \le d$ and $b \le c$ $(k_{21} \le k_{11})$ it is beneficial for the second worker to be idle, thus $Output^5 = min\ (k_{11}, k_{22}) = min\ (c, d) = c.$

Option 6: W2 will be assigned to both stations, as $a \le d$ and it is beneficial for *W1* to be idle, thus $Output^6 = k_{22}*k_{21}/(k_{22} + k_{21} - k_{12}) = b*d/(b + d - a).$

Summary: *Option 3*is better than *Option 6* as $b*d/(b + d - a) \le b*d/(b+ d - c)$ *or* $a \le c$. *Option 4* and *Option 2* result in the lowest output. We can show that $a \le b*d/(d + b - a)$ as it reduces to $(d-a)(b-a) \ge 0$, which is true. Now, we have to prove that *Option 5* (or *Option 1*) results in a better output than *Option 3*. Thus, we have to show that $c \ge b*d/(b+ d - c)$. After rearranging the terms the inequality reduces to $(d - c) * (c - b) \ge 0$ which is true as all terms are positive. Thus, *Option 5* (or *Option 1*) is optimal.

Table 62: Case 10 output rates

	Output rate	*Optimal*
Option 1	*c*	*c*
Option 2	*a*	
Option 3	*b*d/(b+ d - c)*	
Option 4	*a*	
Option 5	*c*	*c*
Option 6	*b*d/(b + d – a)*	

Examples:

k_{ij}	*S1*	*S2*
W1	*9*	*7*
W2	*8*	*10* ^{i}

k_{ij}	*S1*	*S2*
W1	*9*	*7*
W2	*8*	*100* ^{i}

Case 11 analysis: Recall that for Case *11* ($k_{11} \le k_{22} \le k_{12} \le k_{21}$) we have the production rates given in Table 1 and that $a \le b \le c \le d$.

Table 63: Case 11 workers' production rates

k_{ij}	*S1*	*S2*
W1	*a*	*c*
W2	*d*	*b*

Option 1: $Output^1 = min\ (k_{11}, k_{22}) = min\ (a, b) = a.$

Option 2: $Output^2 = min\ (k_{21}, k_{12}) = min\ (d, c) = c.$

Option 3: $Output^3 = k_{22}*k_{21}/(k_{22} + k_{21} – k_{11}) = b*d/(d + b – a).$

Option 4: As $c \le d$, $Output^4 = min\ (c, d) = min\ (k_{21}, k_{12}) = c.$

Option 5: As $b \ge a$, the second worker in the order would never be idle, thus $Output^5 = Output^3 = b*d/(d + b – a).$

Option 6: As $c \le d$ there is no gain if the second worker is idle, thus $Output^6 = Output^4 = c.$

Summary: It is clear that $Output^1 \le$ Output2. We have to show that *Option 4* (or *6)* results in the highest throughput. So we have to show that $c \ge b*d/(d + b – a)$. This inequality reduces to $d*(c – b) + c*(b - a) \ge 0,$ which is true as all the terms are positive. Thus the optimal assignment is *W2-W1*.

Table 64: Case 11 output rates

	Output rate	*Optimal*
Option 1	*a*	
Option 2	*c*	*c*
Option 3	*b*d/(d + b – a)*	
Option 4	*c*	*c*
Option 5	*b*d/(d + b – a)*	
Option 6	*c*	*c*

Examples:

k_{ij}	*S1*	*S2*
W1	*7*	*9*
W2	*i* *10*	*8*

k_{ij}	*S1*	*S2*
W1	*7*	*9*
W2	*i* *100*	*8*

Case 12 analysis

Recall that for Case *12* ($k_{12} \le k_{22} \le k_{11} \le k_{21}$) we have the production rates given in Table 1 and that $a \le b \le c \le d$.

Table 65: Case 12 workers' production rates

k_{ij}	*S1*	*S2*
W1	*c*	*a*
W2	*d*	*b*

Option 1: $Output^1 = min\ (k_{11}, k_{22}) = min\ (c, b) = b.$

Option 2: $Output^1 = min\ (k_{21}, k_{12}) = min\ (d, a) = a.$

Option 3: $Output^3 = min\ (c, b) = min\ (k_{11}, k_{22}) = b.$

Option 4: $Output^4 = min\ (d, a) = min\ (k_{21}, k_{12}) = a.$

Option 5: $Output^5 = Output^3 = b,$ as $b \ge a$ we would not gain if the first worker is assigned to the second task or if the second worker is idle.

Option 6: $Output^6 = k_{22}*k_{21}/(k_{22} + k_{21} – k_{12}) = b*d/(d + b – a).$

We can show that $b \ge b*d/(d + b – a).$ This inequality reduces to $b*(b - a) \ge 0$ which is true, as all terms are positive. Thus, *Option 3* (or *5*) is optimal. So the optimal assignment is *W1-W2*.

Table 66: Case 12 Output rates

	Output rate	*Optimal*
Option 1	*b*	*b*
Option 2	*a*	
Option 3	*b*	*b*
Option 4	*a*	
Option 5	*b*	*b*
Option 6	*b*d/(d + b – a)*	

Examples:

k_{ij}	*S1*	*S2*
W1	*i* 9	7
W2	10	8

k_{ij}	*S1*	*S2*
W1	*i* 9	7
W2	100	8

Summary and analysis of results is presented in chapter 3.

POSSIBLE DYNAMICS FOR TWO WORKERS AND DUPLICATE STATIONS

The assumptions are the same as presented in Chapter 3 for the two worker and two station production line except that workers are never idle due to duplicate tooling.

Duplicate tooling Case 3 analysis

The workers' production rates for Case *3* are presented in, Table 1 also recall that $a \leq b \leq c \leq d$. For Case *3*, it is reasonable to only analyze *Options 1* and *4*. If we look at *Option 2,* we can conclude that it is not beneficial to duplicate S*2* when the workers are ordered *W1 – W2* since $k_{22} \geq k_{11}$ (based on our assumption that $k_{22}=c \geq k_{11} = a$). The same reasoning applies for *Option 3*. For *Case 3*, Options *1* and *4* are beneficial as output will be higher if we have duplicate tooling. Thus we need to compare these output rates to determine the optimal output. We determine the output rates based on the calculations presented in Section 4, thus:

Option 1: Output$^{1} = (k_{21}+k_{11})/(k_{21}/k_{22}+1) = (a + d)/(d/c+1)$.

Option 4: Output$^{4} = (k_{22}+k_{12})/(k_{22}/k_{21}+1) = (b + c)/(c/d+1)$.

When we compare $(a + d)/(d/c+1)$ and $(b + c)/(c/d+1)$, it reduces to the relationship between $a*c$ and $b*d$. Thus, Option *4* is optimal because $b*d$ is always greater than $a*c$.

Duplicate tooling Case 4 analysis

The workers' production rates for Case *4* are presented in, Table 1 also recall that $a \leq b \leq c \leq d$. For Case *4*, it is reasonable to only analyze *Options 1* and *4*. If we look at *Option 3,* we can conclude that it is not beneficial to duplicate S*1* when the workers are ordered *W2 -W1* since $k_{21} \geq k_{12}$ (based on our assumption that $k_{21}=d \geq k_{12} = a$). The same reasoning applies for *Option 2*. For *Case 4*, Options *1* and *4* are beneficial as output will be higher if we have duplicate tooling. Thus we need to compare these output rates to determine the optimal output. We determine the output rates based on the calculations presented in Section 4, thus:

Option 1: Output$^{1} = (k_{21}+k_{11})/(k_{21}/k_{22}+1) = (b + d)/(d/c+1)$.

Option 4: Output$^{4} = (k_{22}+k_{12})/(k_{22}/k_{21}+1) = (a + c)/(c/d+1)$.

When we compare $(b + d)/(d/c+1)$ and $(a + c)/(c/d+1)$, it reduces to the relationship between $c*b$ and $a*d$. If $c*b > a*d$ it is optimal to order the workers *W1 - W2* and duplicate *S1*. Otherwise, it is optimal to order the workers *W2 - W1* and duplicate *S2*.

Duplicate tooling Case 5 analysis

The workers' production rates for Case *5* are presented in Table 1, also recall that $a \leq b \leq c \leq d$. For Case *5*, it is reasonable to only analyze *Options 1* and *3*. *Options 2* and *4* would not result in higher throughput. If we look at *Option 2,* we can conclude that it is not beneficial to duplicate S*2* when the workers are ordered *W1 –W2* since $k_{22} \geq k_{11}$ (based on our assumption that $k_{22}=c \geq k_{11} = a$). The same reasoning applies for *Option 4*. For *Case 5*, Options *1* and *3* are beneficial. Thus we need to compare these output rates to determine the optimal output. We determine the output rates based on the calculations presented in Section 4, thus:

Option 1: Output$^1 = (k_{21}+k_{11})/(k_{21}/k_{22}+1) = (a + b)/(b/c+1)$.

Option 3: Output$^3 = (k_{11}+k_{21})/(k_{11}/k_{12}+1) = (a + b)/(a/d+1)$.

When we compare $(a + b)/(b/c+1)$ and $(a + b)/(a/d+1)$, it reduces to the relationship between $1/(b/c+1)$ and $1/(a/d+1)$. Considering the relationship of workers' production rates, we can say that $b/c \geq a/d$, and consequently $1/(b/c+1) \leq 1/(a/d+1)$. Thus, *Option 3* is optimal and it is optimal to order the workers *W2 – W1* and duplicate *S1*.

Duplicate tooling Case 6 analysis

The workers' production rates for Case *6* are presented in Table 1, also recall that $a \leq b \leq c \leq d$. For Case *6*, it is reasonable to only analyze *Options 2* and *4*. *Options 1* and *3* would not result in higher throughput. If we look at *Option 1,* we can conclude that it is not beneficial to duplicate S*1* when the workers are ordered *W1 –W2* since $k_{11} \geq k_{22}$ (based on our assumption that $k_{11}=d \geq k_{22} = c$). The same reasoning applies for *Option 3*. For *Case 6*, we will look at *Options 2* and *4* and compare these output rates to determine the optimal output. We determine the output rates based on the calculations presented in Section 4, thus:

Option 2: Output$^2 = (k_{12}+k_{22})/(k_{12}/k_{11}+1) = (a + c)/(a/d+1)$.

Option 4: Output$^4 = (k_{22}+k_{12})/(k_{22}/k_{21}+1) = (a + c)/(c/b+1)$.

When we compare $(a + c)/(a/d+1)$ and $(a + c)/(c/b+1)$, it reduces to the relationship between $1/(a/d+1)$ and $1/(c/b+1)$. Considering the relationship of workers' production rates, we can say that $a/d \leq c/b$ $(a/d \leq 1$ and $c/b \geq 1)$, and consequently $1/(a/d+1) \geq 1/(c/b+1)$. Thus, *Option 2* is optimal and it is optimal to order the workers *W1 – W2* and duplicate *S2*.

Duplicate tooling Case 7 analysis

The workers' production rates for Case *7* are presented in Table 1, also recall that $a \leq b \leq c \leq d$. For Case *7*, it is reasonable to only analyze *Options 1* and *3*. If we look at *Option 2,* we can conclude that it is not beneficial to duplicate S*2* when the workers are ordered *W1 –W2* since $k_{22} \geq k_{11}$ (based on our assumption that $k_{22}=b \geq k_{11} = a$). The same reasoning applies for *Option 4.* For *Case 7*, Options *1* and *3* are beneficial and we need to compare these corresponding output rates to determine the optimal output. We determine the output rates based on the calculations presented in Section 4, thus:

Option 1: Output$^1 = (k_{21}+k_{11})/(k_{21}/k_{22}+1) = (a + c)/(c/b+1)$.

Option 3: Output$^3 = (k_{11}+k_{21})/(k_{11}/k_{12}+1) = (a + c)/(a/d+1)$.

When we compare $(a + c)/(c/b+1)$ and $(a + c)/(a/d+1)$, it reduces to the relationship between $1/(c/b+1)$ and $1/(a/d+1)$. Considering the relationship of workers' production rates, we can say that $a/d \leq c/b$ $(a/d \leq 1$ *and* $c/b \geq 1)$, and consequently $1/(a/d+1) \geq 1/(c/b+1)$. Thus, *Option 3* is optimal and it is optimal to order the workers *W2 – W1* and duplicate *S1*.

Duplicate tooling Case 8 analysis

The workers' production rates for Case *8* are presented in Table 1, also recall that $a \leq b \leq c \leq d$. For Case *8*, it is reasonable to only analyze *Options 2* and *4*. If we look at *Option 1,* we can conclude that it is not beneficial to duplicate S*1* when the workers are ordered *W1 –W2* since $k_{11} \geq k_{22}$ (based on our assumption that $k_{11}=d \geq k_{22} = b$) since that station would not be used. The same reasoning applies for *Option 3.* For *Case 8*, Options *2* and *4* are beneficial and we need to compare these corresponding output rates to determine the optimal output. We determine the output rates based on the calculations presented in Section 4, thus:

Option 2: Output$^2 = (k_{12}+k_{22})/(k_{12}/k_{11}+1) = (a + b)/(a/d+1)$.

Option 4: Output$^4 = (k_{22}+k_{12})/(k_{22}/k_{21}+1) = (a + b)/(b/c+1)$.

When we compare $(a + b)/(a/d+1)$ and $(a + b)/(b/c+1)$, it reduces to the relationship between $1/(a/d + 1)$ and $1/(b/c + 1)$. Considering the relationship of workers' production rates, we can say that $b/c \geq a/d$, and consequently $1/(b/c+1) \leq 1/(a/d+1)$. Thus, *Option 2* is optimal and it is optimal to order the workers *W1 – W2* and duplicate *S2*.

Duplicate tooling Case 9 analysis

The workers' production rates for Case *9* are presented in Table 1, also recall that $a \leq b \leq c \leq d$. For Case *9*, it is reasonable to only analyze *Options 1* and *3*. If we look at *Option 2,* we can conclude that it is not beneficial to duplicate S*2* when the workers are ordered *W1 –W2* since $k_{22} \geq k_{11}$ (based on our assumption that $k_{22}=d \geq k_{11} = a$) since that station would not be used. The same reasoning applies for *Option 4*. For *Case 9*, Options *1* and *3* are beneficial and we need to compare these corresponding output rates to determine the optimal output. We determine the output rates based on the calculations presented in Section 4, thus:

Option 1: Output^{1} $= (k_{21}+k_{11})/(k_{21}/k_{22}+1) = (a + b)/(b/d+1)$.

Option 3: Output^{3} $= (k_{11}+k_{21})/(k_{11}/k_{12}+1) = (a + b)/(a/c+1)$.

When we compare $(a + b)/(b/d+1)$ and $= (a + b)/(a/c+1)$, it reduces to the relationship between $1/(b/d + 1)$ and $1/(a/c + 1)$. If $1/(b/d + 1) \geq 1/(a/c + 1)$ (or $a*d \geq b*c$), *Option 1* is optimal, otherwise *Option 3* is optimal, in which case workers should be ordered *W2 – W1* and duplicate *S1*.

Duplicate tooling Case 10 analysis

The workers' production rates for Case 10 are presented in Table 1, also recall that $a \leq b \leq c \leq d$. For Case 10, it is reasonable to only analyze *Options 1* and *4*. If we look at *Option 3,* we can conclude that it is not beneficial to duplicate S*1* when the workers are ordered *W2 -W1* since $k_{21} \geq k_{12}$ (based on our assumption that $k_{21}=b \geq k_{12} = a$). The same reasoning applies for *Option 2*. For *Case 10*, Options *1* and *4* are beneficial as output will be higher if we have duplicate tooling. Thus we need to compare these output rates to determine the optimal output. We determine the output rates based on the calculations presented in Section 4, thus:

Option 1: Output^{1} $= (k_{21}+k_{11})/(k_{21}/k_{22}+1) = (b + c)/(b/d + 1)$.

Option 4: Output^{4} $= (k_{22}+k_{12})/(k_{22}/k_{21}+1) = (d + a)/(d/b + 1)$.

When we compare $(b + c)/(b/d + 1)$ and $(d + a)/(d/b + 1)$, it reduces to the relationship between $c*d$ and $a*b$. Thus, Option *1* is optimal since $c*d$ is always greater than $a*b$.

Duplicate tooling Case 11 analysis

The workers' production rates for Case 11 are presented in Table 1, also recall that $a \leq b \leq c \leq d$. For Case 11, it is reasonable to only analyze *Options 1* and *4*. If we look at *Option 2,* we can

conclude that it is not beneficial to duplicate S*2* when the workers are ordered *W1 –W2* since $k_{22} \geq k_{11}$ (based on our assumption that $k_{22}=b \geq k_{11} = a$). The same reasoning applies for *Option 3*. For *Case 11*, Options *1* and *4* are beneficial and we need to compare these output rates to determine the optimal output. We determine the output rates based on the calculations presented in Section 4, thus:

Option 1: $Output^1 = (k_{21}+k_{11})/(k_{21}/k_{22}+1) = (a + d)/(d/b + 1)$.

Option 4: $Output^4 = (k_{22}+k_{12})/(k_{22}/k_{21}+1) = (b + c)/(b/d + 1)$.

When we compare $(a + d)/(d/b + 1)$ and $(b + c)/(b/d + 1)$, it reduces to the relationship between $a*b$ and $c*d$. Thus, Option *4* is optimal since $c*d$ is always greater than $a*b$.

Duplicate tooling Case 12 analysis

The workers' production rates for Case *12* are presented in Table 1, also recall that $a \leq b \leq c \leq d$. For Case *12*, it is reasonable to only analyze *Options 2* and *4*. If we look at *Option 1,* we can conclude that it is not beneficial to duplicate S*1* when the workers are ordered *W1 –W2* since $k_{11} \geq k_{22}$ (based on our assumption that $k_{11}=c \geq k_{22} = b$) since that station would not be used. The same reasoning applies for *Option 3*. For *Case 12*, Options *2* and *4* are beneficial and we need to compare these corresponding output rates to determine the optimal output. We determine the output rates based on the calculations presented in Section 4, thus:

Option 2: $Output^2 = (k_{12}+k_{22})/(k_{12}/k_{11}+1) = (a + b)/(a/c + 1)$.

Option 4: $Output^4 = (k_{22}+k_{12})/(k_{22}/k_{21}+1) = (a + b)/(b/d + 1)$.

When we compare $(a + b)/(a/c + 1)$ and $(a + b)/(b/d + 1)$, it reduces to the relationship between $1/(a/c + 1)$ and $1/(b/d + 1)$. If $1/(a/c + 1) \geq 1/(b/d + 1)$ (or $c*b \geq a*d$), *Option 2* is optimal. Otherwise, *Option 4* is optimal and workers should be ordered *W2 – W1* and duplicate *S2*.

A summary and analysis of these results is presented Chapter 3.

PARALLEL LINE COMPARISONS

If we assume that both stations have duplicate tooling, we can look at the system as two parallel production lines. By two parallel lines we mean that each worker works at both stations in sequential order and finishes parts independently of the other worker. It can be shown that the

throughput obtained from two parallel lines is at most as good as the output produced when we have optimally ordered the workers and have chosen the correct station to duplicate. In both scenarios workers are never idle and it is beneficial not to duplicate both stations.

For Case 1, we know that either *Option 1* or *Option 4* would result in the optimal throughput. So, if *Option 1* is optimal we can show that inequality $(c+a)/(c/d+1) \geq a*b/(a+b)+c*d/(c+d)$ is true, which states that the output from two parallel lines is at most as good as output from a line with one station with duplicate tooling, reduces to $bc \leq ad$, which us true under the assumption that Option *1* is optimal (see Case *1* analysis section 2.1). If *Option 4* is optimal we can show that inequality $(d+b)/(d/c+1) \geq a*b/(a+b)+c*d/(c+d)$ is true, and reduces to $ad \leq bc$, which is true under the assumption that Option 4 is optimal.

For Cases 2 and 3, we have to show that $(b+c)/(c/d+1) \geq a*b/(a+b)+c*d/(c+d)$ is true. This inequality reduces to $1/(c/d + 1) \geq 1/(b/a + 1)$. This inequality is true since $c \leq d$, or $c/d \leq 1$, and also $b \geq a$, or $b/a \geq 1$.

For Case 4, we know that either Option *1* or Option *4* would result in the optimal throughput. So, if Option *1* is optimal we can show that inequality $(b+d)/(d/c+1) \geq a*b/(a+b)+c*d/(c+d)$, which states that the output from two parallel lines is at most as good as output from a line with one station with duplicate tooling, reduces to $bc \geq ad$, which us true under the assumption that Option *1* is optimal. If Option *4* is optimal, we have to show that $(a+c)/(c/d+1) \geq a*b/(a+b)+c*d/(c+d)$ is true. This inequality reduces to $ad \geq bc$, which is true under the above assumption.

For Case 5, we have to show that $(a+b)/(a/d+1) \geq a*d/(a+d)+c*b/(c+b)$. This inequality reduces to $d*b \geq a*c$, which is true.

For Cases 6 and 7, it can be shown that $a*d/(a+d)+ c*b/(c+b) \leq (a+c)/(a/d+1)$. Applying simple algebra, the expression reduces to $a*b \leq c*d$. Based on the assumed relationships of *a, b, c, d* we can say that the output rates from two parallel lines for cases *6* and *7* are always lower than the optimal worker ordering combined with the best choice of duplicate tooling.

For Case 8, we have to show that $(a+b)/(a/d+1) \geq a*d/(a+d)+c*b/(c+b)$. This inequality reduces to $b*d \geq a*c$ which is true based on our assumption.

For Case 9, we have to show that $(a+b)/(b/d+1) \geq a*c/(a+c)+d*b/(d+b)$ is true if *Option 1* is optimal, or $(a+b)/(a/c+1) \geq a*c/(a+c)+d*b/(d+b)$ if *Option 3* is optimal. The first inequality

reduces to $a*d \geq b*c$ (see Section 4.9), which is true if *Option 1* is optimal. The second inequality reduce to $b*c \geq a*d$, which is true if *Option 3* is optimal.

For Cases 10 and 11, we have to show that $(b+c)/(b/d+1) \geq a*c/(a+c)+d*b/(d+b)$. This inequality reduces to $c*d \geq a*b$, which is always true under the above assumption.

For Case 12, we have to show that $(a+b)/(a/c+1) \geq a*c/(a+c)+d*b/(d+b)$ if *Option 2* is optimal. This inequality reduces to $c*b \geq a*d$, which is true if *Option 2* is optimal (see Section 4.12). Also, we have to show that $(a+b)/(b/d+1) \geq a*c/(a+c)+d*b/(d+b)$ if *Option 4* is optimal. This inequality reduces to $a*d \geq c*b$, which is true if *Option 4* is optimal (see Section 4.12).

In summary, we showed that the output from two parallel lines is at most as good as output from a line with one station with duplicate tooling. The break point analysis and the summary table are given in Chapter 3.

APPENDIX B

ADDITIONAL NUMERICAL EXAMPLES

Example 1: Three Workers Six Stations

Table 67: Three Workers Six Stations: Steady state production rates

k_{ij}	S1	S2	S3	S4	S5	S6
W1	11	12	15	7	11	11
W2	8	12	12	12	8	7
W3	11	11	13	9	10	11

Table 68: Solution Three Workers Six Stations

x_{ij}	S1	S2	S3	S4	S5	S6
W1					0.49	0.51
W2		0.02	0.47	0.47	0.04	
W3	0.51	0.49				

Objective: 5.6 parts/hour

Example 2: Three Workers Four Stations

Table 69: Three Workers Four Stations: Steady state production rates

k_{ij}	S1	S2	S3	S4
W1	9	10	12	11
W2	12	10	9	10
W3	13	9	10	12

Table 70: Solution: Three Workers Four Stations

x_{ij}	S1	S2	S3	S4
W1		0.55	0.45	
W2	0.7	0.3		
W3			0.3	0.7

Objective: 8.4 parts/hour

APPENDIX C

Table 71: Data set: Learning and forgetting data

	p_{ij}											
W\S	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
1	884.4	11.12	607.1	994.2	798.4	937.7	1118	418	1081	14.95	575.7	11.17
2	61.74	0	322.1	448.1	84.96	1000	965.5	171.1	946.2	156.4	1186	118.9
3	255.8	636.4	115.2	102.6	771.5	223.7	33.51	186.4	0	992	708.3	110
4	261.3	512.1	125.1	0	983.1	217.8	140.4	871.1	847.7	817.2	59.67	103
5	842.6	838.3	836.4	0	326	595.1	553.3	1107	184.5	62.73	917.5	426
6	59.66	14.24	311.8	93.92	1080	64.52	18.53	1011	934.5	0	992	255.8
	r_{ij}											
W\S	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
1	161.5	566.9	285.5	411.1	196.8	168.6	447.9	215.5	740.2	914.5	59.76	373.2
2	67.28	76.74	270.1	863.2	87.46	738.6	283.8	630.9	263.7	314.3	74.46	114.8
3	46.14	721.2	396.2	694.6	918.4	97.84	491.8	29.8	120.1	322.6	47.55	6.08
4	938.9	62.46	42.99	8.18	235.8	199	538.6	370.5	81.01	217.7	446.6	141.1
5	11.13	97.74	873.8	141.4	157	457.1	519.4	441.7	362	74.15	72.02	309.9
6	868.1	380.9	209	12.97	83.91	53.08	513.8	40.39	542.9	56.44	627.5	55.36

Table 71: (continued)

	k_{ij}											
W\S	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
1	25.02	28.29	29.25	29.16	28.22	35.79	33.1	35.76	37.46	30.57	27.48	37.92
2	25.93	32.85	34.88	26.1	35.92	32.34	28.89	34.03	35.28	30.86	25.53	31.49
3	38.5	25.79	37.48	33.65	28.18	37.77	30.24	39.3	29.52	38.83	27.09	32.9
4	38.63	34.56	36.99	37.79	37.38	36.4	27.18	35.28	35.34	27.56	34.61	37.11
5	26.86	33.81	38.1	36.04	28.61	27.62	30.19	31.99	25.69	31.81	30.67	38.43
6	36.67	31.23	30.82	26.52	35.57	28.65	30.13	26.78	31.33	33.42	34.79	30.59
	α_{ij}											
W\S	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
1	4.71	1.24	4.4	3.78	0.27	0.77	0.25	3.7	0.62	1.86	3.09	4.09
2	1.01	1.16	3.41	4.6	1.63	3.13	0.23	3.53	3.97	0	4.38	3.69
3	1.36	3.98	4.42	0.12	3.82	2.02	1.73	0.59	4.93	2.1	3.6	1.87
4	0.86	0	0.74	0	4.22	2.94	3.11	2.44	0.33	1.73	1.01	4.32
5	1.36	4.32	1.74	4.73	0.17	2.81	4.01	2.44	0.33	1.73	1.01	4.32
6	3.74	1.36	1.57	2.33	1.63	4.13	2.49	1.25	1.5	4.48	2.11	3.72

Bucket Brigades Pseudo code and Flowchart

Input data: The length of production horizon**,** Number of workers, Number of stations, Worker order and starting positions, Workers' individual learning and forgetting characteristics.

Possible phases:

A. Worker starts work at the station.

B. Worker finishes work at the station.

Question 1 (Q1): Is the queue empty?

No. Take the part. Go to **A**.

Yes**. Question 2 (Q2):** Is the worker at the last station?

Yes. Go to **D.** (D. Go to the previous station.)

No**. Question 3 (Q3):** Is the next station empty?

Yes. Go to the next station. Go to **A**.

No. Go to the next station. Go to **E.** (**E**. Wait in the queue).

C. Part is taken from the worker.

Q1: Is anybody in the queue?

Yes. Go to E.

No. **Question 5 (Q5):** Is worker at the first station?

Yes. Go to **E.**

No. Go to **D.**

D. Go to the previous station.

Question 4 (Q4): Is station empty?

Yes. **Q5:** Is worker at the first station?

Yes. Take a new part. Go to **A.**

No. Go to **D.**

No. Take a part from a worker. Go to **A.**

E. Wait in the queue.

Assign the worker to the queue at the station.

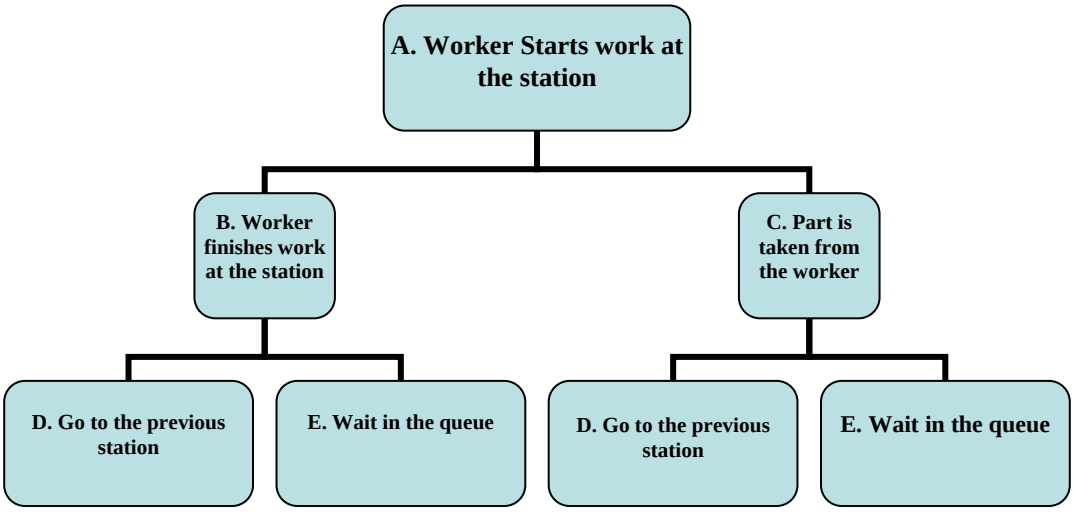


Figure 9: Possible phases in a bucket brigade system

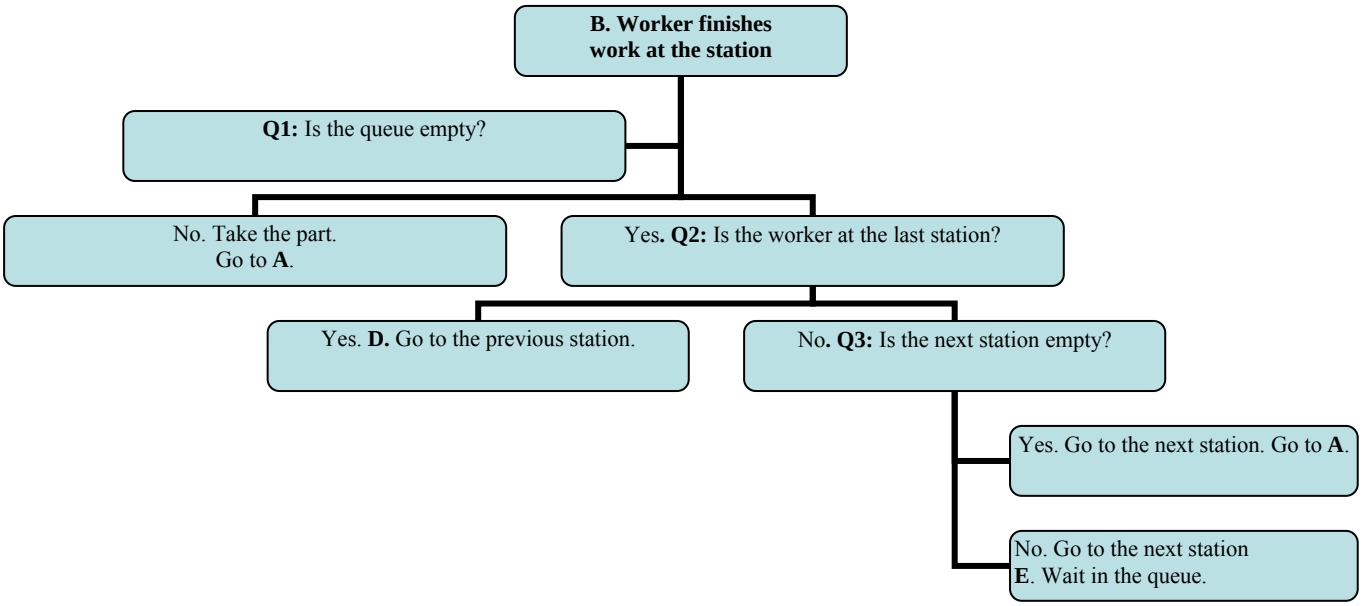


Figure 10: Event B: Worker finishes work at the station

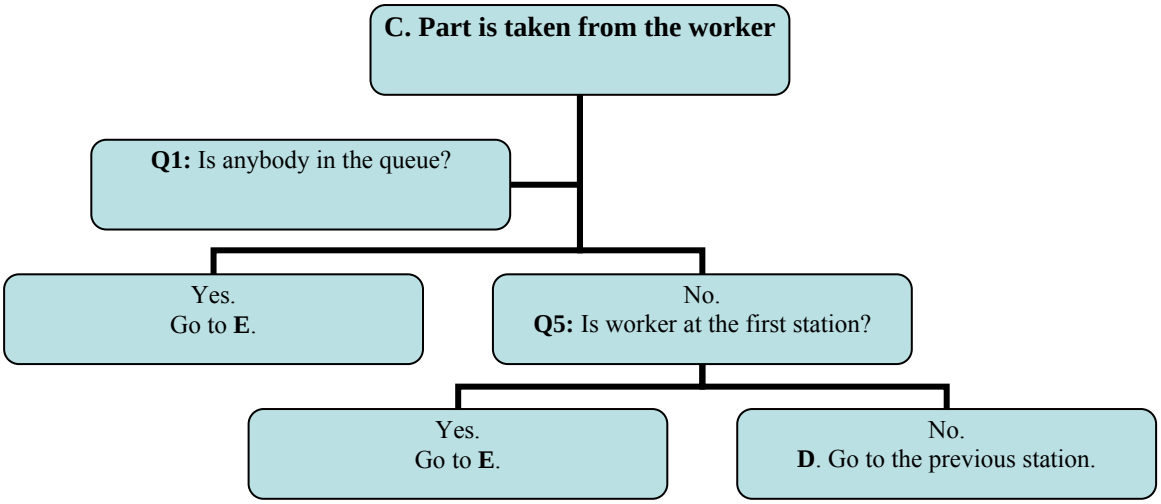


Figure 11: Event C: Part is taken form the worker

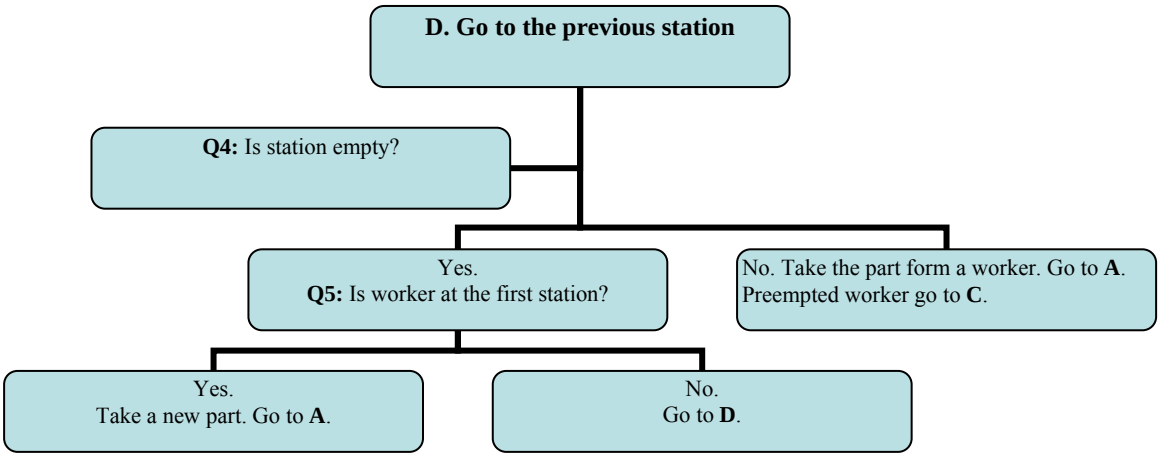


Figure 12: Event D: Backward phase

APPENDIX D

PAIRWISE EXCHANGE HEURISTIC

Input data: total production time *t*, number of workers *n*, number of stations in a serial production line *m* (including the starting buffer capacity), learning and forgetting parameters (*k*, *p*, *r* and *a*). The algorithm steps are as follows:

```
1  generate an initial assignment;
2  calculate the total production, BestOUT, for the initial assignment;
3  ind= 1;  (ind is an improvement indicator)
4  while ind = 1
5        ind = 0
6        for t ← 0 to t-1
7              for i ← 0 to n-1
8                    for j ← i+1 to n-1
9                          exchange x[i][t] ↔ x[j][t];
10                         calculate OUT;
11                         if (Out > BestOUT)
12                               set ind = 1 and record besti, bestj and best;
13                         exchange x[i][t] ↔ x[j][t];
14       note the value of the best solution found, BestOUT
15 end;
```

APPENDIX E

SIMULATED ANNEALING ALGORITHM

Notation and Assumptions:

- *tpr* - starting temperature;
- α is a temperature reduction function parameter;
- *tpr* = α**tpr*; annealing schedule or temperature reduction function;
- *tpr* > ε stopping condition and the last temperature level;
- *x[i][t]* – initial solution, *i=1,2,...n, t=1,2,...,p;* For example vector *[2 3 0 1 4 5 7 6]* means that worker *0* does task *2*, worker *1* task *3*, worker *3* task *0*, etc…. during the given period;
- *bestx[i][t]* – best solution found so far;
- *N(x[i][t])* - all points in *x[i][t]* neighborhood. A neighborhood is defined as *N(x[i][t])* which are all adjacent solutions as any two workers assignments are exchanged during the given period;
- A move is defined as the exchange of two worker's assignments;
- *OUT* - initial objective function value;
- Number of iterations at each temperature level was defined as *MAX=nrep*;
- Number of iterations at each temperature level without an improvement = NOIMPROV;
- *Opt* - previously accepted solution's objective function value*;*
- *Best* - best objective function value obtained;
- *LocalOpt* - current solution's objective function value;
- *DELTA = LocalOpt – Opt*, the difference in the current solution and the previous solution.

A pseudocode of the algorithm is presented and the C code is attached in the Appendix. The algorithm steps are as follows:

Pseudocode

Input data: total production time *t*, number of workers *n*, number of stations in a serial production line *m* (buffer capacity = *0*), steady-state production rate *k*:

- Calculate the total production, *OUT,* for the initial assignment;
- *Best = 0*;
- *Opt=OUT*;
- Assign a starting temperature *tpr*;
- ***while*** *(tpr > ε)*
- ***while*** *(count < MAX)*
- Generate two random numbers in the range of *1* to *n (workers);*
- Generate a random number in the range of *1* to *p (time period);*
- ***while*** (*count < MAX and noimprov < NOIMPROV*);

  Exchange two randomly selected worker's assignments during the selected period;
- Calculate *OUT*;
- *LocalOpt = Out*;
- *DELTA = LocalOpt – Opt*;
- ***if*** *(DELTA < 0)*
- *Opt = LocalOpt*; (accept the move)
- *bestx[i][t] = x[i][t]*; (accept the solution for all *i, t*)
- *count += 1*; (increment the count)
- ***if*** (*OPT > BEST*) (remember the best solution)
- *BEST = OPT;*
- *bestx[i][t] = x[i][t]* for all *i*;
- ***else***
- *r = ((double)rand() / (double)(RAND_MAX+1));*
- *ee = exp(-DELTA/tpr);*
- ***if*** (*r < ee*)
- *Opt = LocalOpt*; (accept the move if true)

- *count +=1*; (increment the count)
- ***else***
- switch back two assignments;
- ***end***
- *tpr = α*t pr*; (temperature reduction function)
- **end**

APPENDIX F

ADDITIONAL DATA SETS AND RESULTS

Table 72: MIP Instance L1: Simulated Annealing Results

SIMULATED ANNEALING RESULTS				
	Assignment 1		**Assignment 2**	
Seed	**Solution**	**CPU Time**	**Solution**	**CPU Time**
87	**408**	2.782	**408**	2.768
39	**408**	2.768	**408**	2.767
678	**408**	2.767	**408**	2.783
890	**408**	2.783	**408**	2.783
3290	**408**	2.799	**408**	2.97
7654	**408**	2.767	**408**	2.814
543	**408**	2.783	**408**	2.783
119	**408**	2.767	**408**	2.799
7	**408**	2.783	**408**	2.783
1928	**408**	2.768	**408**	2.783
MAX	**408**	2.799	**408**	2.97
MIN	**408**	2.767	**408**	2.767
AVERAGE	**408**	2.7767	**408**	2.8033
	Assignment 3		**Assignment 4**	
Seed	**Solution**	**CPU Time**	**Solution**	**CPU Time**
87	**408**	2.783	**408**	2.814
39	**408**	2.783	**408**	2.846
678	**408**	2.752	**408**	2.845
890	**408**	2.751	**408**	2.861
3290	**408**	2.799	**408**	2.893
7654	**408**	2.752	**408**	2.939
543	**408**	2.783	**408**	2.924
119	**408**	2.783	**408**	2.891
7	**408**	2.845	**408**	2.813
1928	**408**	2.877	**408**	2.969
MAX	**408**	2.877	**408**	2.969
MIN	**408**	2.751	**408**	2.813
AVERAGE	**408**	2.7908	**408**	2.8795

Table 73: Instance L1: Workers production rates

k_{ij}	**S1**	**S2**	**S3**	**S4**	**S5**	**S6**	**S7**	**S8**
W1	20	33	40	20	33	40	35	35
W2	28	24	24	41	38	20	28	35
W	33	21	27	27	20	41	32	36
W4	28	29	32	21	32	31	31	34
W5	38	37	37	32	44	21	43	23
W6	40	26	25	21	26	37	38	22
W7	38	26	30	37	41	39	25	32
W8	43	33	23	28	31	21	26	23

Table 74: MIP Instance L2: Simulated Annealing Results

Assignment 1			**Assignment 2**		**Assignment 3**		**Assignment 4**	
Seed	**Solution**	**CPU Time**	**Solution**	**CPU Time**	**Solution**	**CPU Time**	**Solution**	**CPU Time**
87	355	3.995	352	3.966	355	4.086	355	4.056
39	**356**	3.936	**356**	3.906	352	4.086	355	4.005
678	355	3.976	**356**	3.905	355	4.016	**356**	4.196
890	355	3.925	354	4.026	**356**	3.986	355	4.096
3290	**356**	4.006	**356**	3.946	**356**	4.015	355	4.026
7654	355	3.906	355	3.945	355	3.986	**356**	4.076
543	354	3.935	**356**	4.086	354	4.236	354	3.976
119	**356**	3.946	355	4.166	355	4.126	352	4.236
7	**356**	3.916	355	4.146	**356**	4.076	**356**	3.985
1928	**356**	3.975	355	4.206	355	4.026	355	4.076
MAX	**356**	4.006	**356**	4.206	**356**	4.236	**356**	4.236
MIN	354	3.906	352	3.905	352	3.986	352	3.976
AVERAGE	355.2	3.9516	355	4.0298	354.8	4.0639	354.5	4.0728

Table 75: Instance L2: Workers production rates

k_{ij}	S1	S2	S3	S4	S5	S6	S7	S8
W1	35.95	45	40.04	33.92	22.95	27.4	24.59	23.37
W2	22.42	23.39	19.42	27.06	29.09	26.02	20.66	19.81
W3	25.53	21.05	24.34	25.14	25.6	29.83	27.99	22.41
W4	24.92	31.03	24.3	27.83	22.15	37.32	27.39	22.41
W5	21.92	32.21	23.81	26.86	20.39	25.78	23.84	23.01
W6	28.62	27.24	22.99	29.5	30.51	33.12	28.74	23.6
W7	45	27.97	28.7	38.68	30.07	29.29	29.18	30
W8	25.58	45	23.04	30.77	43.15	34.27	31.41	31.53

Table 76: Instance L3: Simulated Annealing Results

SIMULATED ANNEALING RESULTS

Instance 3: 8 8 20

tpr = 5; t = 0.99 * t; Stopping condition t > 0.01; MAX iterations = 1000 Nonimprov = 1000.

	Assignment 1		**Assignment 2**		**Assignment 3**		**Assignment 4**	
Seed	**Solution**	**CPU Time**	**Solution**	**CPU Time**	**Solution**	**CPU Time**	**Solution**	**CPU Time**
87	594	29.457	593	29.753	594	30.581	594	29.767
39	594	29.848	594	29.658	**595**	31.846	593	29.815
678	594	30.317	594	29.674	594	30.033	**595**	30.095
890	**595**	29.785	**595**	30.596	594	30.08	593	30.283
3290	594	29.676	593	30.081	593	30.189	594	30.721
7654	594	29.722	**595**	31.252	593	30.424	**595**	29.065
543	**595**	29.973	594	30.955	594	30.486	593	29.815
119	594	30.16	594	29.955	593	29.892	**595**	30.783
7	593	29.77	593	30.143	594	29.783	593	31.033
1928	594	29.878	593	30.018	593	29.924	594	30.439
MAX	**595**	30.317	**594**	31.252	**595**	31.846	**595**	31.033
MIN	593	29.457	593	29.658	593	29.783	593	29.065
AVERAGE	593.9	29.8586	593.6	30.2085	593.7	30.3238	593.7	30.1816

Table 77: MINLP small data sets

MINLP small instances: 3 workers, 3 stations and 8 time periods					
DATA SET 1					
k values			**r values**		
35.95	45	40.04	1165.2	1205.93	1733.26
22.42	23.39	19.42	3.23	4.09	4.62
25.53	21.05	24.34	9.33	13.09	13.26
p values			**a values**		
3156.93	1183.46	1910.76	2.12	5	0
0	0	12.37	5	0	5
8.12	0	21.6	4.11	0	4.99
DATA SET 2					
k values			**r values**		
28.62	27.24	22.99	103.08	128.22	149.19
45	27.97	28.7	339.88	377.8	413.7
25.58	45	23.04	623.82	647.86	692.7
p values			**a values**		
158.64	240.15	0	0.9	0	0
227.44	533.83	370.83	0	0.11	0
1444.03	464.63	6181.6	0	4.21	0
DATA SET 3					
k values			**r values**		
25.6	29.83	27.99	18.19	20.03	21.44
22.15	37.32	27.39	54.08	54.18	54.22
20.39	25.78	23.84	79.31	87.78	94.25
p values			**a values**		
0	53.3	0	0	5	0
37	0	131.57	0.29	0.61	4.47
69.59	52.29	74.74	0.66	0.13	1.7
DATA SET 4					
k values			**r values**		
33.12	28.74	23.6	261.77	316.29	336.69
29.29	29.18	30	558.64	619.38	622.64
34.27	31.41	31.53	901.84	914.1	927.13
p values			**a values**		
336.92	511.21	595.06	4.3	0.01	0
499	908.78	1837.9	0.28	0	0
575.52	13165.66	1248.85	0.87	0.33	0.35

Table 78: Simulated Annealing Algorithm: Small Data Sets Solutions

Simulated Annealing Solutions: 3 workers, 3 stations and 8 time periods

Data Set 1: Initial Assignment 1

Seed	Solution	CPU Time
2345	**144**	5.156
55	**144**	5.187
87	**144**	5.203
771	**144**	5.282
3705	**144**	5.437
39	**144**	5.36
68	**144**	5.39
890	**144**	5.328
3290	**144**	5.375
7654	**144**	5.157
MAXIMUM	**144**	5.437
MINIMUM	**144**	5.156
AVERAGE	**144**	5.2875

Data Set 3: Initial Assignment 2

Seed	Solution	CPU Time
2345	**96**	3.578
55	**96**	3.484
87	**96**	3.422
771	**96**	3.687
3705	**96**	3.563
39	**96**	3.406
68	**96**	3.469
890	**96**	3.391
3290	**96**	3.484
7654	**96**	3.422
MAXIMUM	**96**	3.687
MINIMUM	**96**	3.391
AVERAGE	**96**	3.4906

Data Set 2: Initial Assignment 2

Seed	Solution	CPU Time
2345	**150**	5.062
55	**150**	5
87	**150**	5.078
771	**150**	5.016
3705	**150**	5.219
39	**150**	5.25
68	**150**	5.281
890	**150**	5.031
3290	**150**	5.063
7654	**150**	4.937
MAXIMUM	**150**	5.281
MINIMUM	**150**	4.937
AVERAGE	**150**	5.0937

Data Set 4: Initial Assignment 2

Seed	Solution	CPU Time
2345	**144**	3.625
55	**144**	3.61
87	**144**	3.64
771	**144**	3.625
3705	**144**	3.641
39	**144**	3.609
68	**144**	3.625
890	**144**	3.625
3290	**144**	3.625
7654	**144**	3.578
MAXIMUM	**144**	3.641
MINIMUM	**144**	3.578
AVERAGE	**144**	3.6203

Table 79: MINLP Large Instance L1

MINLP Large Instance 1: 8 workers, 8 stations and 20 time periods

a_{ij}								
worker/station	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1	2.12	5	0	0	5	5	1.44	0
2	5	0	5	0.68	5	1.43	0	5
3	4.11	0	4.99	0	0	5	0	1.32
4	1.81	1	0	0	0.29	0.61	4.47	0
5	0	2.27	0.07	0.94	0.66	0.13	1.7	0.92
6	0.9	0	0	2.56	5	4.3	0.01	0
7	0	0.11	0	1.05	0	0.28	0	0
8	0	4.21	0	0.03	2.44	0.87	0.33	0.35
p_{ij}								
worker/station	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1	3156.93	1183.46	1910.76	3704.23	0	0	0	0
2	0	0	12.37	28.37	29.84	28.87	0	37.86
3	8.12	0	21.6	0	0	53.3	0	9.92
4	27.03	293.97	11.54	186.47	37	0	131.57	98.8
5	78.68	22.72	38.17	6.08	69.59	52.29	74.74	70.37
6	158.64	240.15	0	321.95	1014.69	336.92	511.21	595.06
7	227.44	533.83	370.83	256.43	575.45	499	908.78	1837.9
8	1444.03	464.63	6181.6	619.71	853.67	575.52	13165.66	1248.85
r_{ij}								
worker/station	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1	1165.2	1205.93	1733.26	1839.29	0.15	1.3	2.62	2.92
2	3.23	4.09	4.62	6.31	6.4	6.83	8.07	8.3
3	9.33	13.09	13.26	16.23	18.19	20.03	21.44	23.65
4	26.84	37.56	40.5	51.98	54.08	54.18	54.22	60.8
5	65.29	69.63	70.96	77.51	79.31	87.78	94.25	101.85
6	103.08	128.22	149.19	184	221.38	261.77	316.29	336.69
7	339.88	377.8	413.7	498.77	499.6	558.64	619.38	622.64
8	623.82	647.86	692.7	722.73	813.89	901.84	914.1	927.13
k_{ij}								
worker/station	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1	35.95	45	40.04	33.92	22.95	27.4	24.59	23.37
2	22.42	23.39	19.42	27.06	29.09	26.02	20.66	19.81
3	25.53	21.05	24.34	25.14	25.6	29.83	27.99	22.41
4	24.92	31.03	24.3	27.83	22.15	37.32	27.39	22.41
5	21.92	32.21	23.81	26.86	20.39	25.78	23.84	23.01
6	28.62	27.24	22.99	29.5	30.51	33.12	28.74	23.6
7	45	27.97	28.7	38.68	30.07	29.29	29.18	30
8	25.58	45	23.04	30.77	43.15	34.27	31.41	31.53

Table 80: MINLP: Simulated Annealing Results for L1

Initial Assignment 1		
Seed	**Solution**	**CPU Time**
2345	**409**	671.91
55	**409**	675.00
87	**409**	675.86
771	**409**	683.16
3705	**410**	691.44
39	**409**	676.78
68	**409**	675.92
890	**410**	670.12
3290	**409**	674.89
7654	**407**	675.47
MAXIMUM	**410**	691.44
MINIMUM	**407**	670.12
AVERAGE	**409**	677.055

Table 81: MINLP: Simulated Annealing Results for L2

Initial Assignment 2		
Seed	**Solution**	**CPU Time**
2345	**408**	676.24
55	**410**	684.11
87	**410**	685.23
771	**412**	685.69
3705	**410**	683.94
39	**410**	680.64
68	**410**	681.92
890	**410**	679.03
3290	**412**	684.58
7654	**410**	684.03
MAXIMUM	**412**	685.69
MINIMUM	**408**	676.24
AVERAGE	**410**	682.54

BIBLIOGRAPHY

Anuar R. and Bukchin Y., "Design and Operation of dynamic assembly lines using work-sharing", *International Journal of Production research,* 44, 18-19, pp. 4043-4065. 2006.

Armbruster D., Gel E. S., "Bucket brigades revisited: Are they always effective," *European Journal of Operational Research,* 172, 1, 213. 2006.

Askin, R. G. Chen, J., "Dynamic task assignment for throughput maximization with worksharing", *European Journal of Operational Research, Balancing Assembly and Transfer Lines,* Vol. 168, Issue 3, 853*,* 2006

Askin, R. G. and Huang Y., "Employee Training and Assignment for Facility Reconfiguration," *Institute of Industrial Engineers 6th Industrial Engineering Research Conference Proceedings*, 426, 1997.

Askin, R. G. and Huang Y., "Forming Effective Worker Teams for Cellular Manufacturing," *International Journal of Production Research*, 39 (11), 2431, 2001.

Bartholdi III, J. J. and Eisenstein D., "A production line that balances itself," *Operations Research,* 44 (1), 21, 1996.

Bartholdi J., Bunimovich L. A. and Eisenstein D, "Dynamics of two- and three-worker bucket brigade production lines ," *Operations Research* 4 (3), 488, 1999.

Bartholdi J., Eisenstein D. and Foley R. D., "Performance of Bucket Brigades when work is stochastic," *Operations Research,* 49(5), 710, 2001.

Baz M., Hunsaker B., Brooks P. and Gosavi A., "Automated Tuning of Optimization Software Parameters (STOP)", University of Pittsburgh Department of Industrial Engineering Technical Report 2007-7, 2007.

Bhaskar, K. and G. Srinivasan, "Static and Dynamic Operator Allocation Problems in Cellular Manufacturing Systems," *International Journal of Production Research*, 35(12), 3467, 1997.

Bratcu A., and Dolgui A., "A survey of the self-balancing production lines ("bucket brigades"), *Journal of Intelligent Manufacturing*, 16 (2), 13, 2005.

Cesani, V.I. and Steudel, H.J., "A study of labor assignment flexibility in cellular manufacturing systems', *Computers and Industrial Engineering* 48, 571, 2005.

Cerny V., "A thermodynamical approach to the traveling salesman problem, an efficient simulation algorithm", *Journal of Optimization Theory and Applications,* 45, 41, 1985

Chen, J, and Askin, R. G., "Throughput maximization in serial production lines with worksharing", *International Journal of Production Economics,* 99, 88, 2006.

Gel E.S., Hopp W.J. and Van Oyen M.P., "Factors affecting opportunity of worksharing as a dynamic line balancing mechanism", *IIE Transactions,* 34 (10), 847, 2002.

Hopp, W. J., Tekin, E. and Van Oyen, M.P., "Benefits of Skill Chaining in Serial Production Lines with Cross-Trained Workers", *Management Science,* 50 (1), 83*,* 2004.

Hopp, W. J. and Van Oyen M. P., "Agile Workforce Evaluation: A Framework for Cross-training and Coordination", *IIE Transactions,* 36, 919, 2004

Jaber M.Y. and Sikstrom S., "A numerical comparison of three potential learning and forgetting models," *Int. J. Production Economics*, 92, 281, 2004.

ILOG CPLEX C++ API 9.0Reference Manual, 2003.

Kirkpatrick S., Gelatt C.D. and Vecchi M.P., "Optimization by Simulated Annealing", *Science,* 220, 671, 1983

Leopairote, K., "Policies for Multi-Skilled Worker Selection, Assignment, and Scheduling", Doctoral Dissertation under the supervision of Associate Professor David A. Nembhard, at Penn State University (The Dissertation was finished at the University of Wisconsin-Madison), 2004.

Lim, Y. F. and Yang, K. K., "Maximizing Throughput of Bucket Brigades on Discrete Work Stations", Lee Kong Chian School of Business, Singapore Management University, *Working Paper.* 2006.

Mazur, J. E. and R. Hastie (1978), "Learning as Accumulation a Reexamination of the Learning Curve," *Psychological Bulletin*, 85(6), 1256, 1978.

McClain, J.O., Schultz, K.L. and Thomas, L. J., "Management of worksharing systems", *Manufacturing & Service Operations Management,* 2 (1), 49-67. 2000.

McClain, J.O., Thomas L. J., and Sox, C., On-the-fly line balancing with very little WIP. *International Journal of Production Economics.* 27, 283-289. 1992.

Molleman, E. and Slomp, J, Functional Flexibility and Team Performance. *International Journal of Production Research*, 37(8), 1837, 1999.

L.F Munoz and J.R. Villalobos. Work allocation strategies for serial assembly lines under high labour turnover. *International Journal of Production Research*, 40(8), 1835, 2002.

Ostolaza J., McClain J. O. and Thomas J. The use of dynamic state-dependent assembly line balancing to improve throughput. *Journal of Manufacturing and Operation Management* 3, 105*,* 1992.

Nembhard, D. A. , "The Effects of Task Complexity and Experience on Learning and Forgetting: A Field Study," *Human Factors*, 42(2), 272, 2000.

Nembhard, D. A. and M. Uzumeri "An Individual-Based Description of Learning Within an Organization," *IEEE Transactions on Engineering Management*, 47(2), 370-378. 2000a.

Nembhard, D. A. and M. Uzumeri "Experiential Learning and Forgetting for Manual and Cognitive Tasks," *International Journal of Industrial Ergonomics*, 25, 315, 2000b.

Nembhard, D. A. and N. Osothsilp, "An Empirical Comparison of Forgetting Models," *IEEE Transaction on Engineering Management*, 48(3), 283, 2001.

Nembhard, D. A. and B. A. Norman, "Worker Efficiency and Responsiveness in Cross-Trained Teams," *Technical Report* 02-02, Department of Industrial Engineering, University of Pittsburgh, 15261, 2002.

Nembhard , D. A and Prichanont K., "Cross-Training in Serial Production with Process Characteristics and Operational Factors", IEEE Transactions on Engineering Management, 54(3), 565, 2007.

Norman, B. A., W. Tharmmaphornphilas, K. L. Needy, B. Bidanda, and R. C. Warner (), "Worker Assignment in Cellular Manufacturing Considering Technical and Human Skills," *International Journal of Production Research*, 40(6), 1479, 2002.

Ostolaza J., McClain J. O. and Thomas J., "The use of dynamic state-dependent assembly line balancing to improve throughput", *Journal of Manufacturing and Operation Management* 3, 105*,* 1990.

Reeves C. R., "Modern Heuristic Techniques for Combinatorial Problems", *JOHN WILEY & SONS, INC.* 1993.

Shafer S. M., Nembhard D. A. and Uzumeri M. U., "The effects of worker learning, forgetting, and heterogeneity on assembly line productivity", *Management Science,* 47, 1639, 2001.

Sennott L. I., Van Oyen M. P. and Iravani S., "Optimal dynamic assignment of a flexible worker on an open production line with specialists," *European Journal of Operational Research*, 170 (2), 541*,*2006.

Slomp J., Bokhorst J. and Molleman E., "Cross-training in a cellular manufacturing environment", *Computers and Industrial Engineering,* 48, 609*,* 2005.

Slomp J. and E. Molleman, "Cross-Training Policies and Performance of Teams," *Group Technology/Cellular Manufacturing World Symposium*, *San Juan, Puerto Rico,* 107-112. 2000.

Schultz K.L., Juran D.C., Boudreau J. W., McClain J. O. and Thomas L. J., "Modeling and worker motivation in JIT production systems", *Management Science,* 44(12), 1595. 1998.

Wisner, J. D. and J. N. Pearson, "An Exploratory Study of the Effects of Operator Relearning in a Dual Resource Constrained Job Shop," *Production and Operations Management*, 2(1), 55. 1993.