2006

Minimizing the makespan in a flexible flowshop with sequence dependent setup times, uniform machines, and limited buffers

Bret Crowder
West Virginia University

Minimizing the Makespan in a Flexible Flowshop with Sequence Dependent Setup Times, Uniform Machines, and Limited Buffers

Bret Crowder

Thesis submitted to the
College of Engineering and Mineral Resources
West Virginia University
In partial fulfillment of the requirements
For the degree of

Master of Science
In
Industrial Engineering

Wafik Iskander, Ph.D., Chair
Bhaskaran Gopalakrishnan, Ph.D.
Alan R. McKendall, Ph.D.

Department of Industrial and Management Systems Engineering

Morgantown, West Virginia
2006

Keywords: Flexible Flowshop, Makespan, Sequence Dependent Setup Times, Uniform Machines, Scheduling, Simulated Annealing, Lower Bound, Limited Buffers

Abstract

Minimizing the Makespan in a Flexible Flowshop with Sequence Dependent Setup Times, Uniform Machines, and Limited Buffers

Bret Crowder

This research addresses the problem of minimizing the makespan in a flexible flowshop with sequence dependent setup times, uniform machines, and limited buffers. A mathematical model was developed to solve this problem. The problem is NP-Hard in the strong sense and only very small problems could be solved optimally. For exact methods, the computation times are long and not practical even when the problems are relatively small. Two construction heuristics were developed that could find solutions quickly. Also a simulated annealing heuristic was constructed that improved the solutions obtained from the construction heuristics. The combined heuristics could compute a good solution in a short amount of time. The heuristics were tested in three different environments: 3 stages, 4 stages, and 5 stages. To assess the quality of the solutions, a lower bound and two simple heuristics were generated for comparison purposes. The proposed heuristics showed steady improvement over the simple heuristics. When compared to the lower bounds, the heuristics performed well for the smaller environment, but the performance quality decreased as the number of stages increased. The combination of these heuristics defiantly shows promise for solving the problem.

Acknowledgments

I would like to thank Dr. Iskander for guidance and support while working on this thesis. His door was always open and he was always willing to help. While in graduate school I've enjoyed taking his classes and working with him in the Industrial Assessment Center. I would like to thank Dr. Gopala for his input on my research and for the responsibilities that he has given me through the Industrial Assessment Center. While working under him I've learned a great deal about energy and about the softer skills of dealing with people that are not taught in the classroom. Even after leaving WVU I will continue to benefit from my experience in the IAC working under these two Professors.

I would like to thank Dr. McKendall for his input on my research and for his guidance while taking his classes in the graduate program. His IENG 350 course sparked my interest in the operations research field and was one of the factors in my decision to come to graduate school.

Last I would like to thank Dr. Plummer for taking me in at the IAC. Working at the IAC helped me finance my degree. While working under Dr. Plummer I learned about focusing on the important issues and how to work with people. Also working with the other students in the IAC helped me to integrate into the graduate program.

Table of Contents:

Abstract ii
List of Tables vi
List of Figures vii
List of Acronyms vii
1 Introduction 1
2 Literature Review 7
2.1 Flowshop Scheduling Overview 7
2.2 Exact Solution Methods 9
2.2.1 Flowshop 9
2.2.2 Flexible Flowshop Exact Methods 11
2.3 Heuristic Methods 12
2.3.1 Flowshop 12
2.3.1.1 Classic Flowshop Methods 12
2.3.1.2 Recent Flowshop Heuristics 13
2.3.1.3 Flowshop with Setup/Buffers 15
2.3.2 Flowshop Heuristics Using Meta-heuristics 17
2.3.3 Flowshop Scheduling Using Combination of Meta-heuristics 20
2.4 Flexible Flowshop Heuristics 21
2.4.1 Two-Stage Flexible Flowshop 22
2.4.2 Three-Stage Flexible Flowshop 25
2.4.3 General Flexible Flowshop 25
2.4.4 Flexible Flowshop Bottleneck Heuristics 27
2.4.5 Flexible Flowshop with Setup Times/Buffers 28
2.4.6 Flexible Flowshop Using Meta-heuristics 30
2.4.7 Flexible Flowshop with limited buffers Using Meta-heuristics 31
2.4.9 Flexible Flowshop with Setup Times Using Meta-heuristics 31
2.5 Simulated Annealing 32
3 Problem Statement and Objectives of Research 34
3.1 Problem Statement 34
3.2 Problem Assumptions 35
3.3 Objectives and Methodology of Research 37
4 Mathematical Model 38
5 Heuristics 43
5.1 RBFFS Construction Heuristic 44
5.2 RBFFS Construction Heuristic Example 46
5.3 RBFFS Simulated Annealing Heuristic 55
5.4 Route Based Simulated Annealing Example 62
5.5 PBFFS Construction Heuristic 62
5.6 PBFFS Construction Heuristic Example 64
6 Lower Bound 71
6.1 Step 1 of the Lower Bound 74
6.2 Step 2 of the Lower Bound 75

6.3 Step 3 of the Lower Bound 79
7 Simple Heuristics 80
7.1 Simple Heuristic One 80
7.2 Simple Heuristic Two 81
8 Experimentation 83
8.1 Problem Instances 83
8.2 Data Generation 84
8.3 Experimentation Overview 86
8.4 Simulated Annealing Parameters 86
8.5 Experimentation Results 87
9 Conclusion 92
9.1 Overview of Research 92
9.2 Contribution to Literature: 94
9.3 Recommendations for Future Research: 94
10 References 96
Appendix A: Experimental Results 107

List of Tables

Table 4-1 Mathematical Model Nomenclature 38
Table 5-1 Construction Heuristic Ex. Process Times Stage 1 46
Table 5-2 Construction Heuristic Ex. Setup Times Stage 1 47
Table 5-3 Construction Heuristic Ex. Process Times Stage 2 47
Table 5-4 Construction Heuristic Ex. Setup Times Stage 2 47
Table 5-5 Construction Heuristic Ex. Process Times Stage 3 47
Table 5-6 Construction Heuristic Ex. Setup Times Stage 3 48
Table 5-7 Step 1 of RBFFS Example. 48
Table 5-8 Step 2 of RBFFS Example. 48
Table 5-9 Priority Routing for RBFFS Example. 49
Table 5-10 RBFFS Makespan for Sequence {3, 2} 50
Table 5-11 RBFFS Makespan for Sequence {2, 3} 50
Table 5-12 RBFFS Makespan for Sequence {3, 2, 4} 51
Table 5-13 RBFFS Makespan for Sequence {3, 4, 2} 51
Table 5-14 RBFFS Makespan for Sequence {4, 3, 2} 52
Table 5-15 RBFFS Makespan for Sequence {3, 4, 2, 1} 52
Table 5-16 RBFFS Makespan for Sequence {3, 4, 1, 2} 53
Table 5-17 RBFFS Makespan for Sequence {3, 1, 4, 2} 53
Table 5-18 RBFFS Makespan for Sequence {1, 3, 4, 2} 54
Table 5-19 Simulated Annealing Nomenclature 56
Table 5-20 Step 1 of PBFFS Example. 65
Table 5-21 Step 2 and 3 of PBFFS Example. 65
Table 5-22 PBFFS Makespan for Priority {3, 2} 66
Table 5-23 PBFFS Makespan for Priority {2, 3} 66
Table 5-24 PBFFS Makespan for Priority {3, 2, 4} 67
Table 5-25 PBFFS Makespan for Priority {3, 4, 2} 67
Table 5-26 PBFFS Makespan for Priority {4, 3, 2} 68
Table 5-27 PBFFS Makespan for Priority {3, 4, 2, 1} 69
Table 5-28 PBFFS Makespan for Priority {3, 4, 1, 2} 69
Table 5-29 PBFFS Makespan for Priority {3, 1, 4, 2} 69
Table 5-30 PBFFS Makespan for Priority {1, 3, 4, 2} 70
Table 6-1Lower Bound Nomenclature 71
Table 6-2 Lower Bound Example Stage 1 Processing Times 73
Table 6-3 Lower Bound Example Setup Times 73
Table 6-4Lower Bound Example Stage 2 Processing Times 73
Table 6-5 Lower Bound Example Stage 3 Processing Times 73
Table 6-6 Lower Bound Example Stage 4 Processing Times 73
Table 6-7 Step 2 Lower Bound 76
Table 6-8 Step 3 Lower Bound 79
Table 7-1 SH1 Example Gantt Chart 81
Table 7-2 Sum of Base Processing Times 81
Table 7-3 SH2 Example Gantt Chart 82
Table 8-1 Experiment Nomenclature 85
Table 8-2 Experimentation Results for the 3-1-2 Environment 89

Table 8-3 Experimentation Results for the 2-1-2-1 Environment 90
Table 8-4 Experimentation Results for the 3-3-2-2-1 Environment 91

List of Figures

Figure 1.1 Complexity Hierarchy of Deterministic Scheduling Environment (Pinedo, 2002) 3
Figure 3.1 Flexible Flowshop with limited Buffers 35
Figure 5.1 Simulated Annealing Process Flow Diagram4 59
Figure 5.1 cont: Simulated Annealing Process Flow Diagram 60
Figure 5.1 cont: Simulated Annealing Process Flow Diagram 61

List of Acronyms

AIS	=	Artificial Immune System
BB	=	Branch and Bound
CDS	=	Flowshop algorithm developed by Campbell, Dudek, and Smith
CDS2	=	Combination of CDS with RA
CMD	=	Cumulative Minimum Deviation
ECTH	=	Earliest Completion Time Heuristic
EFM	=	Earliest Finished Machine
FAMH	=	Fastest Available Machine Heuristic
FCFS	=	First Come First Serve
FFS	=	Flexible Flowshop
FIFO	=	First In First Out
FS	=	Flowshop
GA	=	Genetic Algorithm
GCMD	=	Generalized Cumulative Minimum Deviation Rule developed by Narasimham
Ho	=	Algorithm developed by Ho (1995)
LB	=	Lower Bound
LBJD	=	Least Total Weighted Between-Jobs Delay
LP	=	Linear Programming
LPT	=	Longest Processing Time
LR	=	Lagrangian Relaxation

LSM	=	Latest Start Machine
MD	=	Minimum Deviation
MH	=	Mixed Heuristic
MIP	=	Mixed Integer Program
NEH	=	Flowshop algorithm developed by Nawaz, Enscore, and Ham
NP	=	Non-polynomial
PAM	=	Modified version of Palmer's algorithm
PBFFS	=	Priority Based Construction Heuristic
PBFFS_SA	=	Priority Based Simulated Annealing Heuristic
PBI	=	Progressive Bottleneck Improvement
RA	=	Rapid Access algorithm developed by Dannenbring
RAES	=	Rapid Access with Extensive Search algorithm developed by Dannenbring
RBFFS	=	Route Based Construction Heuristic
RBFFS_SA	=	Route Based Simulated Annealing Heuristic
SA	=	Simulated Annealing
SDST	=	Sequence Dependent Setup Times
SH1	=	First Come First Serve Simple Heuristic
SH2	=	Largest Processing Time Simple Heuristic
SPT	=	Shortest Processing Time
TS	=	Tabu Search
TSP	=	Traveling Salesman Problem

1 Introduction

Scheduling and sequencing is a form of decision making that plays an important role in manufacturing and service industries. Scheduling is the allocation of resources to perform a set of tasks over time and sequencing is the order in which the tasks are performed. Resources can be a number of things such as, machines, workstations in a factory, operators, personal, delivery trucks, office workers, taxis, or runways in an airport. The products that are being processed are usually referred to as jobs and each one of these jobs has a specific set of tasks assigned to it. These problems arise whenever there is a choice to be made on the order in which a number of tasks can be performed and by using which resources. Many scheduling problems are solved by chance, scheduling the tasks in the order they arrive (First in First Out) (Conway et al., 1967). This works in some instances such as purchasing tickets to an event or placing an order at a fast food restaurant. However, in a manufacturing environment this approach can waste valuable time and money. Even when scheduling systems are used, many times they are developed by the operator with objectives that differ from the company's objectives. The operators may only be worried about their workstation and choose jobs based on what will be easiest on them, without concern about other stations. In a flowshop environment this may result in starving stations downstream, or blocking stations upstream. Sometimes these scheduling rules will result in local optima, but when they are all put together the entire system will suffer. In today's global market place with fierce competition, an efficient schedule can be very useful in reducing processing costs and lead times. Scheduling also has an important role in lean manufacturing, which uses scheduling to level production loads. As long as companies and businesses face capacity

constraints, there will be a need for effective scheduling. Scheduling and sequencing involve decision making and by studying these problems we can learn about decision making, and apply it to other areas as well, so it has general practical value (Baker, 1974).

The first notable person to start working on the scheduling problem was Henry Gantt in the early 1900's. Various forms of the Gantt chart that he developed are still used today to represent schedules by scheduling software systems. The first papers on scheduling did not appear until the 1950's and they were authored by scheduling pioneers such as S.M. Johnson, W.E. Smith, and J.R. Jackson (Pinedo, 2002). Since that time a great deal of literature has been devoted to scheduling research. Within scheduling, there are infinite numbers of different problem situations that can be studied. The two main categories are dynamic and static schedules. Dynamic scheduling involves systems that are constantly changing, new jobs can enter or jobs can leave the scheduling environment. In static scheduling systems, the available resources and the number of jobs are known and fixed. Scheduling systems can be further broken down into deterministic and stochastic. In the static, deterministic environment, the number of jobs is fixed and all job data are known, such as processing times. In the static, stochastic environment, the number of jobs is still fixed, but the job data may not be known with certainty. This paper deals with the deterministic static scheduling environment, which will be discussed further. The deterministic static scheduling complexity hierarchy is listed in Figure 1.

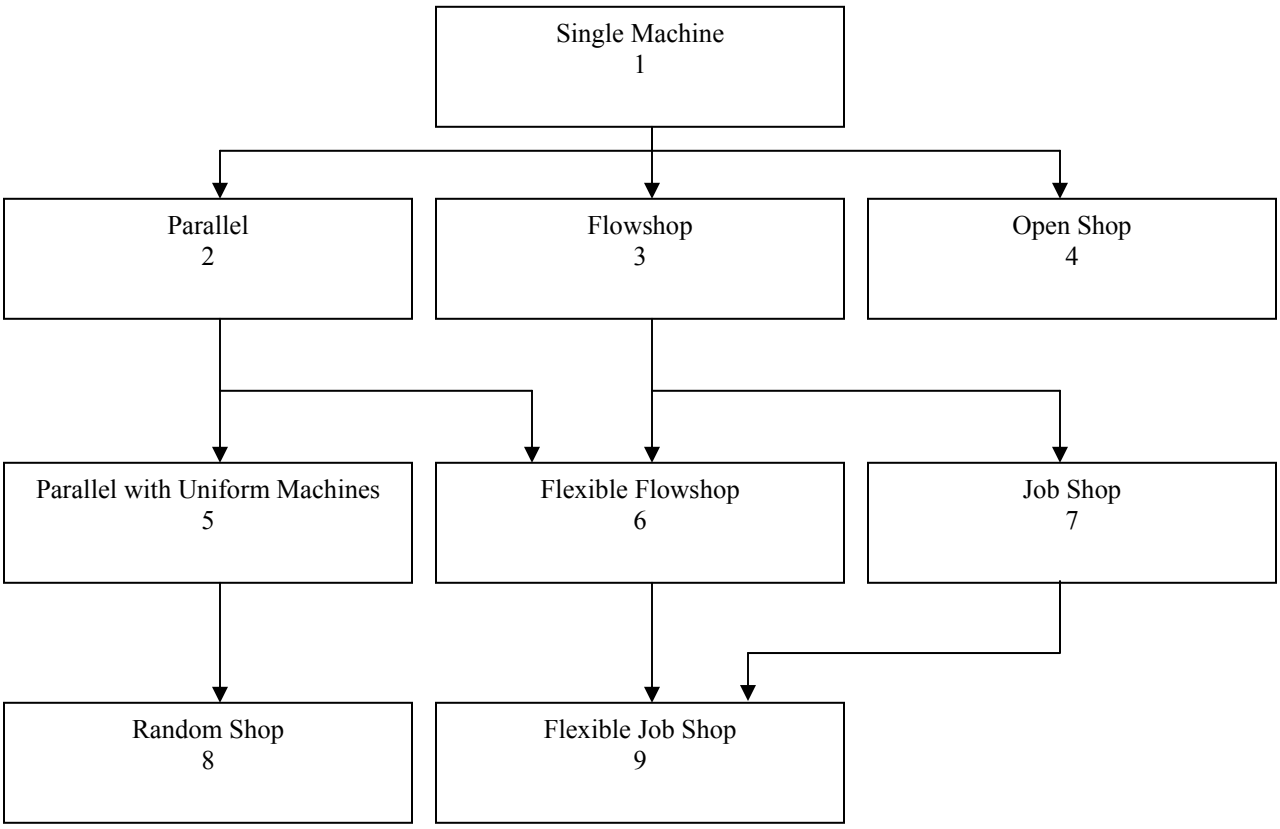


Figure 1.1 Complexity Hierarchy of Deterministic Scheduling Environment (Pinedo, 2002)

(1) The single machine environment is the simplest of all forms, each job *j* must be processed on one machine.

(2) The parallel machine environment involves one stage where there are multiple identical machines, and each job *j* must be processed on one of the machines.

(3) In the flowshop there are *m* machines arranged in a series and each job *j* must be processed on each machine and all jobs must follow the same route. Usually jobs are assumed to move through the system in a first in first out manner.

(4) In the open shop each job *j* must be processed on each of *m* machines, but some of the processing times may be zero. Also there are no restrictions on the routing of the jobs and each job may have a different route.

(5) The parallel machine environment with uniform machines consists of *m* machines in parallel, but with different speeds. The processing time depends on which machine the job is processed on and is calculated by dividing the processing time by the machine speed.

(6) The flexible flowshop is a combination of the flowshop and the parallel environment. There is a number of stages in series with each stage having one or more identical machines in parallel. Each job *j* must be processed on exactly one machine at each stage and the queues may or may not operate on the first in first out or last in first out basis.

(7) A job shop has *m* machines and each job *j* has a predetermined route to follow. In some cases a job may visit a machine more than once.

(8) The random shop is very similar to the parallel machine environment with uniform machines. In the random shop there are *m* machines in parallel with varying speeds, but the machine speeds may be different for each job.

(9) The flexible job shop is a combination of the job shop and parallel machine environments. The flexible job shop consists of work stations with a number of

identical machines in parallel at each station. Each job has its predetermined route and it must visit one machine at each work station on its route.

Flexible flowshops are a common occurrence in many industries. If the processing times at one stage dominate others, then it is common to add another machine. If changes in demand occur, new machines may be purchased over time and have varying speeds. When the machines or resources are costly, the older versions may not be discarded because they still have some value, even if they are not as fast as the newer models.

Most scheduling problems studied in the literature consider setup times to be included in the processing times. This can affect the quality of the solution if setup times differ based on the preceding jobs. Setup times usually consist of preparing tools, setting jigs, setting fixtures, and positioning the job. Since some jobs may be similar in the work they require, it may require smaller time for set up between these jobs and vice versa. When jobs are waiting for setup, no value is added and in some instances the setup time is directly related to cost, so if the setup times are not properly accounted, time and money could be wasted (Allahverdi, 1999).

This research studies a combination of the flowshop and the parallel shop with uniform machines. The problem considers a flexible flowshop with uniform machines, sequence dependent setup times and limited buffers with the objective of minimizing the makespan. This problem is strongly NP hard (Pinedo, 2002), which means that it belongs to a class of problems that are still NP hard even when all numbers in the input are bounded by some polynomial in the length of the input

(National Institute of Standards and Technology <http://www.nist.gov>). As a result finding optimal solutions is not practical for large problems. A construction heuristic will be developed based on the flowshop scheduling algorithm of Nawaz et al. (1983). This algorithm uses a priority list that gives preference to jobs with larger processing times and looks at $[n(n+1)/2] - 1$ solutions instead of all $n!$ solutions; where n equals the number of jobs. This drastically reduces the number of possible solutions considered; for example, for a problem consisting of 5 jobs there are 120 possible solutions, but this algorithm will only look at 14 solutions. After the initial solution is constructed a simulated annealing algorithm will be used to improve the initial solution.

In this research, "job" refers to a product, "machine" refers to a resource that can perform any one of a number of tasks, and "stage" refers to a set of one or more machines in a group that can all perform the same tasks. The jobs need to be processed at every stage in chronological order, and they can be processed on any machine at any given stage. Each stage may have one or more machines and the machines may have differing speeds.

2 Literature Review

2.1 *Flowshop Scheduling Overview*

Since the pioneering work of Johnson (1954), many researchers have studied various forms of the flowshop scheduling problem. In an article by Dudek et al. (1992), a review is done of flowshop scheduling and it is concluded that the flowshop scheduling problem is a mathematically challenging problem. Some have tried exact approaches such as branch and bound or mixed integer programming. These approaches are useful in understanding the problem structure, but are only practical in solving very small problems. The majority of the work on flowshop scheduling has been devoted to heuristics. The heuristics are designed to find optimal or near optimal solutions in a reasonable time period. Taillard (1993) gives data for testing permutation flowshop schedules. Many papers have referenced his article and tested heuristics against the data and lower bounds presented for makespan minimization. Unless otherwise stated, the objective function of each algorithm in this literature review is to minimize the makespan, which is the time period between when the first job starts processing on the first stage to the time when the last job is finished at the final stage. Since the focus of this research is on the static and deterministic scheduling situations, only those will be considered in this review.

The scheduling literature is so vast that any literature search needs to be selective. The theory of sequencing and scheduling is unlimited in problem types. Lawler et al. (1993) presented a survey on deterministic machine scheduling. They discussed complexity and approximation algorithms for problems involving single machine, parallel machines, open shops, flowshops, and job shops. Sequencing and scheduling is

concerned with the optimal allocation of resources to activities over time. A machine is a resource that can perform at most one activity at a time, and activities are considered to be jobs that can be worked on by at most one machine at a time (Lawler et al., 1993). Four of the major approaches to dealing with static sequencing are: (1) combinatorial approaches, which change jobs from one permutation to another, (2) general mathematical programming, (3) heuristics, and (4) Monte Carlo sampling (Day and Hottenstein, 1970).

The flexible flowshop is a case of the flowshop where at each stage there may be one or more machines. When there is more than one machine at a stage, three major issues need to be considered (1) which machines will process which jobs, (2) in which order the machines will process the jobs, and (3) how to decide if the schedule is good or not. There can be many reasons to add machines at a stage such as to increase throughput and reduce the problem of a bottleneck. Also, adding another machine may increase the system reliability and flexibility. If machines are added at different time periods, they may have different speeds resulting in what is termed as uniform machines (Cheng, T., 1990). Linn and Zhang (1999) presented a survey on the flexible flowshop. They stated that most of the research can be grouped into three main categories: 2-stage, 3-stage, and >3-stage. There is not much research reported in the literature on uniform machines which considers setup times. In most of the flowshop scheduling research, the setup times are assumed to be included in the processing times, but in reality that may not be the case. When a machine changes from processing one part to another of a different type, the setup times can be 20%-40% of the processing time. Allahverdi et al. (1999) presented a review of scheduling research involving setup considerations. Setup may be

needed in obtaining tools, positioning wip, returning tool, cleanup, adjusting tools, setting fixtures, and inspection. Two major types of setups are sequence-dependent and sequence-independent. For this research we will be concerned with the first case.

This literature review is not designed to be an in-depth study of the entire field of scheduling and sequencing research. In order to get a better overview, refer to the articles mentioned in this section. This research is mainly focused on the flexible flowshop with buffers, sequence dependent setup times, and uniform machines. Also an overview of simulated annealing is presented in the last section.

2.2 *Exact Solution Methods*

2.2.1 Flowshop

The two machine flowshop can easily be solved by Johnson's algorithm (1954). However, branch and bound and dynamic programming have been used to solve some special cases of the two machine flowshop problem. Corwin et al. (1974) presented a dynamic programming formulation for the two-machine scheduling problem with sequence dependent setup times at one stage. When the sequence dependent setup times are at the second stage and there are less than fourteen jobs, dynamic programming is preferred over branch and bound. Agnetis et al. (1998) used a branch-and-bound algorithm in the two machine flowshop, where the input buffer of the second machine is limited. This algorithm can compute optimal or suboptimal solutions for up to 40 jobs. Other than certain special cases, the flowshop with three or more machines is considered to be NP-Hard (Garey, 1976 and Koulamas, 1998). Ignall and Schrage (1965) and Lee, C. et al. (1993) presented branch and bound techniques for the three machine flowshop

problem. Branch and bound algorithms have also been developed for the multiple machine flowshop, where the number of machines can be greater than three. Carlier et al. (1996) presented two branch and bound algorithms for the m-machine permutation flowshop. Both methods use the depth first strategy. The first method could not solve problems with more than 30 jobs, while the second method could solve up to 50 jobs. Rios-Mercado and Bard (1999A) presented a branch and bound algorithm for the permutation flowshop problem with sequence dependent setup times. The setup times were asymmetric, and lower bounds were calculated by relaxing the problem and reducing it to the two machine case. An upper bound was found by using GRASP, which is a heuristic developed by the authors. The setup times used are usually between 20%-40% of the processing times. Also, the brand and bound procedure permits a partial enumeration search procedure that can calculate approximate solutions.

There are other special cases of the flowshop scheduling problem that have special structures which allow them to be modeled as a traveling salesman problem (TSP). Gupta, J.N.D. (1976) looked at the flowshop scheduling problem with no intermediate storage and uninterrupted flow. The objective was to minimize the weighted sum of idle times on machines. The problem was modeled as a TSP and solved using TSP techniques. Gupta, J.N.D. (1986), also modeled the uninterrupted flowshop with sequence dependent setup times as a TSP.

2.2.2 Flexible Flowshop Exact Methods

Arthanari et al. (1971) presented a branch and bound algorithm to optimally solve the special case of the two-stage flexible flowshop (FFS) where there are multiple machines at the first stage and only one machine at the second. Brah and Hunsucker (1991) also presented a branch and bound algorithm for the general flexible flowshop with multiple stages. This algorithm allows for machine idle time and calculates lower bounds based on jobs, machines, and a composite of both. This enumeration method was originally used for parallel machines, but has been modified to fit the flexible flowshop. It optimally solves problems with 4-8 jobs, 2-5 stages, and 2-3 machines at each stage. Azizoglu et al. (2001) presented a branch and bound algorithm to solve the flexible flowshop where the solutions are not restricted to permutation schedules. They used Brah and Hunsucker's (1991) branching scheme and tested the proposed algorithm on two data sets, one with small processing times and one with large. The algorithm works only for small and medium sized problems. Moursli (1995) proposed a branch and bound approach to solve the multistage flexible flowshop problem with multiple machines at each stage. He also presented three improvements to Brah and Hunsucker's algorithm with three new lower bounds. Portman et al. (1998) presented a branch-and-bound algorithm crossed with genetic algorithms (GA). They also made improvements to Brah and Hunsucker's (1991) lower bound. They used several heuristics to calculate initial upper bounds, and then GA to improve the search value of the upper bound. By adding GA, the proof of the optimal solution is found easier and the optimal solution is found more often. Rajendran et al. (1992) presented a branch and bound algorithm for the parallel machine flowshop where only permutation schedules are considered. Vignier et

al. (1996) presented a branch and bound approach to minimize the total completion time in a k-stage flexible flowshop. The upper bound is found by using the shortest processing time.

Sawik (2002) presented a mixed integer programming approach for the flexible flowshop. There were limited buffers viewed as machines with zero processing times. Also in Sawik (2000), two mixed integer programs were presented for the FFS where one case has buffers and the other does not. Aghezzaf et al. (1998) presented a mixed integer linear programming model for the flexible flowshop with product and machine dependent setup times, but it is feasible only for small problems.

2.3 Heuristic Methods

2.3.1 Flowshop

2.3.1.1 Classic Flowshop Methods

Since the pioneer work of Johnson in 1954 to minimize makespan in flowshop sequencing, many have tried to develop algorithms that can solve the multiple machine flowshop with the simplicity that his algorithm solved the two machine case. There are four main classic heuristics that have been studied, referenced, used in comparison, and used in constructing new algorithms by many researchers. All of the classic heuristics use permutation schedules. The first is Palmer (1965) who presented a heuristic that gives priority to jobs whose processing time changes from short to long. The slope order produces schedules fairly close to optimal. The second classic algorithm CDS was developed by Campbell, Dudek, and Smith (1970). This procedure can be applied by hand and generates (m-1) schedule sequences, where *m* is the total number of machines.

It outperforms Palmer's, which produces only one sequence. The algorithm can solve for 3-60 jobs and 3-30 machines, or even larger problems. The third classic algorithm was given by Dannenbring (1977). In his paper, he compares eleven flowshop heuristics, three of which were previously unreported. Heuristics from previous literature included Palmer and CDS, but the new heuristic he developed and called Rapid Access with Extensive Search (RAES) performed the best. Problems were solved with up to 50 jobs and 50 machines. In the larger problems, RAES found the best solution in 71.25% of the problems, while the next best rule found the best solution in 18.13% of the problems. The algorithm also has a neighborhood search strategy that changes neighbors until there is no improvement. Even though Dannenbring's algorithm worked well, the best of the classic algorithms is the fourth, which was presented by Nawaz, Enscore, and Ham (1983) (NEH). Taillard (1990) and others came to the same conclusion when comparing the classic heuristics. This algorithm is still considered to be one of the best methods of constructing initial solution (Koulamas, 1998). Numerous researchers still use NEH in their construction algorithms before applying an improvement heuristic. NEH is superior to CDS and all other algorithms developed up to 1983. It uses the assumption that jobs with more processing times should get a higher priority. The only way CDS would outperform NEH is if the number of machines greatly outnumbers the number of jobs (Turner and Booth, 1987).

2.3.1.2 Recent Flowshop Heuristics

Hundal et al. (1988) extended Palmer's algorithm and allowed it to generate three sequences instead of one. They combined CDS with Palmer and added a pairwise

exchange search strategy. Using these extensions improves the solution, but also adds computation time. Koulamas (1998) presented a construction heuristic for the flowshop scheduling problem. In this heuristic, non-permutation as well as permutation schedules are considered. Koulamas's heuristic performed as well as NEH when the optimal schedule is a permutation and better when the optimal schedule is not a permutation. Ho et al. (1991) developed a heuristic designed to reduce gaps between operations in solutions generated by other heuristics. The algorithm was compared to CDS, Palmer, Gupta, Dannenbring, Hundal and Rajgopal's (1988). For minimizing makespan, Ho, CDS, and Hundal and Rajgopal's worked the best. Sarin et al. (1993) developed a heuristic, that attempts to minimize the idle time on the last machine. Only permutation schedules were considered. When compared with NEH (1983), NEH worked better for small problems, but Sarin's heuristic dominated for larger problems.

Chen et al. (1996) presented a heuristic for the three machine flowshop. The heuristic is based on Johnson's algorithm and has a worst case performance ratio that is better than 2. Lee, C. et al. (1993) also presented heuristics and their error bounds for the three machine flowshop problem. Barman (1998) looked at combining 3 different priority rules for the three-stage flowshop where there are two machines at each stage. The objectives were to minimize mean lateness, mean tardiness, maximum tardiness, and percent tardy. The results showed that the combination of rules work better than using any one rule by itself.

Carlier (1982) presented a unique algorithm that gives the bottleneck stage priority over all others. The jobs are scheduled in a way that minimizes the time spent at the bottleneck stage, also the algorithm considers release and processing times. Egin et

al. (2004) proposed an artificial immune system (AIS) that is similar to neural networks. Wang et al. (1997) presented two heuristics, the first heuristic reduces machine idle time, and the second reduces machine idle time and job queue times.

2.3.1.3 Flowshop with Setup/Buffers

Other special cases of flowshop scheduling are those having setup times, limited or no buffers, or uniform machines. In general flowshop scheduling, the processing times include all setup times, the buffers are considered to be unlimited, and all machines are identical (i.e. have the same speed). Sule (1982) looked at scheduling jobs on a two machine flowshop. The processing time is separated into setup, processing, and removal times. Also it further categorizes operations into inside and outside processing. Gupta, J.N.D. et al. (1987) presented a heuristic for flowshop scheduling with sequence-dependent additive setup times. The approximate solution can yield optimal schedules for the two machine case. The heuristic produced better solutions than Sule's algorithm.

Simmons (1992) presented four heuristics for the flowshop sequencing problem with sequence dependent setup times. The results showed that sophistication does not necessarily lead to better performance. One of the heuristics (TOTAL) uses the sum of processing and setup times, where another (SETUP) just works with the sum of setup times. TOTAL performed the best. Rios-Mercado and Bard (1998) presented two heuristics for the flowshop scheduling problem with sequence dependent setup times, where only permutation schedules are considered. The first heuristic was an extension of the NEH flowshop heuristic, and the second was a greedy randomized adaptive search procedure (GRASP). GRASP has a construction and improvement phase. Both

heuristics were compared with Simmons's (1992) (SETUP) heuristic using two data sets. For the first data set, a traveling salesman heuristic (TSP) worked best for a small number of machines, but (GRASP) worked better for a large number of machines. For the second data set SETUP worked the best. Rios-Mercado and Bard (1999B) presented another heuristic for solving the permutation flowshop problem with setup times. The procedure transformed the problem into a Traveling Salesman Problem (TSP). Large setup times and makespan are factors that penalize the objective function. The setup times used were asymmetric and equal to 20%-40% of the processing time. The heuristic worked better when the number of machines was small.

Allahverdi (2000) looked at minimizing the mean flow time in the two-machine flowshop with sequence independent setup times. Three heuristics were given with an overall average error of 0.7% of the optimal. Also, optimal solutions were found for two special cases. Experiments were performed with job sets of 10, 15, 20, 25, 30, and 35. The flowshop with sequence dependent setup times and limited buffers was studied by Gupta, J.N.D. (1986), but no experiments were performed.

King and Spachis (1980) looked at existing and new heuristics to solve the flowshop problem. The new heuristics studied the permutation, no-wait flowshop. The no-wait flowshop is reducible to the asymmetric traveling salesman problem. For no passing, the least total weighted between-jobs delay (LBJD) heuristic performed well. For the no-wait problem, the minimum covering level, the maximum left shift savings, and LBJD heuristics worked well. Details of these heuristics can be found in King and Spachis (1980).

2.3.2 Flowshop Heuristics Using Meta-heuristics

Most of the recent work in flowshop scheduling has been devoted to the combinatorial approach and the use of meta-heuristics. The three traditional meta-heuristics used in the literature are Tabu Search (TS), Genetic Algorithms (GA), and Simulated Annealing (SA). SA and TS have both been found to work well for the permutation flowshop problem. In this research, SA will be used, so the literature covered will focus mainly on SA with some brief mention of the other methods. Also, it should be noted that a new heuristic, called Ant Systems has recently been applied to the permutation flowshop problem with comparable results to those of SA and TS.

Osman and Potts (1989) presented a simulated annealing (SA) heuristic for the permutation flowshop scheduling problem. To search the neighborhood, a non-adjacent interchange and a forward/backward shift system are used. Four different SA algorithms were used and they perform better than NEH, but require more time. Ogbu et al. (1990B) presented a different simulated annealing heuristic for the flowshop scheduling problem. This heuristic used an acceptance probability that is the same for all iterations. The paper states the importance of reducing the temperature slowly. Also the heuristic used a last improvement technique that is different from Osman and Potts (1989) which used first improvement. For this heuristic, the insertion/shift perturbation scheme worked better than the interchange/pairwise. The paper also stated that for the n-job m-machine flowshop, SA outperforms all other heuristics. Ogbu and Smith (1990A) also used simulated annealing to solve the flowshop scheduling problem. The probability of acceptance of an inferior solution is independent of the change in function value. Two perturbation schemes are used; one uses pairwise exchange and the other uses job

insertion. Insertion seemed to work better in the larger problems. In theory, the choice of a starting solution will not matter, but practical experience showed that a good starting solution leads to more efficient convergence. The initial starting solutions of Palmer and Dannenbring were used here. Two sets of problems were considered, the small set had 7 jobs and 5 machines, and the large has 15 jobs and 10 machines. The small problems were solved by branch and bound for comparison, and for the large problems the solutions were compared by using the best solution found. The proposed heuristic found better solutions in a set time period than traditional approaches. Kouvelis et al. (1992) also developed a simulated annealing procedure. The initial solution is randomly generated, but the article states that initial solutions that exploit specific problem structure will get better final solutions. Ishibuchi et al. (1995) developed two simulated annealing algorithms with modifications to the generation mechanism. The generation mechanism is designed so the choice of cooling schedule will not affect the quality of the solution; it works with a pool of possible solutions. They found that Tabu search is slightly inferior to their algorithm. The proposed algorithm was also compared to the SA algorithm presented by Osman and Potts (1989), and it was found that the proposed is better for the 20 job problem and Osman and Potts algorithm was better for the 50 job problem. Zegordi et al. (1995) presented a heuristic that combines simulated annealing with problem specific knowledge. Johnson's rule is used to develop a move desirability index. The proposed algorithm was compared with NEH, CDS, and Osman and Potts. Osman and Potts performed better on larger problems, but the proposed algorithm was better on smaller problems. Low (2004) developed a simulated annealing algorithm that uses a modified NEH algorithm as the initial starting solution. Three different

neighborhood search schemes are used: adjacent pairwise exchange, general pairwise exchange, and insertion. General pairwise exchange and insertion can yield good solutions, but they are computationally costly. Adjacent pairwise exchange is considered to be not as computationally costly. The algorithm was tested using Taillard's (1993) data and it was found that the proposed SA with an initial starting solution of SPT/FCFS worked the best. Tian et al. (1999) used simulated annealing to solve combinatorial optimization problems with permutation properties. The problems studied are traveling salesperson, flowshop, and quadratic assignment. Six different perturbation schemes were used for each problem and the results were presented. For the symmetric traveling salesperson, block reversal worked best. For the flowshop, two job random exchange and block insertion were the best performers.

Widmer and Hertz (1989) presented a two phase heuristic where the first phase used a construction heuristic based on an analogy of the problem with the traveling salesperson problem. In the second phase, the initial solution is improved using tabu search. Their algorithm was compared with NEH, CDS, and Dannenbring's RA, and the proposed algorithm produced the best sequence 80% of the time. Taillard (1990) presented a tabu search heuristic that was compared to NEH and shown to be superior. Ying et al. (2004) presented an ant systems approach to the permutation flowshop problem. The test data of Taillard (1993) were used. Ant systems can get good solutions with reasonable computation time. The results showed that the proposed ant system heuristic is on average minimally better than SA.

Weng (2000) used a modified NEH algorithm to solve the flowshop scheduling problem with limited buffers. NEH is used to generate the initial permutation schedule,

and then tabu search (TS) is used to improve the solution. The objective function is to minimize the mean job flow time. TS was shown to improve the solution, and buffers greater than four are not needed. As the numbers of machines or buffers increase, the TS improvement decreases.

Parathasarathy et al. (1997) used simulated annealing to solve the flowshop scheduling problem with sequence dependent setup times and the objective of minimizing the mean weighted tardiness. A case study was carried out in a drill bit manufacturing industry. The simulated annealing heuristic was compared to tabu search and out-performed it 70% of the time. Also the proposed algorithm solved the 95 job problem in 3 hours while tabu search took 12 hours. Norman (1999) presented a tabu search heuristic for the flowshop problem with limited buffers and sequence dependent setup times. NEH was used as the construction algorithm. Also, another heuristic was developed using greedy improvement, but tabu search works much better. Good lower bounds are difficult to establish because of the buffers and the asymmetric setup times.

2.3.3 Flowshop Scheduling Using Combination of Meta-heuristics

A new method of solving the flowshop problem is to combine two meta-heuristics; most of this has been done by combining SA and GA. By combining these two heuristics, good solutions can be obtained. Wang et al. (2003B) presented a hybrid heuristic for the flowshop scheduling problem. The algorithm used NEH, GA, and SA. The heuristic outperformed NEH and was comparable to TS, SA, and GA. Allahverdi et al. (2004) presented a hybrid simulated annealing and genetic algorithms heuristic to solve the no-wait flowshop. The objective was to minimize the weighted sum of

makespan and maximum lateness. Nearcchou (2004A) used a hybrid simulated annealing algorithm to solve the flowshop problem. The heuristic has the basic structure of SA with some aspects borrowed from genetic algorithms and local search techniques. A random exchange perturbation scheme is used. This heuristic worked with a population of possible solutions at each iteration. The solutions obtained are comparable to the best available in the literature. Nearcchou (2004B) presented a hybrid simulated annealing algorithm to solve the permutation flowshop problem. The simulated annealing algorithm is crossed with genetic algorithms. Initial solutions are randomly generated and a shift perturbation scheme is used. The proposed algorithm can find makespans faster than other known meta-heuristics.

2.4 Flexible Flowshop Heuristics

Finding the minimum makespan in a flexible flowshop problem is considered to be NP-Hard (Hoogeveen et al., 1996). Therefore most of the research in FFS scheduling has revolved around heuristic methods. Since heuristics are approximation methods and are not likely to find the optimal solution, a method of testing their performance is needed. There are three main methods to test a heuristic, test it against a lower bound, against other heuristics for the same problem, or against the best solution found. Santos et al. (1995) presented global lower bounds for makespan minimization on the flexible flowshop scheduling problem. The purpose was to give a bench-mark to test the quality of heuristics. The optimal makespan was predicted 38% of the time and the average relative error was 4.6%. Also in 85% of the 653 problems tested, the bounds were within 10% of the optimal solution.

2.4.1 Two-Stage Flexible Flowshop

Gupta, J.N.D. (1988) showed that the two-stage flowshop where there are multiple identical machines at each stage is NP-Complete. Gupta, J.N.D. et al. (1997) presented a heuristic to solve the two-stage flexible flowshop problem with multiple machines at the first stage and one machine at the second stage. The heuristic uses Johnson's rule to create a list of jobs, then this list is used in one of five heuristics. Also a steepest descent strategy is used to find local optima from the heuristic's solution. Lee et al. (1998) presented two heuristics for the same problem, one is a look ahead and the other a look behind. Also, a dynamic programming algorithm was developed for the two stage problem with two machines at each stage. Oi (1996) developed four heuristics for the same problem. He assumed that the manufacturing is done at the first stage and assembly is done at the second stage. The objective was to minimize the sum of weighted customer lead times. Only permutation schedules were considered. The four heuristics are Lagrangian relaxation, greedy search, total weighted shortest processing time, and one that is simply tardiness based. Chen (1995) addressed the two-stage problem with parallel machines at one stage and a single machine at the other. Oguz et al. (1997) looked at the two-stage flexible flowshop problem with different machines at the first stage and one common machine at the second stage. He developed a heuristic based on Johnson's rule which was found to be effective.

Narasimham et al. (1984) looked at scheduling rules in a two-stage flexible flowshop with one machine at the first stage and two machines at the second. The objective was to minimize the sum of machine idleness and in-process waiting time at the second stage. Cumulative minimum deviation worked the best, minimum deviation

(MD) second, and SPT and LPT were last. Gupta, J.N.D. et al. (1991) looked at scheduling jobs in two-stage flexible flowshop with one machine at the first stage and multiple machines at the second. They developed an algorithm that can be used to find approximate solutions or to increase the efficiency of a branch and bound algorithm. They had two objectives; the first was to minimize makespan and the second to minimize the number of machines. This problem was also considered by Tsubone et al. (1996). In Tsubone et al., the first stage used the shortest processing time, longest processing time, RATIO, and a Modified Johnson's rule. The second stage simply used first come first serve. The RATIO rule worked best for makespan minimization. Li (1997) presented a backward and a forward scheduling heuristic for the same problem. The heuristics performed better than shortest processing time (SPT) and longest processing time (LPT). A Pratt and Whitney Blade production line was the basis for this scheduling problem. The backward algorithm performed better than the forward. Huang (1998) studied a similar problem, but the first stage had one machine and the second stage had multiple uniform machines. In his paper, the parts are grouped into families with major and minor setup times that are independent of the machine speeds. The problem was examined by using two heuristics and eight sequencing rules. This scheduling problem also came from a Pratt and Whitney Blade manufacturing facility. Average results were between 20% and 60% above the lower bounds. Riane et al. (1998) developed two heuristics for the two stage hybrid flowshop with one machine in the first stage and two dedicated machines in the second stage. The first heuristic was based on dynamic programming, while the second used a construction heuristic combined with a greedy algorithm. The dynamic programming approach worked best. For the same problem, Riane et al. (2002)

proposed two heuristics that were found to be efficient. A dynamic programming approach was also used for problems with less than 15 jobs.

Lee et al. (1994) presented a heuristic for solving the general two-stage flexible flowshop. Five lower bounds were used to test the algorithm and an additional heuristic was developed for the case with more than 2 stages. As the number of machines increases, the problem becomes harder. Guinet et al. (1996) looked at scheduling of two-stage flexible flowshops with unlimited storage. A MIP formulation was presented and lower bounds were calculated. A sequence first and allocation second heuristic was developed that uses Johnson's rule and different combinations of seven other rules developed by other authors to create a priority list, then the jobs are scheduled using the priority list. Lin et al. (2003) studied the two-stage flexible flowshop with sequence dependent setup times at stage one and dedicated machines at the second stage. This problem came from a label sticker manufacturing company. A tabu search algorithm was used in the first stage and a first in first out (FIFO) algorithm was used in the second stage. The objective was to minimize the maximum weighted tardiness. The heuristic performed well and was to be implemented at the company for which it was designed. Narasimham et al. (1987) tested three scheduling rules for the two-stage flexible flowshop. The rules tested were shortest processing time (SPT), longest processing time (LPT), and minimum deviation. The problem was modeled after a scheduling problem in the Schlitz Brewing Company. The objective function has multiple criteria such as minimizing total cost of order waiting, machine idle time, makespan, and average completion time. Generalized cumulative minimum deviation rule (GCMD) worked the best. Soewandi et al. (2003) developed three heuristics for scheduling in the two-stage

flexible flowshop with uniform machines at each stage. Some of the rules used are the earliest finished machine (EFM) and latest start machine (LSM). Uetake et al. (1995) looked at the two-stage FFS with one machine at the first stage and multiple different machines at the second stage. The objective was to minimize makespan and maximum work in process. This type of system occurs in steel, chemical, and paper industries. The first stage used SPT, LPT, RATIO, Johnson's rule, and the second stage used the first come first serve rule (FCFS). Verma et al. (1999) showed that the two-stage FFS with parallel and uniform machines is NP-Hard. Three different heuristics were presented; earliest completion time (ECTH), fastest available machine heuristic (FAMH), and mixed heuristic (MH). The mixed heuristic is just a combination of the others and tends to perform the best.

2.4.2 Three-Stage Flexible Flowshop

Koulamas et al. (2000) presented a linear time algorithm for two-stage and three-stage flexible flowshops. The worst case bounds are no worse than those currently available. Soewandi et al. (2001) presented two heuristics for the three-stage problem. Some rules that were used in the formation of the heuristics are Johnson's, first available machine, last busy machine, and modified Johnson's rule.

2.4.3 General Flexible Flowshop

Wittrock (1985) looked at scheduling in flexible flow lines. The objective was to maximize throughput and reduce work in process. He considered two scheduling decisions, one is when should a job enter the system, and the other is what should the

daily product mix be. The buffer works on a first come first serve basis and the heuristic tries to balance workloads. Wittrock (1988) also looked at an adaptable scheduling algorithm to minimize makespan and queueing in the flexible flowshop. The heuristic breaks the problem into three subproblems; machine allocation, sequencing, and timing. No queues are allowed in front of an idle machine, even though sometimes it may improve the solution. Kochhar et al. (1988) studied entry point scheduling. This technique uses local perturbation to obtain schedules within a few percent of optimal. A construction heuristic and random starting solutions were used. They state in their paper that a good starting solution is one that will lead the local search into different areas. The local search techniques used are pairwise exchange and block exchange, both with steepest descent and multiple starting solutions. Ding et al. (1994) presented three heuristics for the flexible flowshop with 1 to 3 machines at each stage. Two of the heuristics are based on Johnson's and CDS algorithms, while the third is based on Gupta (1972). The algorithms based on Johnson's algorithm worked the best. Guinet and Solomon (1996) looked at scheduling jobs in a flexible flowshop with the objective of minimizing the makespan or maximum tardiness. The heuristics used are based on CDS, NEH, and Townsend (1977). NEH worked the best for makespan minimization. Leon et al. (1997) presented a heuristic that was previously used for job shop scheduling, but adapted it for the flexible flowshop. The heuristic uses the shortest processing time rule combined with local search techniques. The local search techniques used are single neighborhood, steepest descent, and first improvement. Brah and Lou (1999) compared five heuristics for the FFS with the objective of minimizing makespan and mean flowtime. The heuristics tested are NEH, HO, CDS, PAM, and CDS2. NEH performed

the best for makespan. The heuristics were compared using the global lower bounds presented by Santos (1995). Chang (1994) developed an algorithm based on Lagrangian relaxation and minimum cost linear network flow. A rescheduling algorithm is also given. The objectives are to avoid overdue penalty, reduce cost of WIP, and reduce cost of overtime. Jayamohan et al. (2000) looked at minimization of flow time and tardiness of jobs in a flexible flowshop. Two approaches were investigated; one uses the same dispatching rule at all stages while the other uses different rules at different stages. This study found that using one good rule at all stages works best, which contradicts a previous study done by Barman (1998), but his study was on the three stage flowshop. Also, it should be noted that using different rules at different stages adds a great amount of computation time.

2.4.4 Flexible Flowshop Bottleneck Heuristics

A few researchers took a different approach to scheduling in the FFS by putting emphasis on the sequence at the bottleneck stage. Adams et al. (1988) presented a shifting bottleneck heuristic (SBH) that is designed for the job shop scheduling problem, but can be applied to the flexible flowshop. It works by scheduling jobs on the bottleneck machine and then re-sequencing the other stages. Their algorithm can solve a ten job, ten machine problem in less than six minutes. Cheng et al. (2001) used the shifting bottleneck heuristic for the FFS in order to minimize the maximum lateness. The heuristic works in two phases, the first schedules *m* stages one by one successively in descending order of lower bounds, and the second phase re-optimizes each stage. The results showed that optimal or near optimal solutions are found in short time periods.

Phadnis et al. (2003) presented a progressive bottleneck improvement (PBI) procedure to solve the FFS. Only one job is sequenced at a time and it is sequenced at all stages. The bottle neck stage is identified before each job is scheduled. The heuristic is similar to Adams (1988) shifting bottleneck heuristic. Phadnis's heuristic is simple and produced good solutions. Acero-Domingvez et al. (2004) developed a heuristic based on the theory of constraints by optimizing the bottleneck stage. By optimizing the bottleneck stage the overall system is improved; if other areas are improved they will only achieve local improvement and the entire system may not benefit. The heuristic requires three steps: 1) bottleneck identification, 2) scheduling of jobs at bottleneck stage, and 3) schedule non-bottleneck stages. The algorithm was compared with the Shifting Bottleneck Heuristic and performed comparably. Lee et al. (2004) used a bottleneck focused algorithm in order to minimize total tardiness in a flexible flowshop. The problem in question comes from the printed circuit board industry. The assumptions made include unlimited buffers and that the machines never fail. The algorithm first schedules the bottleneck stage and then schedules the other stages.

2.4.5 Flexible Flowshop with Setup Times/Buffers

Another special case of the flexible flowshop sequencing problem is that of having setup times, limited buffers, or both. Kochhar and Morris (1987) presented heuristic methods to solve the flexible flowshop sequencing problem with limited buffers and setup times. The problem was broken into two parts, entry point scheduling and dispatching. The problem was further broken down into three parts; initial sequence, how to manage work in process, and if multiple jobs are available, which ones are

chosen. The steepest descent neighborhood search technique was used with the perturbation schemes of pairwise and block exchange. A realistic problem of 16 machines was solved in 30 minutes while smaller problems took seconds. Twelve cases were solved and the average performance was 0.5% from optimal, with the worst case performance about 2% from optimal. Sawik (1987) developed three heuristics for the flexible flowshop with limited buffers. Beside makespan, the objective function included minimization of work in process holding costs. In two of the heuristics, schedules are period by period and fixed, while the third is capable of making adjustments to the schedule when a buffer becomes full or a shortage to a machine occurs. Sawik (1995) also presented a single pass part-by-part heuristic for the flexible flowshop with no buffers. The schedule is determined only once and run time is short. Khmelnitsky et al. (1997) developed a numerical method and two heuristics to solve the flexible flowshop with partial sequence dependent setup times. Aghezzaf et al. (1998) presented a LP based heuristic for the FFS where setup times are product and machine dependent, the heuristic is based on the duality of linear programming. This algorithm requires a large amount of computation time; at each iteration a trans-shipment problem must be solved. Liu et al. (2000) presented a Lagrangian relaxation based approach for the flexible flowshop with sequence dependent setup times. The objectives were to meet due dates, reduce work in process, and reduce setup. In testing 16 problems, the average difference between the proposed solution and the true optimum was 15%. Wang et al. (2003A) looked at using a neural network system to solve the flexible flowshop problem. The learning strategy borrows from simulated annealing and lagrangian relaxation. The lower

bounds used for comparison were obtained from Santos (1995). Neural networks did not provide good solutions in short times when compared to other heuristics.

2.4.6 Flexible Flowshop Using Meta-heuristics

This section discusses some of the meta-heuristics that have been applied to the flexible flowshop, with the main focus on simulated annealing. Haouari et al. (1997) used two two-phased heuristics based on simulated annealing and tabu search to solve the two stage flexible flowshop with parallel machines at each stage. To construct an initial solution, the most work remaining rule was used to create a priority list. The objective was to minimize makespan. Tabu search performed just a little better than simulated annealing. Riane et al. (1999) presented a simulated annealing algorithm to solve the flexible flowshop problem where only permutation sequences are considered. The paper emphasizes the importance of temperature reduction; spending too much time at a high temperature wastes time and decreasing the temperature too fast will limit the search to local optima. Negenman (2001) tested a variable depth search method combined with three simulated annealing heuristics and three tabu search heuristics to solve the flexible flowshop. The combination of variable depth search with tabu search performed the best. Loukil et al. (2005) used simulated annealing for multiple-objective scheduling for the one-machine, parallel machine, and permutation flowshop problems. Their neighborhood search schemes consisted of random job exchange and job insertion. A random choice of these two was made.

2.4.7 Flexible Flowshop with limited buffers Using Meta-heuristics

Shieh (2003) studied the flexible flowshop with limited buffers. The objective was to minimize the makespan and maximize the machine free time. An exact algorithm was developed using mixed integer programming, and was able to solve only very small problems. It works in two phases; the first minimizes the makespan and the second maximizes the machine free time. A construction algorithm was developed called (MGPAFFS). Then a simulated annealing improvement heuristic was used to improve the construction algorithms solution. Two sets of data were used; one with 30 jobs and the other with 60 jobs. The results show that after the application of SA, the solutions for the small and larger sets were 1.6%-28.3% and 2.2%-32.7% respectively from the lower bounds. Also SA improved the construction heuristics solution by 1.2%-4.9% and 0.9%-8.9% for the small and large data sets respectively.

2.4.9 Flexible Flowshop with Setup Times Using Meta-heuristics

Low (2005) presented three simulated annealing algorithms for the multistage flowshop with unrelated parallel machines. This problem has independent setup and dependent removal times. A well designed solution generator was used to find the initial solution. Sethanan (2001) looked at the problem of scheduling a flexible flowshop with sequence dependent setup times and uniform machines. A mathematical model was developed to solve small problems and two heuristics were presented to solve larger problems. The first heuristic is a construction heuristic called “flexible flowshop with sequence dependent setup times heuristic” (FFSDSTH). The second heuristic is a tabu search and was used to further improve the solution obtained from the construction

heuristic. Two lower bounds are calculated; one uses a forward system and the other a backward. The machine waiting time, idle time, and total setup and processing times on the last machine were used to calculate the lower bounds. The results show that FFSDSTH is efficient at finding solutions and that TS improves the solution between 2.95% and 11.85%.

2.5 Simulated Annealing

Metropolis et al. (1953) were the first to look at the properties of annealing in order to solve mathematical problems. Simulated annealing is derived from an analogy between the physical annealing of solids and combinatorial optimization. Physically, it refers to the heating of a substance to the melting point, then lowering the temperature slowly and spending a long time at temperatures close to the freezing point. The ground state of the solid will have a certain structure and the cooling will not obtain a good structure if it is not done properly (Eglese, 1990). Kirkpatirick et al. (1983) were the first to use simulated annealing to solve combinatorial problems. They listed some of the positive aspects of simulated annealing such as: (1) significant improvements over random starting solutions are possible, (2) there are many near optimal solutions, so a stochastic search procedure such as SA should find some, (3) no one of the ground state solutions is any better than the others, so it is not worth it to search for the absolute optimal (Kirkpatirick et al., 1983). One thing that makes simulated annealing and other metaheuristcs different from simple local search procedures is that the algorithm avoids being trapped in a local optimum by sometimes accepting a neighborhood move that increases the value of the solution (Eglese, 1990). Many researchers have shown that

simulated annealing can find good solutions for scheduling permutation flowshop problems (e.g. Allahverdi et al., 2004, Haouari et al., 1997, Ishibuchi et al., 1995, Kouvelis et al., 1992, Loukil et al., 2005, Low, 2005, Low et al., 2004, Nearchou, 2004A, Negenman, 2001, Ogbu et al., 1990A/B, Osman et al., 1989, Parthasarathy et al., 1997, Riane et al., 1999, Shieh, 2003, Tian et al., 1999, and Zegordi et al., 1995). Simulated Annealing will be used in this study in an effort to improve the initial solution obtained using two construction algorithms.

3 Problem Statement and Objectives of Research

3.1 Problem Statement

This research addresses the flexible flowshop with sequence dependent setup times, uniform machines, and limited buffers. There are one or more machines at each stage, and more than two stages. The machines at each stage have different processing speeds and the first machine is considered to be the fastest. Each job has a base processing time that is the time to be processed on the fastest machine. The processing times for the other jobs are found by dividing the base processing times by the machine speed. There are unlimited buffers before the first stage and after the last stage. For the stages in between, the buffers are assumed to have limited capacity. If a buffer becomes full, it will block machines at the previous stage until space becomes available in the buffer. The environment is deterministic; that is the number of jobs and all of their data are known in advance and fixed. The number of stages and machines at each stage are known in advance and fixed. Jobs are considered to be tasks and must be processed on machines, in each stage, in technological order. Machines are considered to be resources that perform tasks. The objective is to minimize the makespan, which is the completion time of the last job at the last stage. The flexible flowshop with limited buffers is shown in Figure 3.1.

The standard flowshop with more than two machines with the objective of minimizing makespan is considered to be NP-Hard in the strong sense (Pinedo 2002). This problem is considerably more complex because it adds uniform parallel machines at each stage, limited buffers, and sequence dependent setup times. Some research has been done on the two stage flexible flowshops, but for greater than two stages the research is

scarce. No research could be found in the literature that attempts to solve the flexible flowshop with sequence depended setup times, uniform machines, and limited buffers.

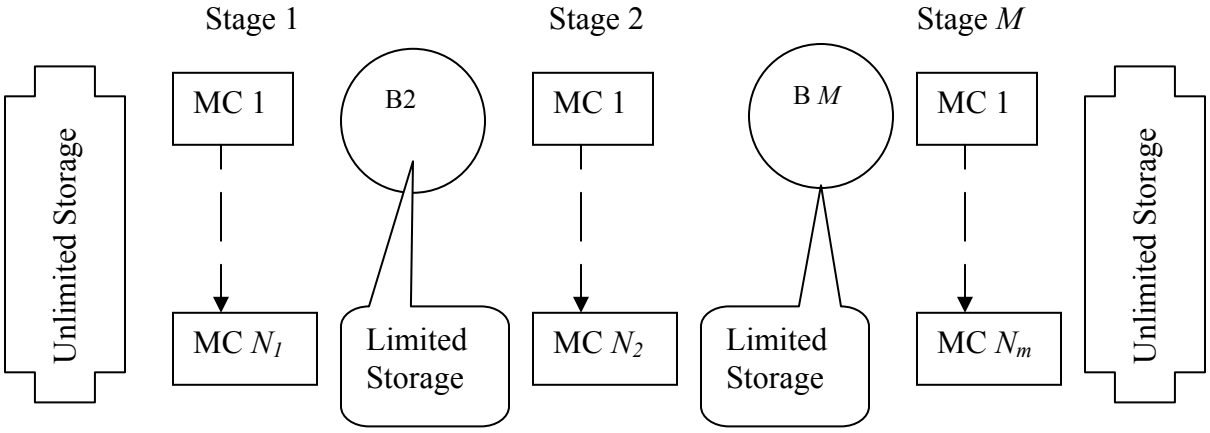


Figure 3.1 Flexible Flowshop with limited Buffers

3.2 Problem Assumptions

Because scheduling problems have a vast amount of variation, the following assumptions are made for the problem under consideration.

1. The number of jobs to be scheduled and their processing times on each machine at each stage is known in advance and fixed.
2. The number of stages and the machine configuration at each stage are known in advance and fixed.
3. Preemption is not allowed.
4. Once a job has started processing, it must be completely finished before it can move to the next stage.
5. No job splitting is allowed, a job must be processed on one and only one machine at each stage.
6. All jobs are ready to begin processing at time period 0.

7. Jobs may or may not be scheduled in the same order at each stage, i.e. job passing is allowed.
8. Setup times are sequence dependent. There is also a setup time associated with startup.
9. Setup times are the same for all machines within each stage regardless of which machine the job is to be processed on.
10. The setup times uniformly range from 20% to 40% of the job processing time.
11. The machines at each stage are uniform. The first machine is considered to be the fastest machine with a speed of 100%, the other machines have either the same or a slower speed than the first (e.g., 60%, 50%, or 40%). The processing time is found by dividing the base processing time by the machine speed.
12. There are unlimited buffers before the first stage and after the last stage, but between stages the buffers are limited. The buffer is numbered the same as the stage that it feeds into.
13. Machine blocking does occur, so when a job is finished processing, it cannot leave the machine unless there is room on a machine at the next stage or the buffer for the next stage has an open space.
14. There are no due dates associated with the jobs and the objective is to minimize the makespan.

3.3 Objectives and Methodology of Research

There are four main objectives to this research:

1. To develop a mathematical model with the objective of minimizing the makespan for the flexible flowshop with sequence dependent setup times, uniform machines, and limited buffers.
2. Since this problem is considered strongly NP-Hard, develop heuristics to solve this problem. First two construction algorithms are developed to obtain solutions.
3. After the construction algorithm is developed; use a simulated annealing heuristic to improve the solutions obtained from the construction algorithms.
4. Since this problem is very complex even small problems may not be able to be solved optimally in reasonable time. A lower bound will be developed and used to test the performance of the heuristic.

4 Mathematical Model

Only small problems can be solved optimally, and even with small problems the computation time is great. Even though only small problems can be solved by using the mathematical model, there are still benefits to developing and studying it. Developing the mathematical model helps one better understand the problem complexity and structure. The difficulties that will arise in choosing a heuristic will usually come to the surface after working through the mathematical model. The model hence has two main benefits, better understanding of the problem and foreshadowing of issues with heuristic development. Variables used in the mathematical model are listed in Table 4.1.

Table 4-1 Mathematical Model Nomenclature

Variable	Notation	Definition
Parameters	*j,i*	Job index.
	J	Total number of jobs.
	s	Stage index.
	m	Machine index.
	t	Time period index.
	M*s*	Number of machines in stage *s*.
	P*sjm*	Processing time of job *j* on machine *m* at stage *s*.
	SUP*ijs*	Setup time from job *i* to job *j* at stage *s*.
	B*s*	Buffer capacity at Stage *s*.
	M	Large Number.
	T	Makespan of arbitrary solution.
	S	Total number of stages.
Integer Decision Variables	C*sjm*	Departure time of job *j* from machine *m* in stage *s*.
	K*sj*	Departure time of job *j* from stage *s*.
	F*sj*	The processing and blocking time of job *j* in stage *s*.
	E	Makespan

Variable	Notation	Definition
Binary Decision Variables	X*sjm*	=1 if job *j* is processed on machine *m* in stage *s*, = 0 otherwise.
	W*sijm*	=1 if job *j* directly follows job *i* on machine *m* at stage *s*, = 0 otherwise.
	D1*sjt*	= 1 if time period *t* precedes the starting time of job *j* at stage *s*, =0 otherwise.
	D2*sjt*	= 1 if time period *t* is after the departure time of job *j* at stage *s*-1, = 0 otherwise.
	D*sjt*	=1 if job *j* is in the buffer preceding stage *s* at time *t*, =0 otherwise.

Objective Function

Minimize makespan, E.

Constraints:

There is a zero job that is used to define the setup time for the first job to be processed on each machine. The zero job has no value for processing time, but the setup times from zero to other jobs vary.

The first constraint ensure that the completion time of job *j* is greater than or equal to the processing and setup time for job *j* in the first stage. The second constraint guarantees that if job *j* is processed on machine *m* in stage *s*, the completion time of job *j* is greater than or equal to the completion time of the preceding job plus the processing and setup time for job *j*. Constraint two also assures that the jobs are not interrupted during processing. The third constraint ensures that if job *j* is the first job processed on machine *m* in stage *s* then the completion time is greater than or equal to the processing and setup time for job *j* plus the completion time for job *j* in stage *s-1*.

$$C_{sjm} \geq P_{sjm} * X_{sjm} + W_{sijm} * SUP_{ijs} \; ; \forall \; j, \; i \neq j, \; s=1, \; m=1..Ms \qquad (1)$$

$$C_{sjm} - C_{sim} + \mathrm{M} * (1 - W_{sijm}) \geq P_{sjm} + W_{sijm} * SUP_{ijs} \; ; \forall \; s, j, \; i \neq j, \; m=1..Ms \qquad (2)$$

$$C_{sjm} - K_{(s-1)j} \geq P_{sjm} - \mathrm{M} * (1 - X_{sjm}) + W_{sijm} * SUP_{ijs} \; ; \; \forall \; s \neq 1, j, \; i \neq j, \; m=1..Ms \qquad (3)$$

The fourth and fifth constraints set the value for K_{sj} as the time that job *j* leaves stage *s*.

$$C_{sjm} \leq K_{sj} \; ; \; \forall \; s, j, \; m=1..Ms \qquad (4)$$

$$C_{sjm} \geq K_{sj} - \mathrm{M} * (1 - X_{sjm}); \; \forall \; s, j, \; m=1..Ms \qquad (5)$$

Constraint six sets the makespan equal to the maximum finishing time of all jobs at the final stage.

$$K_{(S)j} \leq E; \; \forall \; j \qquad (6)$$

Constraint seven ensures that each job is processed on one and only one machine in each stage.

$$\sum_{m=1}^{Ms} X_{sjm} = 1; \; \forall \; s, j \neq 0 \qquad (7)$$

Constraint eight ensures that every job *j* must follow another job *i*.

$$X_{sjm} - \sum_{i<>j}^{J} W_{sijm} = 0; \; \forall \; s, j \neq 0, \; m=1..Ms \qquad (8)$$

Constraint nine states that each job *i* may or may not be followed by another job *j*, but it can only be followed by at most one job.

$$X_{sim} - \sum_{j<>i}^{J} W_{sijm} \geq 0; \; \forall \; s, \; i, \; m=1..Ms \qquad (9)$$

Constraint ten sets the processing and blocking time for job *j* in stage *s*. It is the difference of the finishing time of job *j* and the preceding job *i* in stage *s*. F_{sj} is used in calculating the contents of the buffers.

$$F_{sj} \leq K_{sj} - K_{si} + M * (1 - W_{sijm}); \ \forall \ s, j, i \neq j, m = 1..Ms \qquad (10)$$

Constraints eleven, twelve, and thirteen are the buffer constraints. $D1_{sjt}$ is set to equal 1 if time period *t* is before the start of processing job *j* in stage *s*. $D2_{sjt}$ is set to equal 1 if job *j* has left stage (*s-1*). Constraint fourteen sets D_{sjt} equal to 1 only when $D1_{sjt}$ and $D2_{sjt}$ are both equal to 1. If D_{sjt} is equal to 1 then job *j* is in the buffer before stage *s* at time *t*, because job *j* has left stage (*s-1*) but it has not started processing on stage *s*. From the time a job enters the system until it leaves it is being setup, processed, blocked, or in a buffer.

$$K_{sj} - F_{sj} - t - M * D1_{sjt} + 1 \leq 0; \ \forall \ s, j, t \qquad (11)$$

$$t - K_{(s-1)j} - M * D2_{sjt} \leq 0; \ \forall \ s \neq 1, j, t \qquad (12)$$

$$D_{sjt} = D1_{sjt} + D2_{sjt} - 1; \qquad \forall \ s \neq 1, j, t \qquad (13)$$

Constraint fourteen sets the limits on the buffer's capacity. For each stage there is a maximum limit on the number of jobs that will fit into the buffer *Bs*.

$$\sum_{j}^{J} D_{sjt} \leq Bs; \ \forall \ s \neq 1, t \qquad (14)$$

Constraints fifteen and sixteen ensure that job 0 is processed first on every machine. This is needed to calculate the setup times for the first job processed on each machine at each stage.

$$X_{s0m} = 1; \qquad \forall\ s, m=1..Ms \qquad (15)$$

$$\sum_{i<>j}^{J} W_{si0m} = 0; \quad \forall\ s, j, m=1..Ms \qquad (16)$$

5 Heuristics

Minimizing the makespan in a flexible flowshop with sequence dependent setup times, uniform machines, and limited buffers is considered to be strongly NP-Hard; exact methods are only practical for extremely small problems. To solve problems of this complexity, heuristic algorithms are needed. Heuristics are algorithms designed to give a good solution to a problem in a reasonable amount of time. The solution given by a heuristic may not be optimal, but it should be relatively close to optimal. Heuristics sacrifice some solution quality, but make up for it by being superior in computation time. For example, to optimally solve a three stage problem with three machines in the first stage, one machine in the second stage, one machine in the third stage, and 5 jobs took around one hour, but by using the heuristics given in this research, a five stage, thirty job problem can be solved in less than three minutes.

Two construction heuristics are developed in this research. The solutions from each construction heuristic are used in a simulated annealing meta-heuristic, that is designed for solution improvement. The two construction heuristics, RBFFS and PBFFS, use the same solution representation, but the jobs are assigned to machines in different ways. RBFFS is route based while PBFFS is job priority based. The simulated annealing meta-heuristic uses the same solution representation as the construction heuristics, and the same job/machine assignments corresponding to the construction heuristics. The solutions are represented by a priority permutation of the jobs to be scheduled in the system. This representation limits the solution space to $n!$ solutions, where n is the number of jobs. For example if there are five jobs to be scheduled in the system, a solution could be represented as {2, 3, 1, 5, 4}.

The construction heuristics are based on the NEH algorithm developed by Nawaz, Enscore, and Ham (1983). The NEH algorithm was developed for the multistage flowshop environment. It gives higher priority to jobs with larger total processing times. A priority list is made by arranging the jobs in decreasing order of their total processing time on all machines. Next, the two jobs with the highest priority are chosen and the makespans for job 1 followed by job 2 and for job 2 followed by job 1 are calculated. The sequence with the smallest makespan is chosen as the best sequence, either 1-2 or 2-1, then the job with the third highest priority enters the system. When job 3 enters the system, adjacent pairwise exchange is used to create three new partial sequences, and the makespan for each partial sequence is calculated. The partial sequence with the smallest makespan is chosen as the best three job sequence. This process continues until all jobs are scheduled. This heuristic looks at only $\{[n\ (n+1) / 2] - 1\}$ permutation solutions as opposed to all $n!$ solutions that exist, where n equals the number of jobs. Many researchers reference this algorithm as one of the best for constructing a solution for the flowshop scheduling problem. A modified version of the NEH algorithm is used in this research to construct a solution for the flexible flowshop with sequence dependent setup times, uniform machines, and limited buffers.

5.1 *RBFFS Construction Heuristic*

The RBFFS construction heuristic uses a priority routing system that assigns jobs specific routes through the system. A route is a path that the job will travel through the system. A job's route dictates the machine it will visit at each stage, and the job that it will follow on that machine. The routes are designed so that the machines in each stage

will operate for, as much as possible, the same amount of time. For the purpose of assigning the routes, all jobs are assumed to have the average base processing time. A ratio is then used with the machine speeds to decide on how many jobs will visit each machine. For example, if there are two machines at a stage, the first having a speed of 1.00 and the second a speed of 0.50, the ratio would equal: [(1.00 / 0.50) = 2]. By using this ratio, 2 jobs would be processed on machine one for every 1 job processed on machine two. In this heuristic, the order in which jobs will be processed on each machine is predetermined; it is based on the routing system, the machine configuration, and the job priority. Each job is given a route and follows that route through the system. In this heuristic, a machine may be idle sometimes while it is waiting for a specific job, even though other jobs may be ready for processing. In some cases, inserted idle time can be beneficial to the system.

The steps for the RBFFS construction algorithm are as follows:

1. Sum the processing times on the fastest machine at each stage for each job.
2. Arrange the jobs in descending order of the sums of their processing times. The jobs with the greatest sum of processing times are given the highest scheduling priority.
3. Determine the routes that will be used by the jobs in order to navigate through the system; details of route assignments are shown in the following example. The makespan is calculated by using the job processing times and the routes associated with each job. When the jobs are placed on different routes the makespan may increase or decrease.

4. Take the two jobs with the highest priority and schedule them, using pairwise adjacent exchange. Calculate the makespan for each configuration and set the best RBFFS sequence as the sequence with the smallest makespan.
5. Choose the job with the next highest priority and enter it at the end of the best RBFFS sequence. Use pairwise adjacent exchange and calculate the makespan for each partial sequence. Set the best RBFFS sequence as the partial sequence with the smallest makespan.
6. If all jobs have been scheduled go to step 7, else go to step 5.
7. Set the best n job sequence as the RBFFS sequence and the best makespan as the RBFFS makespan. STOP.

5.2 *RBFFS Construction Heuristic Example*

An example illustrating the construction heuristic is given in this section. It is an instance with 4 jobs that need to be scheduled in a 3 stage environment with 3 machines at the first stage, 1 machine at the second stage, and 2 machines at the third. The buffers are limited to three jobs. The first machine at any stage is the fastest and considered to have a speed of 1.00, any other machine has a speed of 0.50. This means that the fastest machine is twice as fast as the other machines. The processing and setup times are given below.

Table 5-1 Construction Heuristic Ex. Process Times Stage 1

Processing Times Stage 1				
	Job 1	Job 2	Job 3	Job 4
Machine 1	4	14	16	10
Machine 2	8	28	32	20
Machine 3	8	28	32	20

Table 5-2 Construction Heuristic Ex. Setup Times Stage 1

Setup Times Stage 1				
	To 1	To 2	To 3	To 4
From 0	1	4	5	3
From 1	0	3	3	4
From 2	1	0	6	3
From 3	1	3	0	3
From 4	1	3	3	0

Table 5-3 Construction Heuristic Ex. Process Times Stage 2

Processing Times Stage 2				
	Job 1	Job 2	Job 3	Job 4
Machine 1	4	8	10	7

Table 5-4 Construction Heuristic Ex. Setup Times Stage 2

Setup Times Stage 2				
	To 1	To 2	To 3	To 4
From 0	2	2	3	3
From 1	0	3	4	2
From 2	2	0	4	2
From 3	1	2	0	2
From 4	1	2	2	0

Table 5-5 Construction Heuristic Ex. Process Times Stage 3

Processing Times Stage 3				
	Job 1	Job 2	Job 3	Job 4
Machine 1	6	9	14	9
Machine 2	12	18	28	18

Table 5-6 Construction Heuristic Ex. Setup Times Stage 3

Setup Times Stage 3				
	To 1	To 2	To 3	To 4
From 0	2	2	4	2
From 1	0	4	3	4
From 2	1	0	4	2
From 3	1	2	0	2
From 4	1	2	3	0

Step one, two, and three: sum the processing times on the fastest machine at each stage for each individual job. Then set the RBFFS priority list by placing the jobs in descending order of processing time sums.

Table 5-7 Step 1 of RBFFS Example.

Step 1: Sum processing time for each job on the fastest machine.				
	Stage 1	Stage 2	Stage 3	Σ pt
Job 1	4	4	6	14
Job 2	14	8	9	31
Job 3	16	10	14	40
Job 4	10	7	9	26

Table 5-8 Step 2 of RBFFS Example.

RBFFS Priority List	
First	Job 3
Second	Job 2
Third	Job 4
Fourth	Job 1

The proposed heuristic uses a priority routing system, Table 5.9 shows the priority routing for the example problem. By using the ratios in stage one, 2 jobs will be processed on machine one, for each job processed on machine two and three. Since there is only one machine at the second stage all of the jobs will be processed on it. At the third stage, 2 jobs will be processed on machine one for each job processed on machine two.

Table 5-9 Priority Routing for RBFFS Example.

	Stage 1	Stage 2	Stage3
Priority Route* 1	1-0	1-0	1-0
Priority Route* 2	1-1	1-1	2-0
Priority Route* 3	2-0	1-2	1-1
Priority Route* 4	3-0	1-3	1-3
Key: The first digit is the machine that processes the job; the second digit is index of the job that it follows.			
* It should be noted that the priority routing system for each machine configuration is unique, but they do have the common goal of machine utilization. The priority routing system is specifically designed for each problem instance.			

Step four and five: start with the first two jobs on the RBFFS priority list, use pairwise adjacent exchange, calculate the makespans and choose the best. The first two jobs on the RBFFS priority list are job 3 and job 2, so the makespans for the combinations {3, 2} and {2, 3} will be calculated. Table 5.10 and 5.11 show the Gantt Charts for partial sequences {3, 2} and {2, 3} respectfully.

Table 5-10 RBFFS Makespan for Sequence {3, 2}

		20	21	22	23	24	25	33	34	35	36	37	38	39	47	48	49	50	52	53	67	68
Stage 1	mc 1																					
	mc 2																					
	mc 3																					
Stage 2	mc 1																					
Stage 3	mc 1																					
	mc 2																					
		20	21	22	23	24	25	33	34	35	36	37	38	39	47	48	49	50	52	53	67	68

Key	
Job 1	
Job 2	
Job 3	
Job 4	

Table 5-11 RBFFS Makespan for Sequence {2, 3}

		15	16	17	18	19	20	27	28	29	30	38	39	40	41	42	54	55	84	85	86
Stage 1	mc 1																				
	mc 2																				
	mc 3																				
Stage 2	mc 1																				
Stage 3	mc 1																				
	mc 2																				
		15	16	17	18	19	20	27	28	29	30	38	39	40	41	42	54	55	84	85	86

Since this is such a small example the limited buffers do not cause any machine blockage. The partial solution {3, 2} is chosen with a makespan of 68. The next job to inter is job 4, so the partial solutions that will be considered are: {3, 2, 4}, {3, 4, 2}, and {4, 3, 2}. Tables 5.12, 5.13, and 5.14, show the Gantt charts representing partial sequences {3, 2, 4}, {3, 4, 2}, and {4, 3, 2} respectfully.

Table 5-12 RBFFS Makespan for Sequence {3, 2, 4}

		20	21	22	23	24	33	34	35	36	37	38	39	40	47	48	49	50	51	52	53	56	57	58	67	68
Stage 1	mc 1																									
	mc 2																									
	mc 3																									
Stage 2	mc 1																									
Stage 3	mc 1																									
	mc 2																									
		20	21	22	23	24	33	34	35	36	37	38	39	40	47	48	49	50	51	52	53	56	57	58	67	68

Key	
Job 1	
Job 2	
Job 3	
Job 4	

Table 5-13 RBFFS Makespan for Sequence {3, 4, 2}

		20	21	22	23	31	32	33	34	35	36	42	43	44	45	52	53	54	55	62	63	64
Stage 1	mc 1																					
	mc 2																					
	mc 3																					
Stage 2	mc 1																					
Stage 3	mc 1																					
	mc 2																					
		20	21	22	23	31	32	33	34	35	36	42	43	44	45	52	53	54	55	62	63	64

Table 5-14 RBFFS Makespan for Sequence {4, 3, 2}

		12	13	14	15	22	23	24	25	31	32	33	34	35	43	44	45	46	53	54	55	64	65	66	67	76
Stage 1	mc 1																									
	mc 2																									
	mc 3																									
Stage 2	mc 1																									
Stage 3	mc 1																									
	mc 2																									
		12	13	14	15	22	23	24	25	31	32	33	34	35	43	44	45	46	53	54	55	64	65	66	67	76

Key	
Job 1	
Job 2	
Job 3	
Job 4	

The partial solution {3, 4, 2} is chosen with a makespan of 64. The next job to inter is job 1, so the partial solutions that will be considered are: {3, 4, 2, 1}, {3, 4, 1, 2}, {3, 1, 4, 2}, and {1, 3, 4, 2}. Tables 5.15, 5.16, 5.17, and 5.18 show the Gantt charts representing partial sequences {3, 4, 2, 1}, {3, 4, 1, 2}, {3, 1, 4, 2}, and {1, 3, 4, 2} respectfully.

Table 5-15 RBFFS Makespan for Sequence {3, 4, 2, 1}

		8	9	10	20	21	22	23	31	32	33	34	35	42	43	44	45	51	52	53	54	58	59	60	64	65	71
Stage 1	mc 1																										
	mc 2																										
	mc 3																										
Stage 2	mc 1																										
Stage 3	mc 1																										
	mc 2																										
		8	9	10	20	21	22	23	31	32	33	34	35	42	43	44	45	51	52	53	54	58	59	60	64	65	71

Table 5-16 RBFFS Makespan for Sequence {3, 4, 1, 2}

		8	9	10	21	22	32	33	34	35	43	44	47	48	49	51	52	53	58	59	60	63	71	72
Stage 1	mc 1																							
	mc 2																							
	mc 3																							
Stage 2	mc 1																							
Stage 3	mc 1																							
	mc 2																							
		8	9	10	21	22	32	33	34	35	43	44	47	48	49	51	52	53	58	59	60	63	71	72

Key	
Job 1	
Job 2	
Job 3	
Job 4	

Table 5-17 RBFFS Makespan for Sequence {3, 1, 4, 2}

		20	21	22	23	24	26	27	32	33	34	38	39	40	48	49	52	53	57	58	59	63	73	74
Stage 1	mc 1																							
	mc 2																							
	mc 3																							
Stage 2	mc 1																							
Stage 3	mc 1																							
	mc 2																							
		20	21	22	23	24	26	27	32	33	34	38	39	40	48	49	52	53	57	58	59	63	73	74

Table 5-18 RBFFS Makespan for Sequence {1, 3, 4, 2}

		4	5	11	12	18	19	20	24	25	31	32	33	38	39	47	48	56	57	58	59	60	61	70	71
Stage 1	mc 1																								
	mc 2																								
	mc 3																								
Stage 2	mc 1																								
Stage 3	mc 1																								
	mc 2																								
		4	5	11	12	18	19	20	24	25	31	32	33	38	39	47	48	56	57	58	59	60	61	70	71

The final RBFFS solution is chosen as {3, 4, 2, 1} or {1, 3, 4, 2}, since there is a tie, either solution can be chosen. Arbitrarily solution {3, 4, 2, 1} is chosen, so job 3 is given priority route* 1 and job 4 is given priority route* 2, etc. The final makespan, or objective function is represented simply as 71.

Step 7: the RBFFS sequence is {3, 4, 2, 1} and the corresponding makespan is 71 units. STOP

The final RBFFS sequence is then used as the starting solution in a simulated annealing algorithm.

5.3 *RBFFS Simulated Annealing Heuristic*

Simulated annealing is a technique used to find good solutions to combinatorial problems; the solutions may or may not be optimal. It originated from the physical annealing of solids, and the connection between annealing and mathematical problems was first made by Metropolis et al. (1953). Later, Kirkpatirick et al. (1983) used it to solve combinatorial optimization problems, and it has shown good results. It looks at random variations of the current solution. Sometimes a non-improving solution is accepted as the new solution with a probability that decreases as the algorithm progresses. By sometimes accepting a non-improving solution, the simulated annealing algorithm avoids being trapped in local optima and can search different areas in the solution space for the global optima. Many researchers have had good results in using simulated annealing to solve flowshop scheduling problems.

The simulated annealing algorithm takes the final sequence from the RBFFS construction algorithm and searches for improved solutions by rearranging the sequence. This SA algorithm uses the same solution representation as the RBFFS heuristic, so the solution space is also limited to $n!$ solutions while in reality there may be many more possible solutions. The solutions are listed as permutations of jobs, and use the priority route* based scheduling as in the RBFFS construction heuristic. The nomenclature used in the simulated annealing heuristic is given in Table 5.19.

Table 5-19 Simulated Annealing Nomenclature

Variable	Definition
n	Total number of jobs.
m	Total number of stages.
T	Temperature
N	Number of iterations at the current temperature.
minT	Final temperature value.
maxN	Max number of iterations at each temperature setting.
RBFFSms	Makespan obtained from the construction heuristic.
RBFFSseq	Sequence of jobs obtained from the construction heuristic.
bestms	Lowest makespan
bestseq	Sequence of jobs corresponding to the lowest makespan.
curbestms	Current best makespan.
curbestseq	Sequence of jobs corresponding to the current best makespan
Y	Random integer value between 1 and n, including 1 and n.
Z	Random integer value between 1 and n, including 1 and n.
R	Random value between 0 and 1.
SAms	Simulated Annealing makespan.
SAseq	Simulated Annealing sequence.
ΔTC	Value of (currbestms – SAms)
α	Temperature reduction factor.

The solution is represented in the SA heuristic the same way as in the construction heuristic, as a sequence of jobs such as {3, 4, 2, 1} where the first job is given priority route* 1, in this case job 3. The performance of each solution is based on its makespan, which is the completion time of the last job completed at stage *m*.

The steps for the SA algorithm are as follows.

1. Set the heuristic parameters *minT, maxN,* α*, T,* and *N*. The final temperature value *minT* dictates the length of the SA heuristic, i.e. the heuristic will run until *T* reaches *minT*. The maximum number of iterations at each temperature setting *maxN,* is one of the factors in determining how fast the temperature reduces. It regulates how many iterations are ran at a given probability of accepting a non-improving solution. The temperature reduction factor α, ranges between 0 and 1,

and is the other factor in determining how fast the temperature is reduced. The initial temperature *T* is a factor in the acceptance of a non-improving solution. The higher the temperature, the greater the probability of accepting a non-improving solution, which allows the heuristic to search different areas of the solution space. The initial value of *N,* the number of iterations at each temperature setting is set equal to 0.

2. Set *bestms* = *RBFFSms*, and *bestseq* = *RBFFSseq*. In this step the best solution and sequence are set to the solution and sequence obtained from the RBFFS construction algorithm. Also the current makespan and sequence are set equal to the construction heuristic's makespan and sequence, *curbestms* = *RBFFSms* and *curbestseq* = *RBFFSseq*.
3. Generate random values for *Y* and *Z.* The jobs in these two places are the jobs chosen for pairwise interchange; e.g. if *Y* = 10 and *Z* = 6, then the job assigned to priority route* 10 will be assigned to priority route* 6 and the job originally assigned priority route* 6 will now be assigned to priority route* 10. *Y* and *Z* represent locations in the priority list, and not actual job numbers.
4. If *Y* is equal to *Z* then go back to step 3; else go to step 5
5. Calculate the makespan after the exchange and set it equal to *SAms* also set the new sequence as *SAseq*.
6. Calculate ΔTC *(curbestms – SAms)*. In this step, the difference between the current best makespan and the simulated annealing makespan is calculated.
7. If $\Delta TC > 0.001$ then set *curbestms* = *SAms* and *curbestseq* = *SAseq*. Step 7 checks the difference between the *curbest* makespan and the simulated annealing

makespan, if it is greater than 0.001 then it means that the SA solution is better than the *curbest*. If this is the case then the *curbest* values are reset as the SA values. Also if $\Delta TC > 0.001$ then the simulated annealing solution is compared to the overall best solution found so far. If *SAms* < *bestms* then set *bestms* = *SAms* and *bestseq* = *SAseq*.

8. If $\Delta TC < 0.001$ and $e^{\Delta TC/T} >$ R, then set *curbestms* = *SAms* and *curbestseq* = *SAseq*. If the SA solution is non-improving, step 8 checks to see if it will be accepted anyway. *R* is a random value between 0.00 and 1.00. As the increase in the makespan gets smaller, the non-improving solution has a higher chance of being accepted. Also as *T* gets smaller the probability of accepting a non-improving solution decreases.
9. Set $N = N + 1$.
10. If $N = maxN$ then $T = \alpha * T$ and $N = 0$. This step checks how many iterations have taken place at the current temperature. If the maximum number of iterations at this temperature has been reached, *maxN*, then the value of *T* is reset to the current temperature multiplied by the temperature reduction factor. After the new temperature is calculated it is compared with the minimum temperature, if $T <= minT$, the heuristic terminates, else go back to step 3.

The process flow chart for the simulated annealing heuristic is shown in Figure 5.1.

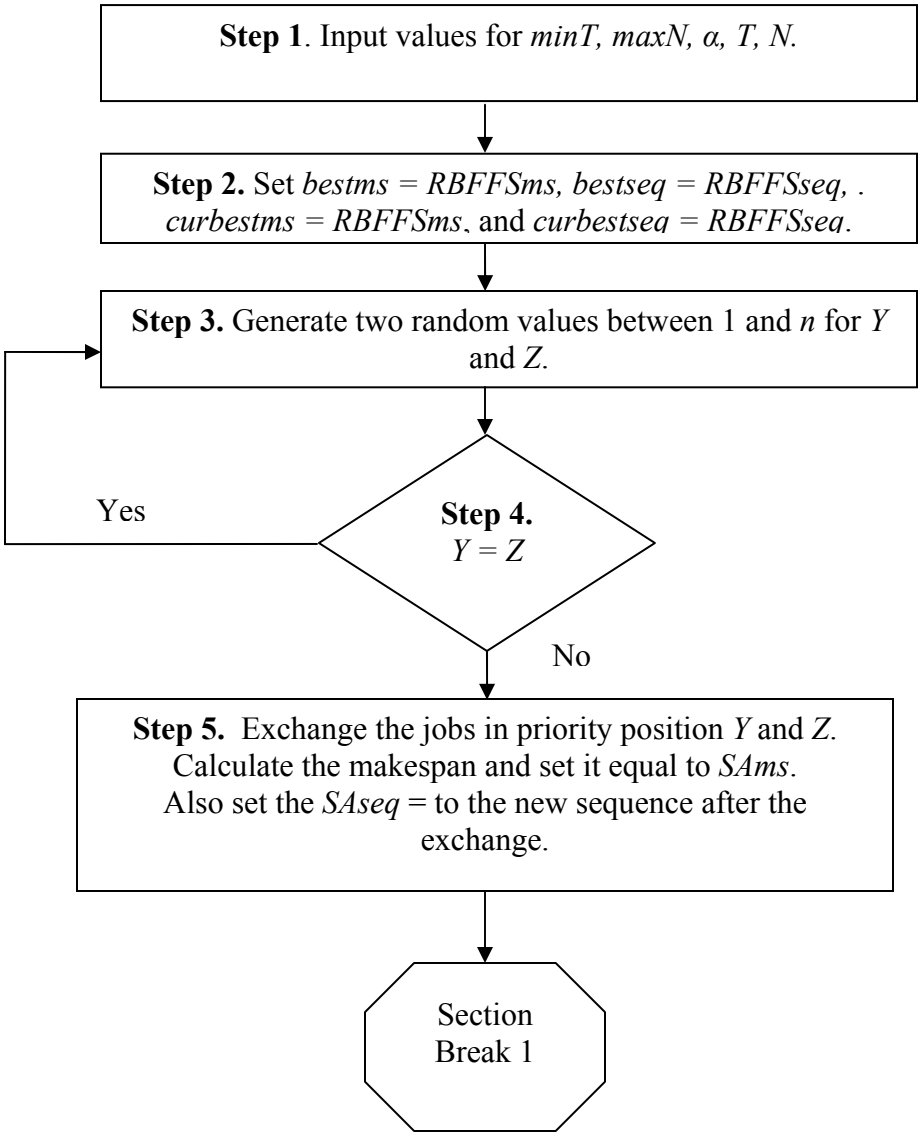


Figure 5.1 Simulated Annealing Process Flow Diagram

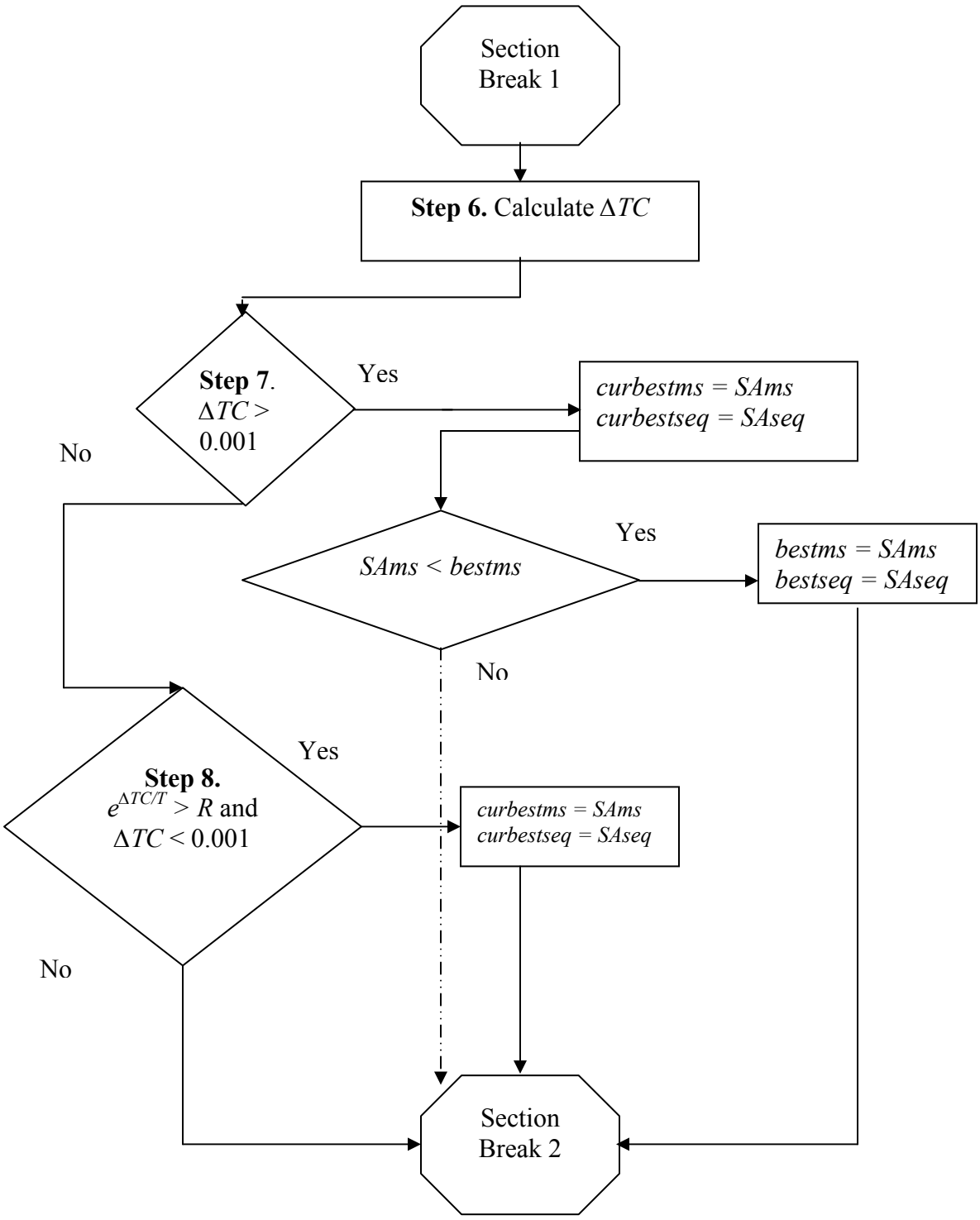


Figure 5.1 cont: Simulated Annealing Process Flow Diagram

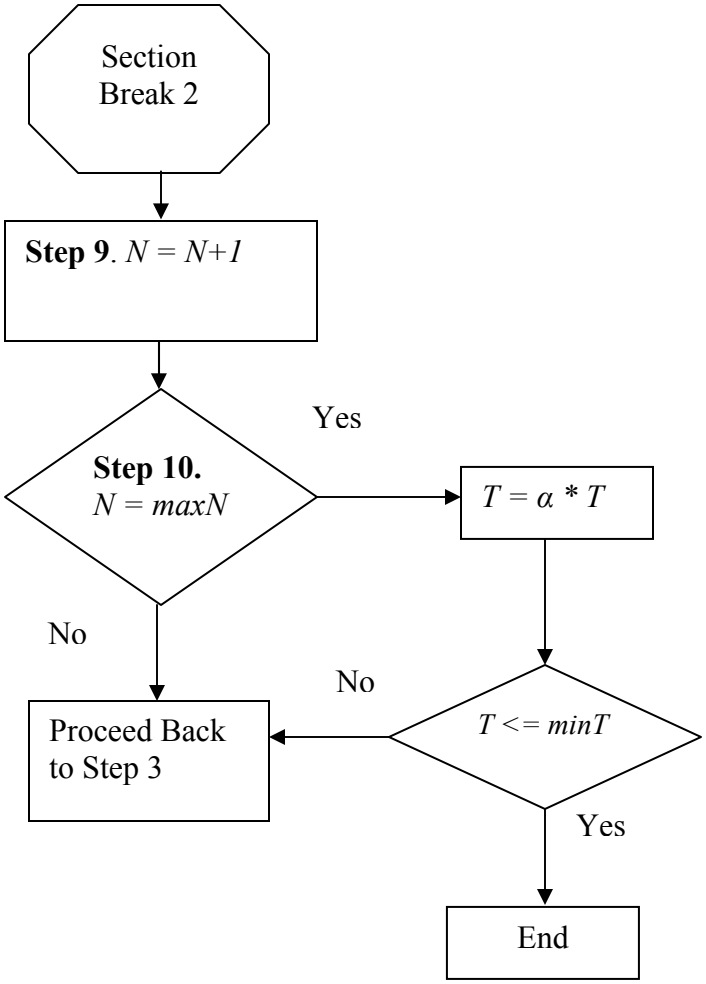


Figure 5.1 cont: Simulated Annealing Process Flow Diagram

5.4 *Route Based Simulated Annealing Example*

A simple illustration of the simulated annealing heuristic is now given. The illustration will use the same data as the RBFFS construction heuristic. The starting solution for the simulated annealing heuristic is the final solution of the RBFFS construction heuristic, so the starting solution for SA is {3, 4, 2, 1}.

Step 1: $minT = 80$, $maxN = 1$, $\alpha = 0.80$, $T = 100$, $N = 0$

Step 2: *bestms* = 71, *bestseq* = {3, 4, 2, 1}, *curbestms* = 71, *curbestseq* = {3, 4, 2, 1}

Step 3: $Y = 2$, $Z = 4$.

Step 4: $Y \neq Z$

Step 5: *SAseq* = {3, 1, 2, 4}, *SAms* = 74

Step 6: $\Delta TC = (curbestms - SAms) = (71 - 74) = -3$

Step 7: ΔTC is not greater than 0.001.

Step 8: $e^{\Delta TC/T} = e^{-3/100} = 0.97$, $R = 0.25$, Also $\Delta TC < 0.001$, so the non-improving solution

is accepted. *curbestms* = 74, *curbestseq* = {3, 1, 2, 4}.

Step 9: $N = 0 + 1 = 1$.

Step 10: $N = maxN$, so $T = 100 * 0.80 = 80$. Since $T <= minT$ the heuristic is stopped.

5.5 *PBFFS Construction Heuristic*

The PBFFS construction heuristic is a job priority based heuristic. Each job is given a priority index, and when a machine becomes open the available job with the lowest index is assigned to that machine. The priority given to each job is the same for all stages; it is an entire system priority. The solution is represented as a list of jobs with

the first job listed as having a priority of 1. For example a four job problem may have the solution {3, 4, 1, 2}, where job 3 has the highest priority, job 4 has the next highest, job 1 the next, and job 2 has the lowest priority. As with the other heuristic, this solution technique limits the solution space to *n*! solutions, where *n* is the number of jobs. The PBFFS heuristic has the same procedure as RBFFS except, the job assignments are not route based. In the PBFFS, jobs are scheduled based on two criteria, their priority index, and their availability for processing. The job order for each machine in this case is not known in advance; it depends on the job's priority and processing times.

The steps for the PBFFS heuristic are as follows:

1. Sum the processing times on the fastest machine at each stage for each job.
2. Arrange the jobs in descending order of the sums of their processing times. The jobs with the greatest sum of processing times are given the highest scheduling priority. Assign each job a priority, a number from 1 to *n*, where 1 is the highest priority.
3. Set all machine open times equal to 0. All jobs are initially ready for processing at stage 1, but will not become available for stage 2 until after they are finished processing at stage 1. This holds true for all stages 3, 4, etc.
4. Take the two jobs with the highest priority and schedule them, using pairwise adjacent exchange. Start scheduling jobs at stage one starting with the first machine; assign the available job with the highest priority. As jobs are scheduled and processed, the machine open time is calculated

as (the previous job completion time + job setup time + job processing time + any blocking time that may occur). Jobs are scheduled on the machine with the lowest open time, until all jobs have been processed through the system. If there is a tie on machine open time, the job will be processed on the machine with the smallest machine index, ex. mc 1, mc 2, or mc 3. Calculate the makespan for each configuration and set the best PBFFS sequence as the sequence with the smallest makespan.

5. Choose the job with the next highest priority and enter it at the end of the best PBFFS sequence. Use pairwise adjacent exchange and calculated the makespan for each partial sequence. Set the best PBFFS sequence as the partial sequence with the smallest makespan.
6. If all jobs have been scheduled, go to step 7. Else go to step 5.
7. Set the best *n* job sequence as the PBFFS sequence and the best makespan as the PBFFS makespan. STOP.

5.6 *PBFFS Construction Heuristic Example*

An illustration of the PBFFS construction heuristic is shown using the same job data that were used in the RBFFS example. The steps for PBFFS are the same as RBFFS except the jobs are scheduled in a different way. Since the example is small the limited buffers do not block any machines.

Step one two, and three: sum the processing times on the fastest machine at each stage, for each individual job. Then set the PBFFS priority list by place the jobs in descending order and assigning a priority index to each job.

Table 5-20 Step 1 of PBFFS Example.

Step 1: Sum processing time for each job on the fastest machine.				
	Stage 1	Stage 2	Stage 3	Σ pt
Job 1	4	4	6	14
Job 2	14	8	9	31
Job 3	16	10	14	40
Job 4	10	7	9	26

Table 5-21 Step 2 and 3 of PBFFS Example.

PBFFS Priority List	
First	Job 3
Second	Job 2
Third	Job 4
Fourth	Job 1

Step four and five: start with the first two jobs on the PBFFS priority list, use pairwise adjacent exchange, calculate the makespans and choose the best. The first two jobs on the PBFFS priority list are job 3 and job 2, so the makespans for the combinations {3, 2} and {2, 3} will be calculated. Table 5.22 and 5.23 show the Gantt Charts for partial priority lists {3, 2} and {2, 3} respectfully.

Table 5-22 PBFFS Makespan for Priority {3, 2}

		19	20	21	22	23	31	32	33	34	35	43	44	45	46	51	52	53	63	64
Stage 1	mc 1																			
	mc 2																			
	mc 3																			
Stage 2	mc 1																			
Stage 3	mc 1																			
	mc 2																			
		19	20	21	22	23	31	32	33	34	35	43	44	45	46	51	52	53	63	64

Key	
Job 1	
Job 2	
Job 3	
Job 4	

Table 5-23 PBFFS Makespan for Priority {2, 3}

		17	18	19	20	27	28	29	36	37	38	39	40	50	51	52	82	83
Stage 1	mc 1																	
	mc 2																	
	mc 3																	
Stage 2	mc 1																	
Stage 3	mc 1																	
	mc 2																	
		17	18	19	20	27	28	29	36	37	38	39	40	50	51	52	82	83

The partial solution {3, 2} is chosen with a makespan of 64. The next job to inter is job 4, so the partial solutions that will be considered are: {3, 2, 4}, {3, 4, 2}, and {4, 3, 2}. Tables 5.24, 5.25, and 5.26, show the Gantt charts representing partial sequences {3, 2, 4}, {3, 4, 2}, and {4, 3, 2} respectfully.

Table 5-24 PBFFS Makespan for Priority {3, 2, 4}

		20	21	22	23	24	32	33	34	35	43	44	45	51	52	53	54	64
Stage 1	mc 1																	
	mc 2																	
	mc 3																	
Stage 2	mc 1																	
Stage 3	mc 1																	
	mc 2																	
		20	21	22	23	24	32	33	34	35	43	44	45	51	52	53	54	64

Table 5-25 PBFFS Makespan for Priority {3, 4, 2}

		20	21	22	23	24	31	32	33	34	35	42	43	44	51	52	53	54	62	63	64
Stage 1	mc 1																				
	mc 2																				
	mc 3																				
Stage 2	mc 1																				
Stage 3	mc 1																				
	mc 2																				
		20	21	22	23	24	31	32	33	34	35	42	43	44	51	52	53	54	62	63	64

Table 5-26 PBFFS Makespan for Priority {4, 3, 2}

		12	13	14	23	24	32	33	34	35	37	38	42	43	44	56	57	61	62	63	73
Stage 1	mc 1																				
	mc 2																				
	mc 3																				
Stage 2	mc 1																				
Stage 3	mc 1																				
	mc 2																				
		12	13	14	23	24	32	33	34	35	37	38	42	43	44	56	57	61	62	63	73

The partial solution {3, 2, 4} and {3, 4, 2} each have a makespan of 64, so either partial sequence can be used. Arbitrarily the partial sequence {3, 4, 2}, is chosen. The next job to enter is job 1, so the partial solutions that will be considered are: {3, 4, 2, 1}, {3, 4, 1, 2}, {3, 1, 4, 2}, and {1, 3, 4, 2}. Tables 5.27, 5.28, 5.29, and 5.30 show the Gantt charts representing partial sequences {3, 4, 2, 1}, {3, 4, 1, 2}, {3, 1, 4, 2}, and {1, 3, 4, 2} respectfully.

Table 5-27 PBFFS Makespan for Priority {3, 4, 2, 1}

		21	22	23	24	26	27	32	33	34	35	43	44	51	52	53	54	59	60	63	64	65	76
Stage 1	mc 1																						
	mc 2																						
	mc 3																						
Stage 2	mc 1																						
Stage 3	mc 1																						
	mc 2																						
		21	22	23	24	26	27	32	33	34	35	43	44	51	52	53	54	59	60	63	64	65	76

Table 5-28 PBFFS Makespan for Priority {3, 4, 1, 2}

		8	9	10	15	16	21	22	23	24	34	35	40	41	43	44	53	54	55	56	57	66	67
Stage 1	mc 1																						
	mc 2																						
	mc 3																						
Stage 2	mc 1																						
Stage 3	mc 1																						
	mc 2																						
		8	9	10	15	16	21	22	23	24	34	35	40	41	43	44	53	54	55	56	57	66	67

Table 5-29 PBFFS Makespan for Priority {3, 1, 4, 2}

		8	9	10	15	16	21	22	23	24	34	35	40	41	43	44	53	54	56	57	65	66	67
Stage 1	mc 1																						
	mc 2																						
	mc 3																						
Stage 2	mc 1																						
Stage 3	mc 1																						
	mc 2																						
		8	9	10	15	16	21	22	23	24	34	35	40	41	43	44	53	54	56	57	65	66	67

Table 5-30 PBFFS Makespan for Priority {1, 3, 4, 2}

		4	5	6	11	12	19	22	23	24	31	32	33	36	37	42	43	52	53	54	55	56	57	73	74
Stage 1	mc 1																								
	mc 2																								
	mc 3																								
Stage 2	mc 1																								
Stage 3	mc 1																								
	mc 2																								
		4	5	6	11	12	19	22	23	24	31	32	33	36	37	42	43	52	53	54	55	56	57	73	74

Key	
Job 1	
Job 2	
Job 3	
Job 4	

The final PBFFS solution is chosen as {3, 4, 1, 2} or {3, 1, 4, 2}, since there is a tie, either solution can be chosen. Arbitrarily solution {3, 4, 1, 2}is selected. The final makespan, or objective function is represented simply as 67.

Step 7: the PBFFS sequence is {3, 4, 1, 2} and the corresponding makespan is 67 units. STOP

The final PBFFS sequence is then used as the starting solution in a simulated annealing algorithm. The simulated annealing algorithm is the same as the one used in conjunction with the RBFFS construction heuristic, and it adjusts the sequence in the same way; the only difference in the development of the schedules is when the makespans are calculated. The makespans are calculated using the method from the PBFFS construction heuristic.

6 Lower Bound

The flexible flowshop with sequence dependent setup times, uniform machines, and limited buffers is a very complex problem, where only extremely small problems can be solved optimally. If the optimal solution cannot be found, other means for measuring the performance of the heuristics need to be employed. Three prominent ways of testing are using upper/lower bounds and comparing with other heuristics. In this research, lower bounds and comparison with other heuristics will be used. This section details how the lower bound is developed. The lower bound solution may be infeasible, but it gives a point of reference in testing the heuristic solutions. Due to the complexity of the problem, the lower bound presented is not very strong, and is not expected to be very close to the value of the optimal solution, but the complexity of the problem makes it very difficult to obtain a stronger lower bound. The lower bound presented was compared with LP relaxation for two randomly chosen medium sized problems, 4 stages and 30 jobs. The presented LB was 39.7% and 31.3% higher than the objective function of the LP relaxation solution.

Variable key for Lower Bound:

Table 6-1Lower Bound Nomenclature

Variable	Description
s	Stage index.
Ms	The number of machines at each stage *s*.
BPTs	Sum of base processing times for stage *s*.
PPs	Processing power of stage *s*.
Bs	*BPTs* divided by *PPs* for each stage *s*.
SSETs	Sum of the smallest setup times for all jobs at each stage *s*.
Ds	*SSETs* divided by *Ms* for each stage *s*.
Gs	The sum of *Bs* and *Ds* for each stage *s*, rounded up to the next integer.
Es	The largest sum (for any job) of the job basic processing time and its smallest setup time for each stage *s*.

Variable	Description
Ks	The maximum between *Gs* and *Es* for each stage *s*.
F	The fastest time any job can reach the bottleneck stage, including the setup times, from 0.
F_n	The n^{th} fastest time a job can reach the bottleneck stage.
Ls	The fastest estimated completion times for two jobs in each stage *s* leading up to the bottleneck stage. It is found by using the 2 jobs with the minimum base processing times and set up times at each stage. The jobs chosen at each of the stages are independent. The value for *Ls* is found by taking the maximum value between (sum processing times / machine processing power using the fastest 2 machines) + (smallest setup times / number of machines at stage *s* (with a maximum of 2 machines at stage *s*)) , and (largest base processing time + smallest setup time for job with largest base processing time)}
H	The sum of *Ls*, for all stages leading up to the bottleneck stage. ex. For stage 3 as the bottleneck, $H = L_1 + L_2$
SH	The earliest possible time processing can start on the second machine in the bottleneck stage. $SH = \max[H, F_2]$, where F_2 is the second smallest sum calculated in the determination of *F*.
I	*SH* minus *F*.
Os	Similar to *Ls* except that is uses the 3 smallest jobs at each stage.
J	The sum of *Os*, for all stages leading up to the bottleneck stage. ex. For stage 3 as the bottleneck, $J = O_1 + O_2$
SJ	The earliest possible time processing can start on the third machine in the bottleneck stage. $SJ = \max[J,F_3]$, where F_3 is the third smallest sum calculated in the determination of *F*.
Q	*SJ* minus *F*.
S2LB	Lower bound after step 2.
BPTBN	Sum of the Base processing times at the bottleneck stage.
PM2	Processing power of machine 2 at the bottleneck stage.
PM3	Processing power of machine 3 at the bottleneck stage.
PBS	Processing power of all machines at the bottleneck stage.
P	The smallest possible value for any job to finish processing after the bottleneck stage including setup.
LB	Lower Bound

An example of the lower bound will be worked out with its presentation. The data used in the example are listed in Tables 6.2 – 6.6. For the example problem a four

stage flexible flowshop with three machines in each of the stages is considered. There are four jobs to be processed, and for simplicity, all of the setup times in this illustration are considered to be equal to 1. Also in the example, the first machine at each stage has a speed of 100% and all other machines have a speed of 50%.

Table 6-2 Lower Bound Example Stage 1 Processing Times

Stage 1 Processing Times					
	Job 1	Job 2	Job 3	Job 4	∑ pt
Machine 1	4	5	3	8	20
Machine 2	8	10	6	16	40
Machine 3	8	10	6	16	40

Table 6-3 Lower Bound Example Setup Times

All Stages Setup Times				
	To Job 1	Job 2	Job 3	Job 4
From Job 0	1	1	1	1
Job 1	X	1	1	1
Job 2	1	X	1	1
Job 3	1	1	X	1
Job 4	1	1	1	X

Table 6-4Lower Bound Example Stage 2 Processing Times

Stage 2 Processing Times					
	Job 1	Job 2	Job 3	Job 4	∑ pt
Machine 1	6	5	6	2	19
Machine 2	12	10	12	4	38
Machine 3	12	10	12	4	38

Table 6-5 Lower Bound Example Stage 3 Processing Times

Stage 3 Processing Times					
	Job 1	Job 2	Job 3	Job 4	∑ pt
Machine 1	8	4	6	8	26
Machine 2	16	8	12	16	52
Machine 3	16	8	12	16	52

Table 6-6 Lower Bound Example Stage 4 Processing Times

Stage 4 Processing Times					
	Job 1	Job 2	Job 3	Job 4	∑ pt
Machine 1	2	6	5	2	15
Machine 2	4	12	10	4	30
Machine 3	4	12	10	4	30

6.1 *Step 1 of the Lower Bound*

The lower bound is developed in three steps. In the first step, the bottleneck stage is determined. To identify the bottleneck stage, the sum of base processing times for each job is calculated for each stage and is designated as (*BPTs).* Then (*BPTs)* is divided by the processing power at that stage (*PPs*), this becomes value (*Bs)*. The processing power is basically the sum of the machine speeds. A more detailed description of processing power is given in the next paragraph. The smallest setup time is determined for each job, and the smallest setup times are added for all jobs and set equal to (*SSETs).* Next *(SSETs),* is divided by the total number of machines in the stage (*Ms*); this value is labeled as (*Ds)*. The final value for this part of step one is found by adding (*Bs)* and (*Ds)* for each stage and rounding up to the next integer, this value becomes (Gs).

The processing power is calculated by summing the speeds of the machines. The processing times for jobs are calculated by dividing the base processing time by the speed of the machine. For example if a stage has three machines with speeds of 100%, 50%, and 50%, and a job has a base processing time of 6 units, the time to process that job on machine 1 will be (6/1) = 6 units, machine 2 and 3 (6/0.50) = 12 units. If a stage has only one machine then the processing power is 1, for a stage with two machines the processing power is (1.00 + 0.50) = 1.50, and for a stage with three machines the processing power is (1.00 + 0.50 + 0.50) = 2.00. Also the largest sum of any single job's base processing time and its smallest setup time is calculated for each stage and labeled as *(Es)*. For each stage, the fastest estimated completion time of all jobs is the largest value of either *(Gs)* or *(Es)* and is labeled as *(Ks)*. The first part of the lower bound determines the stage with the largest (Ks) which is the estimated completion time for that stage. The stage where

the maximum *(Ks)* occurs is considered to be the bottleneck stage and the rest of the lower bound is based around this stage. It should be noted that the bottleneck stage is estimated using lower bounds and may not be the actual bottleneck stage. Using the data given in Tables 6.2- 6.6, calculations for step one of the lower bound are as follows:

For first stage, K_1 = Max {(20/2) + (4/3), (8 + 1)} = Max {10 + 1.33, 9} = 12.

The second stage K_2 = Max {(19/2) + (4/3), (6 + 1)} = Max {9.5 + 1.33, 7} = 11.

The third stage K_3 = Max {(26/2) + (4/3), (8 + 1)} = Max {13 + 1.33, 9} = 15.

The fourth stage K_4 = Max {(15/2) + (4/3), (6 + 1)} = Max {7.5 + 1.33, 7} = 9.

Since the third stage has the largest calculated value of K_s, it is considered to be the bottleneck.

6.2 *Step 2 of the Lower Bound*

The second step in the lower bound calculation accounts for the idle time in the bottleneck stage, before the processing of the first job. If the bottleneck stage is not the first stage then some idle time will occur on each of the machines at the bottleneck stage while they are waiting for jobs to arrive from the previous stages. To find the earliest possible start time for any machine at the bottleneck stage, the base processing and setup up times from zero for each job, at each stage preceding the bottleneck stage, are summed over all stages, and then the smallest total value is chosen and designated as value *(F)*. Table 6.7 shows how this is calculated for the example problem:

Table 6-7 Step 2 Lower Bound

	Job 1	**Job 2**	**Job 3**	**Job 4**
Stage 1	4 + 1 = 5	5 + 1 = 6	3 + 1 = 4	8 + 1 = 9
Stage 2	6 + 1 = 7	5 + 1 = 6	6 + 1 = 7	2 + 1 = 3
Total	**12**	**12**	**11**	**12**

Job 3 would be chosen *(*with F = 11 units*),* so 11 idle units on all machines will be added to the lower bound. All machines at the considered bottleneck stage will be idle for at least 11 time units.

In addition to the idle time that all machines at the considered bottleneck stage will experience, additional idle time may occur on the second machine before it starts processing jobs. After the first machine at bottleneck stage begins processing, some additional idle time may occur on the other machines, because they may have to wait for the next available jobs to arrive. To account for the idle time on the second machine to begin processing at the bottleneck stage, the two jobs with the minimum values of the sum of basic processing times and the smallest setup times are chosen from each stage. The jobs chosen are independent for each stage, so if a job is chosen in one stage it does not have to be used at the following stages. After the jobs are chosen, an estimated completion time for each stage is obtained in a similar way as for value *(Ks)* in the first step of the lower bound. The estimated completion time for each stage *s* is denoted as (*Ls)*. The sum of the values for (*Ls*) leading up to the bottleneck stage is calculated and is denoted as (*H)*. The start time for the second machine at the bottleneck stage, denoted as *(SH)*. *SH* is calculated as the maximum value between [*H, F_2*]. The extra idle time for

the second machine at the bottleneck stage is calculated by subtracting *(F)* from *(SH)* and is denoted as value *(I),*where (*I)* is greater than or equal to zero.

For the example, job 3 and job 1 are chosen from the first stage, job 4 and job 2 are chosen from the second stage.

The first stage (L_1) value is Max {(7/1.5) + (2/2), (4 + 1)} = Max {4.67 + 1, 5} = 6.

The second stage (L_2) value is Max {(7/1.5) + (2/2), (5 +1)} = Max {4.67 + 1, 6} = 6.

The value for *(H)* is calculated as 6 + 6 = 12.

The earliest time for a job to reach machine two at the bottleneck stage *(SH)* is calculated as Max {12, 12} = 12. Since no jobs are ready for processing at the bottleneck stage until time period 11, and the earliest possible time a second job may be ready for processing is at time period 12, one extra idle unit (12 – 11 = 1) will be added to the second machine, *(I = 1)*.

The extra idle time for the third machine is calculated in the same way as for the second machine except the three jobs with the minimum values of the sum of the base processing time and smallest setup times are used. The estimated completion time for each stage *s* is labeled as *(Os).* The extra idle time for machine three is found by summing *(Os)* for each stage leading up to the bottleneck stage and is designated as *(J)*. The start time for the third machine at the bottleneck stage, denoted as (*SJ).* *SJ* is calculated as the maximum $[J,F_3]$. The extra idle time units to add to the third machine are found by subtracting *(F)* from *(SJ)* and this value is designated as *(Q)*, where *(Q)* is greater than or equal to zero.

For the example, jobs 3, 1, and 2 would be chosen from the first stage and job 4, 2, and 1 or 3 would be chosen from the second stage. The jobs at each stage are treated independently.

The first stage value of O_1 = Max {(12/2) + (3/3), (5 + 1)} = Max {6 + 1, 6} = 7.

The second stage value of O_2 = Max {(13/2) + (3/3), (6 + 1)} = Max {6.5 + 1, 7} = 8.

The value of J is calculated as 7 + 8 = 15.

The earliest time for a job to reach machine three in the bottleneck stage *(SJ)* is calculated as Max {12, 15}. Since the minimum time for the third job to reach the bottleneck stage is 15 time units, four extra idle units will be added to the starting time of the third machine at the bottleneck stage, (*Q)* is calculated as 15 – 11 = 4. This process is repeated until all lower bound extra idle times are determined for all of the machines at the bottleneck stage.

The last part of step 2 involves calculating the new lower bound after adding the idle times to the processing time for the bottleneck stage. This is done by using the following equation:

$$\text{Step 2 LB} = \text{Max}\left\{\left[\frac{BPTBN + (I \times PM2) + (Q \times PM3)}{PBS} + D_s\right], E_s\right\} + F$$

Step 2 LB	=	Lower bound after step 2.
BPTBN	=	Sum of the base processing times at the bottleneck stage.
PM2	=	Processing power of machine 2 at the bottleneck stage.
PM3	=	Processing power of machine 3 at the bottleneck stage.
PBS	=	Processing power of all machines at the bottleneck stage.

* It should be noted that the calculated value in the brackets [] is also rounded up.

For the example the lower bound value after step 2 would be calculated as follows:

$$\text{Step 2 LB} = \text{Max}\ \{[\underline{26 + (1 * 0.50) + (4 * 0.50)}\ + (4/3)], (8 + 1)\} + 11$$
$$2$$

$$= \text{Max}\ \{14.25 + 1.33, 9\} + 11$$
$$= 27$$

6.3 *Step 3 of the Lower Bound*

In the last step of the lower bound, extra time is added for the stages that follow the bottleneck stage. All jobs are considered. For each job, the base processing time for each of the post bottleneck stages is added to the smallest possible setup time, excluding the setup times from 0. The smallest value between all the jobs is labeled as *(P)*. *(P)* is then added to the lower bound that was calculated after step 2, and this value is the final lower bound, (*LB)*.

LB = Step 2 LB + P

For the example problem, the value for step three is calculated as follows:

Table 6-8 Step 3 Lower Bound

	Job 1	**Job 2**	**Job 3**	**Job 4**
Stage 4	2 + 1 = 3	6 + 1 = 7	5 + 1 = 6	2 + 1 = 3

Job 1 or 4 is chosen to be the final job processed, and 3 time units are added to the last value obtained in step 2. The final lower bound (*LB*) is calculated as 27 + 3 = 30. Even though the buffers in between stages have a limited capacity and blocking may occur; this could not be included in the calculations of the lower bound.

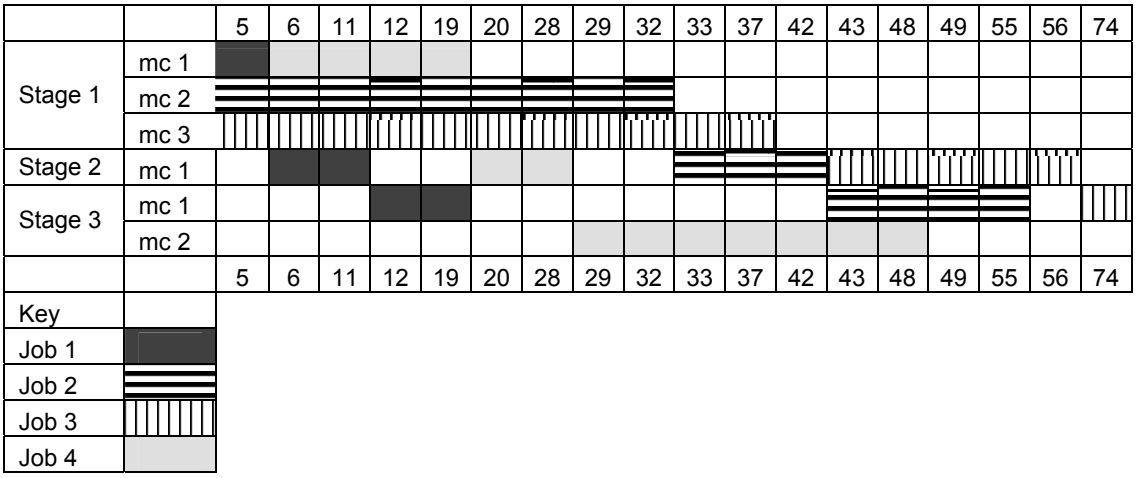
7 Simple Heuristics

Two quick and easy simple heuristics are also developed as another means of testing the solution quality of the RBFFS and PBFFS heuristics. The first simple heuristic SH1 assigns jobs priority based on the order in which they are placed (FCFS). The second simple heuristic SH2 allocates priorities to jobs based on the largest total processing time (LTPT). The simple heuristics then process the jobs through the system and record the maximum completion time. In both heuristics the jobs that are ready for processing are assigned to the first available machine based on the job's priority. The job with priority 1 is chosen over all others, then the job with priority 2, and so forth.

7.1 Simple Heuristic One

In the first simple heuristic, SH1, jobs are designated priorities on a first come first serve basis, and then they are processed through the system. In this heuristic the jobs are assigned a priority index as they enter the system, for example job 1 would receive priority 1, and so forth. Table 7.1 shows an example of the first come FCFS heuristic, SH1. The example is calculated using the same data that were used in the construction heuristic example in section 5.2. The SH1 job priority index is {1, 2, 3, 4}.

Table 7-1 SH1 Example Gantt Chart



The corresponding makespan to the SH1 sequence {1, 2, 3, 4} is calculated as 74.

7.2 *Simple Heuristic Two*

In the second simple heuristic SH2, the jobs are given priorities based on the largest processing time, with the largest one getting the highest priority. To calculate SH2, the LPT based simple heuristic, first the base processing times for each job at all stages must be added together. Table 7.2 shows the results of the summation of the base processing times.

Table 7-2 Sum of Base Processing Times

	Job 1	Job 2	Job 3	Job 4
Stage 1	4	14	16	10
Stage 2	4	8	10	7
Stage 3	6	9	14	9
Total	**14**	**31**	**40**	**26**

The priority list associated with this example would be {3, 2, 4, 1}. The Gantt Chart for the SH2 makespan is shown in Table 7.3.

Table 7-3 SH2 Example Gantt Chart

		21	22	23	24	26	27	32	34	35	44	45	52	53	54	58	59	64	65	71
Stage 1	mc 1																			
	mc 2																			
	mc 3																			
Stage 2	mc 1																			
Stage 3	mc 1																			
	mc 2																			
		21	22	23	24	26	27	32	34	35	44	45	52	53	54	58	59	64	65	71

Key	
Job 1	
Job 2	
Job 3	
Job 4	

The corresponding makespan for the SH2 sequence {3, 2, 4, 1} is 71 units. Both of these simple heuristics provide a feasible solution that can be used as a reference, when determining the solution quality of the more complicated RBFFS and PBFFS heuristics. For the same example data, the construction heuristics RBFFS and PBFFS makespans were 71 and 67 respectively.

8 Experimentation

Since this specific problem has not been addressed in the literature, a means of testing the proposed heuristics is needed. There are two methods that will be used to test the solution quality of each heuristic:

1. Comparison with a lower bound. By comparing with a lower bound, an idea is obtained on how good the solution is. Even though the lower bound may be infeasible and may be much lower than the optimal value, it is a good measuring tool for solution quality.
2. The second method used to test solution quality is to use other heuristics and compare the results. For comparison, two simple heuristics were developed. The first simple heuristic gives the jobs priority on a first come first serve basis. This would occur when jobs are sent to the end of the queue and wait for processing. The second simple heuristic gives jobs priority based on total processing time, where the jobs with larger total processing times get higher scheduling priority. While the lower bound may be infeasible, all heuristics developed in this research give feasible solutions.

8.1 *Problem Instances*

Three different simplified flexible flowshop environments were chosen to test the performance of the proposed heuristics. The first environment consists of 3 stages with a machine configuration of 3-1-2 and a limit of 3 jobs in the buffers between stages. This environment is modeled after part of a hydraulic hose manufacturing process. In the first

stage the hoses are attached to a harness, in the second stage the hoses are put in an oven for heat treating, and in the last stage the fittings are assembled.

The second environment consists of 4 stages with a machine configuration of 2-1-2-1 and a limit of 3 jobs in the buffers between stages. This environment is modeled after a vinyl window manufacture. In the first stage the pieces of vinyl are cut to length, in the second stage the four sides of the window are welded together, in the third stage the latches and tracks are added, and in the last stage the final assembly takes place.

The third environment consists of 5 stages with a machine configuration of 3-3-2-2-1 and a limit of 4 jobs in buffers between stages. This environment is modeled after a high voltage circuit breaker assembly manufacturer. In the first stage painting is preformed, in the second stage wiring takes place, in the third mechanical assembly is done, in the fourth electrical assembly takes place, and in the last step the unit is tested.

8.2 *Data Generation*

Data were not available for the jobs in these manufacturing environments, so they were generated. The first machine at each stage is considered to be the fastest with a speed of 1.00 all other machines have a speed of 0.50. The processing power of a stage is found by adding the speeds of all machines at that stage. The base processing time for a job at any stage is considered to be the job processing time on the fastest machine at that stage. The job setup time for any stage is between 20 to 40% of the job base processing time. The job processing times were generated as to not present any obvious bottleneck stages. If there were a bottleneck stage, the problem would basically reduce to a one stage scheduling problem. The data were generated as follows:

- If a stage has 1 machine, then the base processing time is chosen form a uniform distribution between 2 and 10. This gives an average processing time of 6 units. The average processing time divided by the machine processing power is (6 / 1) = 6 units.
- If a stage has 2 machines, then the base processing time is chosen from a uniform distribution between 3 and 15. This gives an average processing time of 9 units. The average processing time divided by the machine processing power = (9 / 1.5) = 6 units.
- If a stage has 3 machines, then the base processing time is chosen from a uniform distribution between 4 and 20. This gives an average processing time of 12 units. The average processing time divided by the machine processing power = (12 / 2) = 6 time units.
- Ten different data sets with 30 jobs each were generated for each of the three flowshop environments.

The nomenclature used in the experimentation is listed in Table 8.1.

Table 8-1 Experiment Nomenclature

Nomenclature	Definition
SH1	Refers to the simple heuristic, using first come first serve priority allocation.
SH2	Refers to the simple heuristic, using a largest total processing time priority allocation.
RBFFS	Refers to the route based construction heuristic.
PBFFS	Refers to the priority based construction heuristic
RBFFS_SA	Refers to the route based construction heuristic combined with the simulated annealing algorithm.
PBFFS_SA	Refers to the priority based construction heuristic combined with the simulated annealing algorithm.
LB	Lower bound

8.3 *Experimentation Overview*

Before applying the heuristics, the priority lists and the lower bounds are calculated using Microsoft Excel. A program was written in Microsoft Visual Basic-6.0 that takes the priority lists, job processing times, and setup times and calculates the solution for each heuristic. Also a VB program was written that performs all of the exchanges for the construction heuristics and displays the final sequence and the corresponding objective function. The program then inputs the final construction heuristic sequence and makespan as the starting sequence and makespan for the simulated annealing heuristic. The SA algorithm, also a VB program is initiated and runs until the temperature is reduced to a predetermined value.

8.4 *Simulated Annealing Parameters*

The simulated annealing parameters ensure that the temperature does not get reduced too fast, resulting in poor solutions. Also the SA parameters reduce the temperature fast enough that time is not wasted. The initial temperature is set by using the following equation:

$$IT = \frac{(LB - SH1)}{\ln(PA)}$$

Where,

IT = The initial temperature setting.
LB = Value obtained for the lower bound
SH1 = Value of the makespan found for the initial solution,
PA = Initial probability of accepting a non-improving solution,
For this research the probability is set as 40 or 30%.

The initial probability of accepting a non-improving solution is set at 40% for the first five runs (1-5) and set at 30% for the next five runs (6-10). The value for the

temperature reduction factor is set at 0.90. The maximum number of iterations at each temperature setting is 100 for each problem. The heuristic is stopped when the temperature is reduced to 0.01.

8.5 *Experimentation Results*

For each data set, the simple heuristic solutions and lower bounds were calculated first, and then the construction heuristics were initiated. Since the SA heuristic is random in nature and has a small run time it was run 10 different times. The PBFFS_SA heuristics run times were about 70, 95, and 135 seconds for the 3-1-2, 2-1-2-1, and 3-3-2-2-1 environments respectfully. The RBFFS_SA heuristics run times were all under 20 seconds for each environment. Each time the SA was run, it used the same starting solution given by the construction heuristic, but the initial probability of acceptance for a non-improving solution was adjusted. In the first five runs (1-5) the initial probability of acceptance was set at 40% and in the next five runs (6-10) the initial probability of acceptance was set at 30%. In Tables 8.2 through 8.4 only the best solution obtained from the 10 runs is shown, all results for each run can be viewed in appendix A. In these tables, the ratio to LB is calculated as the ratio between the solution in question and LB, and the ratio to BF is calculated as the ratio between the solution in question and the best solution found.

For the 3-1-2 machine scheduling environment, the RBFFS heuristic averaged an increase of 23.2% from the lower bound, 14.8% from the best solution found, and achieved a 1.2% improvement over the best simple heuristic solution. The RBFFS_SA heuristic averaged 7.3% from the lower bound, equaled the best solution found, and achieved a 16% improvement over the best simple heuristic solution. Also by adding simulated annealing to RBFFS, the solution improved by 14.8%. The PBFFS heuristic averaged 18.8% from the lower bound, 10.8% from the best solution found, and achieved a 4.7% improvement over the best simple heuristic solution. The PBFFS_SA heuristic averaged 12.1% from the lower bound, 4.5% from the best solution found, and achieved an 11% improvement over the best simple heuristic solution. By adding simulated annealing to PBFFS, the solution was improved by 6%.

For the 2-1-2-1 machine scheduling environment the RBFFS heuristic averaged 25.7% from the lower bound, 12% from the best solution found, and achieved a 1.7% improvement over the best simple heuristic solution. The RBFFS_SA heuristic averaged 12.3% from the lower bound, 0.1 % from the best solution found, and achieved a 13.8% improvement over the best simple heuristic solution. Also by adding simulated annealing to RBFFS, the solution improved by 11.9%. The PBFFS heuristic averaged 22.1% from the lower bound, 8.8% from the best solution found, and achieved a 4.7% improvement over the best simple heuristic solution. The PBFFS_SA heuristic averaged 15.3% from the lower bound, 2.8% from the best solution found, and achieved a 10.8% improvement over the best simple heuristic solution. By adding simulated annealing to PBFFS, the solution was improved by 5.8%.

For the 3-3-2-2-1 machine scheduling environment the RBFFS heuristic averaged 46% from the lower bound, 12.8% from the best solution found, and performed 3.6% worse than the best simple heuristic solution. The RBFFS_SA heuristic averaged 32.4% from the lower bound, 2.3 % from the best solution found, and 6.3% from the best simple heuristic solution. Also by adding simulated annealing to RBFFS, the solution improved by 10.3%. The PBFFS heuristic averaged 36.2% from the lower bound, 5.2% from the best solution found, and achieved a 3.3% improvement over the best simple heuristic solution. The PBFFS_SA heuristic averaged 29.5% from the lower bound, equaled the best solution found, and achieved an 8.7% improvement over the best simple heuristic solution. By adding simulated annealing to PBFFS, the solution was improved by 5.2%.

Table 8-2 Experimentation Results for the 3-1-2 Environment

Configuration 3-1-2 Buffer capacity =3

		SH1			SH2			RBFFS		
	LB	Sol	Ratio to LB	Ratio to BF	Sol	Ratio to LB	Ratio to BF	Sol	Ratio to LB	Ratio to BF
Data set 1	205	275	1.3415	1.2387	265	1.2927	1.1937	266	1.2976	1.1982
Data set 2	228	285	1.25	1.1492	281	1.2325	1.1331	270	1.1842	1.0887
Data set 3	219	269	1.2283	1.1798	273	1.2466	1.1974	260	1.1872	1.1404
Data set 4	214	256	1.1963	1.1429	268	1.2523	1.1964	264	1.2336	1.1786
Data set 5	246	305	1.2398	1.1466	303	1.2317	1.1391	298	1.2114	1.1203
Data set 6	231	291	1.2597	1.1878	293	1.2684	1.1959	284	1.2294	1.1592
Data set 7	224	280	1.25	1.1864	274	1.2232	1.161	268	1.1964	1.1356
Data set 8	230	299	1.3	1.1818	290	1.2609	1.1462	303	1.3174	1.1976
Data set 9	258	322	1.2481	1.1709	324	1.2558	1.1782	301	1.1667	1.0945
Data set 10	199	253	1.2714	1.15	255	1.2814	1.1591	257	1.2915	1.1682
Average	225.4	283.5	1.259	1.173	282.6	1.255	1.170	277.1	1.232	1.148

		RBFFS_SA			PBFFS			PBFFS_SA		
	LB	Sol	Ratio to LB	Ratio to BF	Sol	Ratio to LB	Ratio to BF	Sol	Ratio to LB	Ratio to BF
Data set 1	205	222	1.0829	1	245	1.1951	1.1036	228	1.1122	1.027
Data set 2	228	248	1.0877	1	266	1.1667	1.0726	254	1.114	1.0242
Data set 3	219	228	1.0411	1	255	1.1644	1.1184	237	1.0822	1.0395
Data set 4	214	224	1.0467	1	254	1.1869	1.1339	239	1.1168	1.067
Data set 5	246	266	1.0813	1	292	1.187	1.0977	280	1.1382	1.0526
Data set 6	231	245	1.0606	1	279	1.2078	1.1388	261	1.1299	1.0653
Data set 7	224	236	1.0536	1	263	1.1741	1.1144	246	1.0982	1.0424
Data set 8	230	253	1.1	1	280	1.2174	1.1067	268	1.1652	1.0593
Data set 9	258	275	1.0659	1	303	1.1744	1.1018	287	1.1124	1.0436
Data set 10	199	220	1.1055	1	241	1.2111	1.0955	227	1.1407	1.0318
Average	225.4	241.7	1.073	1	267.8	1.188	1.108	252.7	1.121	1.045

Table 8-3 Experimentation Results for the 2-1-2-1 Environment

Configuration 2-1-2-1 Buffer capacity =3

		SH1			SH2			RBFFS		
	LB	Sol	Ratio to LB	Ratio to BF	Sol	Ratio to LB	Ratio to BF	Sol	Ratio to LB	Ratio to BF
Data set 1	206	280	1.3592	1.1915	277	1.3447	1.1787	268	1.301	1.1404
Data set 2	237	310	1.308	1.2016	298	1.2574	1.155	284	1.1983	1.1008
Data set 3	225	294	1.3067	1.1713	296	1.3156	1.1793	289	1.2844	1.1514
Data set 4	214	277	1.2944	1.1688	278	1.2991	1.173	279	1.3037	1.1772
Data set 5	248	312	1.2581	1.1143	319	1.2863	1.1393	315	1.2702	1.125
Data set 6	234	307	1.312	1.1585	294	1.2564	1.1094	291	1.2436	1.0981
Data set 7	224	297	1.3259	1.1647	288	1.2857	1.1294	275	1.2277	1.0784
Data set 8	239	304	1.272	1.1259	307	1.2845	1.137	295	1.2343	1.0926
Data set 9	262	327	1.2481	1.1085	322	1.229	1.0915	323	1.2328	1.0949
Data set 10	216	276	1.2778	1.15	285	1.3194	1.1875	275	1.2731	1.1458
Average	230.5	298.4	1.296	1.156	296.4	1.288	1.148	289.4	1.257	1.120

		RBFFS_SA			PBFFS			PBFFS_SA		
	LB	Sol	Ratio to LB	Ratio to BF	Sol	Ratio to LB	Ratio to BF	Sol	Ratio to LB	Ratio to BF
Data set 1	206	235	1.1408	1	256	1.2427	1.0894	246	1.1942	1.0468
Data set 2	237	258	1.0886	1	280	1.1814	1.0853	267	1.1266	1.0349
Data set 3	225	251	1.1156	1	280	1.2444	1.1155	260	1.1556	1.0359
Data set 4	214	237	1.1075	1	267	1.2477	1.1266	245	1.1449	1.0338
Data set 5	248	280	1.129	1	302	1.2177	1.0786	290	1.1694	1.0357
Data set 6	234	265	1.1325	1	282	1.2051	1.0642	270	1.1538	1.0189
Data set 7	224	255	1.1384	1	276	1.2321	1.0824	259	1.1563	1.0157
Data set 8	239	270	1.1297	1	294	1.2301	1.0889	278	1.1632	1.0296
Data set 9	262	299	1.1412	1.0136	306	1.1679	1.0373	295	1.126	1
Data set 10	216	240	1.1111	1	267	1.2361	1.1125	247	1.1435	1.0292
Average	230.5	259	1.123	1.001	281	1.221	1.088	265.7	1.153	1.028

Table 8-4 Experimentation Results for the 3-3-2-2-1 Environment

Configuration 3-3-2-2-1 Buffer capacity = 4

		SH1			SH2			RBFFS		
	LB	Sol	Ratio to LB	Ratio to BF	Sol	Ratio to LB	Ratio to BF	Sol	Ratio to LB	Ratio to BF
Data set 1	214	313	1.4626	1.0982	315	1.472	1.1053	320	1.4953	1.1228
Data set 2	243	334	1.3745	1.0503	348	1.4321	1.0943	351	1.4444	1.1038
Data set 3	226	331	1.4646	1.1336	333	1.4735	1.1404	338	1.4956	1.1575
Data set 4	222	305	1.3739	1.0517	319	1.4369	1.1	329	1.482	1.1345
Data set 5	264	363	1.375	1.0804	377	1.428	1.122	376	1.4242	1.119
Data set 6	247	348	1.4089	1.0943	353	1.4291	1.1101	361	1.4615	1.1352
Data set 7	239	329	1.3766	1.0894	352	1.4728	1.1656	346	1.4477	1.1457
Data set 8	243	355	1.4609	1.0957	355	1.4609	1.0957	365	1.5021	1.1265
Data set 9	268	378	1.4104	1.0957	379	1.4142	1.0986	384	1.4328	1.113
Data set 10	227	311	1.37	1.0799	339	1.4934	1.1771	322	1.4185	1.1181
Average	239.3	336.7	1.408	1.087	347	1.451	1.121	349.2	1.460	1.128

		RBFFS_SA			PBFFS			PBFFS_SA		
	LB	Sol	Ratio to LB	Ratio to BF	Sol	Ratio to LB	Ratio to BF	Sol	Ratio to LB	Ratio to BF
Data set 1	214	287	1.3411	1.007	297	1.3879	1.0421	285	1.3318	1
Data set 2	243	324	1.3333	1.0189	328	1.3498	1.0314	318	1.3086	1
Data set 3	226	297	1.3142	1.0171	326	1.4425	1.1164	292	1.292	1
Data set 4	222	294	1.3243	1.0138	296	1.3333	1.0207	290	1.3063	1
Data set 5	264	346	1.3106	1.0298	358	1.3561	1.0655	336	1.2727	1
Data set 6	247	328	1.3279	1.0314	330	1.336	1.0377	318	1.2874	1
Data set 7	239	312	1.3054	1.0331	323	1.3515	1.0695	302	1.2636	1
Data set 8	243	330	1.358	1.0185	342	1.4074	1.0556	324	1.3333	1
Data set 9	268	356	1.3284	1.0319	356	1.3284	1.0319	345	1.2873	1
Data set 10	227	295	1.2996	1.0243	302	1.3304	1.0486	288	1.2687	1
Average	239.3	316.9	1.324	1.023	325.8	1.362	1.052	309.8	1.295	1.000

9 Conclusion

9.1 *Overview of Research*

The objective of this research was to minimize the makepsan in a flexible flowshop with sequence dependent setup times, uniform machines, and limited buffers. At the time of this research, no attempt to solve this problem could be found in the literature. The goals of this research were to:

1. Develop a mathematical model of the problem.
2. Develop two construction heuristics to construct initial feasible solutions to the problem. Then use a simulated annealing algorithm to improve the initial solutions.
3. Develop a lower bound to test the solution quality.

This problem is extremely complex and it is not practical to solve optimally. A mathematical model for the problem was created, and used to solve small problems optimally. Even very small problems, such as a three stage 5 job problem, took around an hour to solve.

Two construction heuristics, RBFFS and PBFFS were developed based on a flowshop heuristic given by Nawaz et. al (1983). The first heuristic is route based, and the second is job priority based. The heuristics give a higher scheduling priority to jobs that have larger processing times. They use adjacent pariwise exchange and add jobs one at a time until all jobs are scheduled. Next, a simulated annealing heuristic is used to improve the final solutions obtained from the construction heuristics. The construction and SA heuristics use a solution space of $n!$, where n is the total number of jobs. This reduces the solution space but allows good solutions to be found in a reasonable amount of time.

Two methods were used to test the proposed algorithms performance. A lower bound was developed that focused on the bottleneck stage and then scheduled jobs before and after that stage. Also two simple heuristics were used to solve the problem. One simple heuristic, SH1, assigned jobs priority on a first come first serve basis. The second simple heuristic, SH2, assigned jobs priorities based on total processing time, with the higher total processing time jobs receiving higher priority. The simple heuristics provide feasible solutions that are used for comparison.

In all but one case, both construction heuristics obtained a better solution than the best simple heuristic. The simulated annealing heuristic was able to improve on the solutions obtained with the construction heuristics. When comparing to the lower bound, both construction algorithms and simulated annealing heuristics performed well for the smaller systems. For the machine configurations of 3-1-2, 2-1-2-1 and 3-3-2-2-1, the RBFFS_SA heuristic was 7.3%, 12.3%, and 32.4.5% respectively higher than the lower bound; while the PBFFS_SA heuristic was 12.1%, 15.3%, and 29.5% higher than the lower bound. The increase in the gap between the heuristic solution and the lower bound for the larger stage environments is most likely due to deficiencies in the lower bound and not due to poor heuristic performance. The presented lower bound does not account for machine blocking, and as the number of stages increase, the chances of a machine being blocked also increase. The lower bound presented was compared to a lower bound calculated by relaxing the binary and integer variables. The LB presented outperformed the relaxed version.

The RBFFS_SA heuristic performed the best for the small and medium sized problems. The PBFFS_SA heuristic out performed the RBFFS for the larger problem.

The simulated annealing algorithm had a much larger affect on the RBFFS's solution; it was improved between 10.3% and 14.8%, while the SA improved the PBFFS only 5.2% to 6%. Limiting the solution space to permutation schedules may have produced this difference. The RBFFS is much more delicate and dependant on the permutation solution. The RBFFS_SA heuristic has a short run time, but it does take extra time to develop a good routing system. Both heuristics were able to provide good solutions to the problem in a reasonable amount of time.

9.2 *Contribution to Literature*:

The contributions made to the literature by this research are as follows:

- A mathematical model was developed for the flexible flowshop with sequence dependent setup times, uniform machines, and limited buffers.
- Development of two construction heuristics combined with an improvement heuristic to give a good quick solution to the problem.
- Development of a simulated annealing heuristic to improve the solution of the construction heuristics.
- Development of a lower bound for the problem.
- Laid ground work for future research on this type of problem.

9.3 *Recommendations for Future Research:*

The work performed in this research opens the door for more investigation of the flexible flowshop with sequence dependent setup times, uniform machines, and limited buffers problem. Suggestion for future research include:

- Develop a tighter lower bound to test the heuristic approaches to this problem by adding a factor to account for machine blocking.
- Look at other meta-heuristics to improve the solution obtained from the construction algorithm.
- Experiment with different job priority systems for the problem.
- Experiment with other machine configurations and/or different systems of data generation.
- Add job ready times and/or due dates.
- Before scheduling a job check to see which machine would finish processing it faster. The total time may be less if it waits for a machine to open up, instead of being processed on the first available machine.

10 References

Acero-Dominguez, M. and Paternina-Arboleda, C. (2004). Scheduling Jobs on a K-Stage Flexible Flowshop Using a TOC-Based (Bottleneck) Procedure. *Proceedings of the 2004 Systems and Information Engineering Design Symposium*, 295-297.

Adams, J., Balas, E., and Zawack, D. (1988). The Shifting Bottleneck Procedure For Job Shop Scheduling. *Management Science*, 34 (3), 391-401.

Agnetis, A., Rossi, F., and Gristina, G. (1998). An Exact Algorithm for the Batch Sequence Problem in a Two-Machine Flowshop with Limited Buffer. *Naval Research Logistics*, 45, 141-164.

Allahverdi, A. (2000). Minimizing mean flow time in a two-machine flowshop with sequence-independent setup times. *Computers and Operations Research,* 111-127.

Allahverdi, A. and Aldowaisan, T. (2004). No-wait flowshops with bicriteria of makespan and maximum lateness. *European Journal of Operational Research,* 152, 132-147.

Allahverdi, A., Gupta, J.N.D., and Aldowaisan, T. (1999). A review of scheduling research involving setup considerations. *Omega International Journal of Management Science,* 27, 219-239.

Arthanari, T., and Ramamurthy, K. (1971). An Extension of Two Machines Sequencing Problem. *Opsearch,* 8, 10-22.

Azizoglu, M., Cakmak, E., and Suna, K. (2001). A flexible flowshop problem with total flow time minimization. *European Journal of Operational Research,* 132, 528-538.

Baker, K. (1974). *Introduction to Sequencing & Scheduling.* United States: John Wiley & Sons.

Barman, S. (1998). The impact of priority rule combinations on lateness and tardiness. *IIE Transactions*, 30, 495-504.

Brah, S. and Hunsucker, J. (1991). Branch and bound algorithm for the flowshop with multiple processors. *European Journal of Operational Research*, 51, 88-99.

Brah, S. and Loo, L. (1999). Heuristics for scheduling in a flowshop with multiple processors. *European Journal of Operational Research*, 113, 113-122.

Buzacott, J. and Yao, D. (1986). Flexible Manufacturing Systems: A Review of Analytical Models. *Management Science*, 32 (7), 890-905.

Campbell, H., Dudek, R., and Smith, M. (1970). A Heuristic Algorithm for the *n* Job, *m* Machine Sequencing Problem. *Management Science* , 16(10), B-630-37.

Carlier, Jacques (1982). The one-machine sequencing problem. *European Journal of Operational Research ,* 11, 42-47.

Carlier, J. and Rebai, I. (1996). Two branch and bound algorithms for the permutation flowshop problem. *European Journal of Operational Research*, 90, 238-251.

Chang, S. (1994). Scheduling Flexible Flowshops with No Setup Effects. *IEEE Transactions on Robotics and Automation,* 10(2), 112-122.

Chen, Bo., Glass, C., Potts, C., and Strusevich V. (1996). A New Heuristic for Three-Machine Flowshop Scheduling. *Operations Research*, 44(6), 891-898.

Chen, Bo (1995). Analysis of Classes of Heuristics for Scheduling a Two-Stage Flowshop with Parallel Machines at One Stage. *Journal of the Operational Research Society*, 46, 234-244.

Chen, Bo (1994). Scheduling multiprocessor flowshops. *Advances in optimization and approximation,* 1-8.

Cheng, J., Karuno, Y. and Kise, H. (2001). A shifting bottleneck approach for a parallel-machine flowshop scheduling problem. *Journal of the Operations Research Society of Japan,* 44(2), 140-156.

Cheng, T., and Sin, C. (1990). A state-of-the-art review of parallel-machine scheduling research. *European Journal of Operational Research*, 47, 271-292.

Cliffe, R., and MacManus, B. (1980). An approach to optimization with heuristic methods scheduling. *International Journal of Production Research*, 18(4), pp 479-490.

Co, Henry (1992). Streamlining material flow in flexible manufacturing systems: a lesson in simplicity. *International Journal of Production Research*, 32(7), 1483-1499.

Conway, R., Maxwell, W., & Miller, L. (1967). *Theory of Scheduling.* New York: Dover Publications.

Corwin, B. and Esogbue, A. (1974). Two Machine Flowshop Scheduling Problems with Sequence Dependent Setup Times: A Dynamic Programming Approach. *Naval Research Logistics Quarterly*, 21(3), 515-524.

Daniels, R. and Mazzola, J. (1994). Flowshop Scheduling With Resource Flexibility. *Operations Research Society of America*, 42(3), 504-522.

Dannenbring, D. (1977). An Evaluation of Flowshop Sequencing Heuristics. *Management Science* , 23(11), 1174-112

Day, J., and Hottenstein, M. (1970). Review of Sequencing Research. *Naval Research Logistics Quarterly* , 17(1), 11-39.

Ding, F., and Kittichartphayak, D. (1994). Heuristics For Scheduling Flexible Flow Lines. *Computers and Industrial Engineering* , 26(1), 27-34.

Dudek, R., Panwalkar, S., and Smith, M. (1992). The Lessons of Flowshop Scheduling Research. *Operations Research*, 40(1), 7-13.

Eglese, R. (1990). Simulated Annealing: A tool for operational research. *European Journal of Operational Research,* 46, 271-281.

Garey, M., Johnson, D., and Sethi, R. (1976). The Complexity of Flowshop and Jobshop Scheduling. *Mathematics of Operations Research,* 1(2), 117-129.

Gilmore, P., and Gomory, R. (1964). Sequencing a One State-Variable Machine a Solvable Case of the Traveling Salesman Problem. *Operations Research,* 12, 655-679.

Gonzalez, T., Ibarra, O., and Sahni, S. (1977). Bounds for LPT Schedules on Uniform Processors. *SIAM Journal on Computing*, 6(1), 155-166.

Guinet, A. and Solomon, M. (1996). Scheduling hybrid flowshops to minimize maximum tardiness or maximum completion time. *International Journal of Production Research*, 34(6), 1643-1654.

Guinet, A., Solomon, M., Kedia, P., and Dussauchy, A. (1996). A computational study of heuristics for two-stage flexible flowshops. *International Journal of Production Research,* 34(5), 1399-1415.

Gupta, J.N.D. (1972). Heuristic algorithms for multistage flowshop scheduling problem. *AIIE Transactions*, 4(1), 11-18.

Gupta, J.N.D. (1976). Optimal Flowshop Schedules with No Intermediate Storage Space. *Naval Research Logistics Quarterly,* 23, 235-243.

Gupta, J.N.D. (1986). Flowshop Schedules with Sequence Dependent Setup Times. *Journal of the Operations Research Society of Japan*, 29(3), 206-219.

Gupta, J.N.D. (1988). Two-Stage, Hybrid Flowshop Scheduling Problem. *Operational Research Society,* 39(4), 359-364.

Gupta, J.N.D., Hariri, A., and Potts, C. (1997). Scheduling a two-stage hybrid flowshop with parallel machines at the first stage. *Annals of Operations Research*, 69, 171-191.

Gupta, JND and Ruiz-Torres, A. (2000). Minimizing makespan subject to minimum total flow-time on identical parallel machines. *European Journal of Operational Research,* 125, 370-380.

Gupta, J.N.D., and Szwarc, W. (1987). A Flow-Shop Problem with Sequence-Dependent Additive Setup Times. *Naval Research Logistics*, 34, 619-627.

Gupta, J.N.D., and Tunc, E. (1994). Scheduling a two-stage hybrid flowshop with separable setup and removal times. *European Journal of Operational Research*, 77, 415-428.

Gupta, J.N.D., and Tunc, E. (1991). Schedules for a two-stage hybrid flowshop with parallel machines at the second stage. *International Journal of Production Research*, 29(7), 1489-1502.

Gupta, Sushil (1982). N jobs and M machines job-shop problems with sequence-dependent setup times. *International Journal of Production Research*, 20, (5), 643-656.

Haouari, M., and M'Hallah, R. (1997). Heuristic algorithms for the two-stage hybrid flowshop problem. *Operations Research Letters*, 21, 43-53.

Hejeck, B. (1988). Cooling Schedules for optimal annealing. *Mathematics of Operations Research,* 13(2), 311-329.

Ho, J. (1995). Flowshop sequencing with mean flowtime objective. *European Journal of Operational Research,* 81, 571-578.

Ho, J., and Chang, Y. (1991). A new heuristic for the n-job, M-machine flow-shop problem. *European Journal of Operational Research*, 52, 194-202.

Hoogeveen, J., Lenstra, J, and Veltman, B. (1996). Preemptive scheduling in a two-stage multiprocessor flowshop is NP-hard. *European Journal of Operational Research*, 89, 172-175.

Huang, W. (1998). A Two-Stage Hybrid Flowshop with Uniform Machines and Setup Times. *Mathematical and Computer Modelling*, 27(2), 27-45.

Hundal, T., and Rajgopal, J. (1988). An extension of Palmer's heuristic for the flowshop scheduling problem. *International Journal of Production Research*, 26(6), 1119-1124.

Ishibuchi, H., Misaki, S., and Tanaka, H. (1995). Modified simulated annealing algorithms for the flowshop sequencing problem. *European Journal of Operational Research*, 81, 388-398.

Ignall, E. and Schrage, L. (1965). Application of the Branch-and-Bound Technique to Some Flow-Shop Scheduling Problems. *Operations Research,* 13(3)

Jayamohan, M. and Rajendran, C. (2000). A comparative analysis of two different approaches to scheduling in flexible flowshops. *Production Planning and Control*, 11(6), 572-580.

Johnson, S.M. (1954). Optimal Two-and Three-Stage Production Schedules with Setup Times included. *Naval Research Logistics Quarterly*, 1(1), 61-67.

Kadipasaoglu, S., Xiang, W., and Khumawala, B. (1997). A comparison of sequencing rules in static and dynamic hybrid flow systems. *International Journal of Production Research* , 35(5), 1359-1384.

Khmelnitsky, E., Kogan, K., and Maimon, O. (1997). Maximum principle-based methods for production scheduling with partially sequence-dependent setups. *International Journal of Production Research*, 35(10), 2701-2712.

King, J. and Spachis, A. (1980). Heuristic for flow-shop scheduling. *International Journal of Production Research,* 18(3), 345-357.

Kirkpatrick, S., Gelatt, D., and Vecchi, M. (1983). Optimization by Simulated Annealing. *Science,* 220 (4598).

Kochhar, S. and Morris, R. (1987). Heuristic Methods for Flexible Flow Line Scheduling. *Journal of Manufacturing Systems*, 6(4), 299-314.

Kochhar, S., Morris, R., and Wong, W. (1988). The Local Search Approach to Flexible Flow Line Scheduling. *Engineering Costs and Production Economics*, 14, 25-37.

Koulamas, C. (1998). A new constructive heuristic for the flowshop scheduling problem. *European Journal of Operational Research*, 105, 66-71.

Koulamas, C., and Kyparisis, G. (2000). Asymptotically Optimal Linear Time Algorithms for Two-Stage and Three-Stage Flexible Flowshops. *Naval Research Logistics* , 47, 259-268.

Kouvelis, P., and Chiang, W. (1992). A simulated annealing procedure for single row layout problems in flexible manufacturing systems. *International Journal of Production Research,* 30(4), 717-732.

Lawler, E. et al. (1993). Sequencing and Scheduling: Algorithms and Complexity. *Handbook in Operations Research and Management Science*, Edited by Rinnooy Kan and Zipkin, Vol. 4: Logistics of Production and Inventory, pp 445-522. North-Holland, New York.

Lee, C., Cheng, T., and Lin, B. (1993). Minimizing the Makespan in the 3-Machine Assembly-type Flowshop Scheduling Problem. *Management Science*, 39(5), 616-625.

Lee, C. and Vairaktarakis, G. (1994). Minimizing makespan in hybrid flowshops. *Operations Research Letters*, 16, 149-158.

Lee, C. and Vairaktarakis, G. (1998). Performance Comparison of Some Classes of Flexible Flowshops and Job Shops. *The International Journal of Flexible Manufacturing Systems,* 10, 379-405.

Lee, G., Kin, Y., and Choi, S. (2004). Bottleneck-focused scheduling for a hybrid flowshop. *International Journal of Production Research,* 42(1), 165-181.

Lee, Y. and Pinedo, M. (1997). Scheduling jobs on parallel machines with sequence-dependent setup times. *European Journal of Operational Research*, 100, 464-474.

Leon, V., and Ramamoorthy, B. (1997). An adaptable problem-space-based search method for flexible flow line scheduling. *IIE Transactions*, 29, 115-125.

Li, Shanling (1997). A hybrid two-stage flowshop with part family, batch production major and minor setups. *European Journal of Operational Research*, 102, 142-156.

Linn, R. and Zhang, W. (1999). Hybrid Flowshop Scheduling: A Survey. *Computers & Industrial Engineering,* 37, 57-61.

Liu, C., and Chang, S. (2000). Scheduling Flexible Flowshops with Sequence-Dependent Setup Effects. *IEEE Transactions on Robotics and Automation*, 16(4), 408-419.

Loukil, T., Teghem, J., and Tuyttens, D. (2005). Solving multi-objective production scheduling problems using meta-heuristics. *European Journal of Operational Research*, 161, 42-61.

Low, C. (2005). Simulated annealing heuristic for flowshop scheduling problems with unrelated parallel machines. *Computers & Operations Research,* 32, 2013-2025.

Low, C., Yeh, J., and Huang, K. (2004). A robust simulated annealing heuristic for flowshop scheduling problems. *International Journal of Advanced Manufacturing Technology*, 23, 762-767.

Moursli, O. (1995). Branch and Bound Lower Bounds for the Hybrid Flowshop. *Intelligent Manufacturing Systems* , 4th IFAC Workshop, 31-36.

Narasimham, S. and Mangiameli, P. (1987). A Comparison of Sequencing Rules for a Two-Stage Hybrid Flowshop. *Decision Sciences*, 18, 250-265.

Narasimhan, S., and Panwalker, S. (1984). Scheduling in a two-stage manufacturing process. *International Journal of Production Research*, 22(4), 555-564.

Nawaz, M., Enscore, E., and Ham, I. (1983). A Heuristic Algorithm for the m-machine n-Job Flowshop Sequencing Problem. *Omega,* 11(1) , 91-95.

Nearchou, A. (2004). A novel meta-heuristic approach for the flowshop scheduling problem. *Engineering Applications of Artificial Intelligence*, 17, 289-300.

Nearchou, A. (2004). Flow-shop sequencing using hybrid simulated annealing. *Journal of Intelligent Manufacturing,* 15, 317-328.

Negenman, E. (2001). Local search algorithms for the multiprocessor flowshop scheduling problem. *European Journal of Operational Research,* 128, 147-158.

Neron, E., Baptiste, P., and Gupta, JND. (2001). Solving hybrid flowshop problem using energetic reasoning and global operations. *Omega,* 29, 501-511.

Norman, B. (1999). Scheduling flowshops with finite buffers and sequence-dependent setup times. *Computers & Industrial Engineering,* 36, 163-177.

Nowicki, E., and Smutnicki, C. (1993). New results in the worst-case analysis for flow-shop scheduling. *Discrete Applied Mathematics*, 46, 21-41.

Nowicki, E., and Smutnicki, C. (1989). Worst-Case Analysis of an Approximation Algorithm for Flow-Shop Scheduling. *Operations Research Letters*, 8(3), 171-177.

Ogbu, F. and Smith, D. (1990). Simulated Annealing for the Permutation Flowshop Problem. *Omega*, 19(1), 64-67.

Ogbu, F. and Smith, D. (1990). The application of the simulated annealing algorithm to the solution of the n/mCmax flowshop problem. *Computers and Operations Research,* 17(3), 243-253.

Oguz, C., Lin, B., and Cheng, T. (1997). Two-stage flowshop scheduling with a common second-stage machine. *Computers and Operations Research*, 24(12), 1169-1174.

Oi, Mei. (1996). Scheduling two-stage production lines with multiple machines. *Production Planning & Control*, 7(4), 418-429

Osman, I., and Potts, C. (1989). Simulated Annealing for Permutation Flow-Shop Scheduling. *OMEGA International Journal of Management Science*, 17(6), 551-557.

Palmer, D. (1965). Sequencing Jobs through a Multi-Stage Process in the Minimum Total Time-A Quick Method of Obtaining a Near Optimum. *Operations Research Quarterly*, 16, 101-107.

Parthasarathy, S. and Rajendran, C. (1997). A simulated annealing heuristic for scheduling to minimize mean weighted tardiness in a flowshop with sequence-dependent setup times of jobs- a case study. *Production Planning and Control*, 8(5). 475-483.

Peeters, P., Brussel, H., Valckenaers, P., Wyns, Jo., Bongaerts, L., Kollingbaum, M., and Heikkila, T. (2001). Pheromone based emergent shop floor control system for flexible flowshops. *Artificial Intelligence in Engineering*, 15, 343-352.

Phadnis, S., Brevick, J., and Irani, S. (2003, March). Development of a New Heuristic For Scheduling Flow-Shops with Parallel Machines By Prioritizing Bottleneck Stages. *Transactions of the Society for Design and Process Science*, 7(1), 87-97.

Pinedo, M., (2002). *Scheduling Theory, Algorithms, and Systems*. New Jersey: Prentice Hall.

Portman, M., Vignier, A., Dardilhac, D., and Dezalay, D. (1998). Branch and bound crossed with GA to solve hybrid flowshops. *European Journal of Operational Research*, 107, 389-400.

Raban, S., and Nagel, R. (1991). Constraint-based control of flexible flow lines. *International Journal of Production Research*, 29(10), 1941-1951.

Rajendran, C., and Chaudhuri, D. (1992). Scheduling in n-job, m-stage flowshop with parallel processors to minimize makespan. *International Journal of Production Economics*, 27, 137-143.

Randhawa, S., and Kuo, C. (1997). Evaluating scheduling heuristics for non-identical parallel processors. *International Journal of Production Research,* 35(4), 969-981.

Randhawa, S,. and Smith, T. (1995). An experimental investigation of scheduling non-identical, parallel processors with sequence-dependent setup times and due dates. *International Journal of Production Research*, 33(1), 59-69.

Riane, F., Artiba, A., and Elmaghraby, S. (1998). A hybrid three-stage flowshop problem: Efficient heuristics to minimize makespan. *European Journal of Operational Research*, 109, 321-329.

Riane, F., Raczy, C., and Artiba, A. (1999). Hybrid Auto-adaptable Simulated Annealing based Heuristic. *Computers & Industrial Engineering,* 37, 277-280.

Riane, F.k, Artiba, A., and Elamaghraby, S. (2002). Sequencing a hybrid two-stage flowshop with dedicated machines. *International Journal of production Research*, 40(17) , 4353-4380.

Rios-Mecado, R. and Bard, J. (1999). A branch-and-bound algorithm for permutation flowshops with sequence-dependent setup times. *IIE Transactions*, 31, 721-731.

Rios-Mercado, R. and Bard, J. (1999). An Enhanced TSP-Based Heuristic for Makespan Minimization in a Flowshop with Setup Times. *Journal of Heuristics*, 5, 53-70.

Rios-Mercado, R. and Bard, J. (2003). The Flowshop Scheduling Polyhedron with Setup Times. *Journal of Combinatorial Optimization,* 7, 291-318.

Rios-Mercado, R. and Bard, J. (1998). Heuristics for the flow line problem with setup costs. *European Journal of Operational Research*, 110, 76-98.

Sarin, S. and Lefoka, M. (1993). Scheduling Heuristic for the n-Job m-Machine Flowshop. *OMEGA International Journal of Management Science,* 21(2), 229-234.

Sawik, T. (2002). An Exact Approach for Batch Scheduling in Flexible Flow Lines with Limited Intermediate Buffers. *Mathematical and Computer Modelling*, 36, 461-471.

Sawik, T. (2000). Mixed Integer Programming for Scheduling Flexible Flow Lines with Limited Intermediate Buffers. *Mathematical and Computer Modelling*, 31, 39-52.

Sawik, T. (1995). Scheduling flexible flow lines with no in-process buffers. *International Journal of Production Research*, 33(5), 1357-1367.

Sawik, T. (1987). Multilevel Scheduling of Multistage Production with Limited In-Process Inventory. *Journal of the Operational Research Society*, 38(7), 651-664.

Sethanan, K. (2001). Scheduling Flexible Flowshops with Sequence Dependent Setup Times. *Dissertation for Doctor of Philosophy in Decision Sciences and Production Systems, West Virginia University.*

Shieh, A. (2003). A Simulated Annealing Approach for Flexible Flowshop Scheduling to Maximize Flexiblity. *Masters Thesis, for Master of Science in Industrial Engineering, West Virginia University.*

Simons, J. (1992). Heuristics in Flowshop Scheduling with Sequence Dependent Setup Times. *OMEGA International Journal of Management Science*, 20(2), 215-225.

Soewandi, H., and Elmaghraby, S. (2003). Sequencing on two-stage hybrid flowshops with uniform machines to minimize makespan. *IIE Transactions*, 35, 467-477.

Soewandi, H., and Elmaghraby, S. (2001). Sequencing three-stage flexible flowshops with identical machines to minimize makespan. *IIE Transactions*, 33, 985-993.

Srikar, B. and Ghosh, S. (1986). A MILP model for the n-job, M-stage flowshop with sequence dependent setup times. *International Journal of Production Research*, 24(6), 1459-1474.

Sriskandarajah, C., and Sethi, S. (1989). Scheduling algorithms for flexible flowshops: Worst and average case performance. *European Journal of Operational Research*, 43, 143-160.

Stafford, E., and Tseng, F. (1990). On the Srikar-Ghosh MILP model for the N x M SDST flowshop problem. *International Journal of Production Research,* 28(10), 1817-1830.

Stecke, K. and Kim, I. (1991). A flexible approach to part type selection in flexible flow systems using part mix ratios. *International Journal of Production Research,* 29(1), 53-75.

Szwarc, W. and Gupta, J. (1987). A Flow-Shop Problem with Sequence-Dependent Additive Setup Times. *Naval Research Logistics*, 34, 619-627.

Taillard, E. (1993). Benchmarks for basic scheduling problems. *European Journal of Operational Research,* 64, 278-285.

Taillard, E. (1990). Some efficient heuristic methods for the flowshop sequencing problem. *European Journal of Operational Research*, 47, 65-74.

Townsend, D. (1977). Sequencing *n* jobs on *m* machines to minimize tardiness: a branch and bound solution. *Management Science*, 23, 1016-1019.

Tsubone, H., Ohba, M., and Uetake, T. (1996). The impact of lot sizing and sequencing on manufacturing performance in a two-stage hybrid flowshop. *International Journal of Production Research,* 34(11), 3037-3053.

Turner, S. and Booth, D. (1987). Comparison of Heuristics for Flowshop Sequencing. *Omega,* 15(1), 75-85.

Uetake, T., Tsubone, H., and Ohba, M. (1995). A production scheduling system in a hybrid flowshop. *International Journal of Production Economics*, 41, 395-398.

Verma, S., and Dessouky, M. (1999). Multistage Hybrid Flowshop Scheduling With Identical Jobs and Uniform Parallel Machines. *Journal of Scheduling*, 2, 135-150.

Vignier, A., Dardilhac, D., Dezalay, D., and Proust, C. (1996). A branch and bound approach to minimize the total completion time in a k-stage hybrid flowshop.

Procedings: IEEE Conference on Emerging Technologies and Factory Automation, 215 – 220.

Wang, C., Chu, C., and Proth, J. (1997). Heuristic approaches for *n/m/F/ΣCi* scheduling problems. *European Journal of Operational Research*, 96, 636-644.

Wang, H., Jacob, V., and Rolland, E. (2003). Design of efficient hybrid neural networks for flexible flowshop scheduling. *Expert Systems*, 20(4), 208-231.

Wang, L. and Zheng, D. (2003). An Effective Hybrid Heuristic for Flowshop Scheduling. *International Journal of Manufacturing Technology,* 21, 38-44.

Weng, M. (2000). Scheduling Flow-Shops with Limited Buffer Spaces. *Proceedings of the 2000 Winter Simulation Conference,* 1359-1363.

White, C., and Wilson, R. (1977). Sequence dependent setup times and job sequencing. *International Journal of Production Research*, 15(2), 191-202.

Widmer, M. and Hertz, A. (1989). A new heuristic method for the flowshop sequencing problem. *European Journal of Operational Research*, 41, 186-193.

Wittrock, R. (1988). An Adaptable Scheduling Algorithm for Flexible Flow Lines. *Operations Research Society of America*, 36(3), 445-453

Wittrock, R. (1985). Scheduling algorithms for flexible flow lines. *IBM Journal of Research and Development,* 29(4), 401-412.

Ying, K. and Liao, C. (2004). An ant colony system for permutation flow-shop sequencing. *Computers an Operations Research*, 31, 791-801.

Zegordi, S., Itoh, K., and Enkawa, T. (1995). Minimizing makespan for flowshop scheduling by combining simulated annealing with sequencing knowledge. *European Journal of Operational Research,* 85, 515-531.

Zhou, C. and Egbelu, P. (1989). Scheduling in a manufacturing shop with sequence-dependent setup times. *Robotics and Computer-Integrated Manufacturing*, 5(1), 73-81.

Appendix A: Experimental Results

Data set 1		Ratio to LB	Stage Configuration 3-1-2		Data set 2		Ratio to LB	Stage Configuration 3-1-2	
LB	205				LB	228			
SH1	275	1.3415	Buffers size = 3		SH1	285	1.2500	Buffers size = 3	
SH2	265	1.2927			SH2	281	1.2325		
RBFFS	266	1.2976			RBFFS	270	1.1842		
Data set 1	RBFFS_SA	Ratio RBFFS_SA to LB	Ratio Best SH to RBFFS_SA	Ratio RBFFS to RBFFS_SA	Data set 2	RBFFS_SA	Ratio RBFFS_SA to LB	Ratio Best SH to RBFFS_SA	Ratio RBFFS to RBFFS_SA
Exp 1	223	1.0878	1.1883	1.1928	Exp 1	248	1.0877	1.1331	1.0887
Exp 2	229	1.1171	1.1572	1.1616	Exp 2	251	1.1009	1.1195	1.0757
Exp 3	226	1.1024	1.1726	1.1770	Exp 3	249	1.0921	1.1285	1.0843
Exp 4	222	1.0829	1.1937	1.1982	Exp 4	252	1.1053	1.1151	1.0714
Exp 5	225	1.0976	1.1778	1.1822	Exp 5	249	1.0921	1.1285	1.0843
Exp 6	226	1.1024	1.1726	1.1770	Exp 6	254	1.1140	1.1063	1.0630
Exp 7	225	1.0976	1.1778	1.1822	Exp 7	249	1.0921	1.1285	1.0843
Exp 8	230	1.1220	1.1522	1.1565	Exp 8	248	1.0877	1.1331	1.0887
Exp 9	222	1.0829	1.1937	1.1982	Exp 9	253	1.1096	1.1107	1.0672
Exp 10	231	1.1268	1.1472	1.1515	Exp 10	252	1.1053	1.1151	1.0714

Data set 3		Ratio to LB	Stage Configuration 3-1-2		Data set 4		Ratio to LB	Stage Configuration 3-1-2	
LB	219				LB	214			
SH1	269	1.2283	Buffers size = 3		SH1	256	1.1963	Buffers size = 3	
SH2	273	1.2466			SH2	268	1.2523		
RBFFS	260	1.1872			RBFFS	264	1.2336		
Data set 3	RBFFS_SA	Ratio RBFFS_SA to LB	Ratio Best SH to RBFFS_SA	Ratio RBFFS to RBFFS_SA	Data set 4	RBFFS_SA	Ratio RBFFS_SA to LB	Ratio Best SH to RBFFS_SA	Ratio RBFFS to RBFFS_SA
Exp 1	236	1.0776	1.1398	1.1017	Exp 1	228	1.0654	1.1228	1.1579
Exp 2	231	1.0548	1.1645	1.1255	Exp 2	229	1.0701	1.1179	1.1528
Exp 3	231	1.0548	1.1645	1.1255	Exp 3	224	1.0467	1.1429	1.1786
Exp 4	232	1.0594	1.1595	1.1207	Exp 4	232	1.0841	1.1034	1.1379
Exp 5	228	1.0411	1.1798	1.1404	Exp 5	231	1.0794	1.1082	1.1429
Exp 6	229	1.0457	1.1747	1.1354	Exp 6	230	1.0748	1.1130	1.1478
Exp 7	236	1.0776	1.1398	1.1017	Exp 7	230	1.0748	1.1130	1.1478
Exp 8	232	1.0594	1.1595	1.1207	Exp 8	228	1.0654	1.1228	1.1579
Exp 9	234	1.0685	1.1496	1.1111	Exp 9	226	1.0561	1.1327	1.1681
Exp 10	234	1.0685	1.1496	1.1111	Exp 10	232	1.0841	1.1034	1.1379

Data set 5		Ratio to LB	Stage Configuration 3-1-2		Data set 6		Ratio to LB	Stage Configuration 3-1-2	
LB	246				LB	231			
SH1	305	1.2398	Buffers size = 3		SH1	291	1.2597	Buffers size = 3	
SH2	303	1.2317			SH2	293	1.2684		
RBFFS	298	1.2114			RBFFS	284	1.2294		
Data set 5	RBFFS_SA	Ratio RBFFS_SA to LB	Ratio Best SH to RBFFS_SA	Ratio RBFFS to RBFFS_SA	Data set 6	RBFFS_SA	Ratio RBFFS_SA to LB	Ratio Best SH to RBFFS_SA	Ratio RBFFS to RBFFS_SA
Exp 1	272	1.1057	1.1140	1.0956	Exp 1	246	1.0649	1.1829	1.1545
Exp 2	269	1.0935	1.1264	1.1078	Exp 2	245	1.0606	1.1878	1.1592
Exp 3	267	1.0854	1.1348	1.1161	Exp 3	246	1.0649	1.1829	1.1545
Exp 4	270	1.0976	1.1222	1.1037	Exp 4	254	1.0996	1.1457	1.1181
Exp 5	267	1.0854	1.1348	1.1161	Exp 5	249	1.0779	1.1687	1.1406
Exp 6	271	1.1016	1.1181	1.0996	Exp 6	255	1.1039	1.1412	1.1137
Exp 7	266	1.0813	1.1391	1.1203	Exp 7	252	1.0909	1.1548	1.1270
Exp 8	270	1.0976	1.1222	1.1037	Exp 8	252	1.0909	1.1548	1.1270
Exp 9	272	1.1057	1.1140	1.0956	Exp 9	247	1.0693	1.1781	1.1498
Exp 10	271	1.1016	1.1181	1.0996	Exp 10	252	1.0909	1.1548	1.1270

Data set 7		Ratio to LB	Stage Configuration 3-1-2		Data set 8		Ratio to LB	Stage Configuration 3-1-2	
LB	224				LB	230			
SH1	280	1.2500	Buffers size = 3		SH1	299	1.3000	Buffers size = 3	
SH2	274	1.2232			SH2	290	1.2609		
RBFFS	268	1.1964			RBFFS	303	1.3174		
Data set 7	RBFFS_SA	Ratio RBFFS_SA to LB	Ratio Best SH to RBFFS_SA	Ratio RBFFS to RBFFS_SA	Data set 8	RBFFS_SA	Ratio RBFFS_SA to LB	Ratio Best SH to RBFFS_SA	Ratio RBFFS to RBFFS_SA
Exp 1	239	1.0670	1.1464	1.1213	Exp 1	256	1.1130	1.1328	1.1836
Exp 2	241	1.0759	1.1369	1.1120	Exp 2	254	1.1043	1.1417	1.1929
Exp 3	236	1.0536	1.1610	1.1356	Exp 3	253	1.1000	1.1462	1.1976
Exp 4	237	1.0580	1.1561	1.1308	Exp 4	259	1.1261	1.1197	1.1699
Exp 5	236	1.0536	1.1610	1.1356	Exp 5	258	1.1217	1.1240	1.1744
Exp 6	240	1.0714	1.1417	1.1167	Exp 6	256	1.1130	1.1328	1.1836
Exp 7	240	1.0714	1.1417	1.1167	Exp 7	259	1.1261	1.1197	1.1699
Exp 8	241	1.0759	1.1369	1.1120	Exp 8	257	1.1174	1.1284	1.1790
Exp 9	239	1.0670	1.1464	1.1213	Exp 9	256	1.1130	1.1328	1.1836
Exp 10	237	1.0580	1.1561	1.1308	Exp 10	257	1.1174	1.1284	1.1790

Data set 9		Ratio to LB	Stage Configuration 3-1-2		Data set 10		Ratio to LB	Stage Configuration 3-1-2	
LB	258				LB	199			
SH1	322	1.2481	Buffers size = 3		SH1	253	1.2714	Buffers size = 3	
SH2	324	1.2558			SH2	255	1.2814		
RBFFS	301	1.1667			RBFFS	257	1.2915		
Data set 9	RBFFS_SA	Ratio RBFFS_SA to LB	Ratio Best SH to RBFFS_SA	Ratio RBFFS to RBFFS_SA	Data set 10	RBFFS_SA	Ratio RBFFS_SA to LB	Ratio Best SH to RBFFS_SA	Ratio RBFFS to RBFFS_SA
Exp 1	275	1.0659	1.1709	1.0945	Exp 1	225	1.1307	1.1244	1.1422
Exp 2	280	1.0853	1.1500	1.0750	Exp 2	220	1.1055	1.1500	1.1682
Exp 3	278	1.0775	1.1583	1.0827	Exp 3	222	1.1156	1.1396	1.1577
Exp 4	277	1.0736	1.1625	1.0866	Exp 4	226	1.1357	1.1195	1.1372
Exp 5	276	1.0698	1.1667	1.0906	Exp 5	223	1.1206	1.1345	1.1525
Exp 6	282	1.0930	1.1418	1.0674	Exp 6	224	1.1256	1.1295	1.1473
Exp 7	277	1.0736	1.1625	1.0866	Exp 7	223	1.1206	1.1345	1.1525
Exp 8	276	1.0698	1.1667	1.0906	Exp 8	221	1.1106	1.1448	1.1629
Exp 9	276	1.0698	1.1667	1.0906	Exp 9	223	1.1206	1.1345	1.1525
Exp 10	277	1.0736	1.1625	1.0866	Exp 10	222	1.1156	1.1396	1.1577

Data set 1		Ratio to LB	Stage Configuration 3-1-2		Data set 2		Ratio to LB	Stage Configuration 3-1-2	
LB	205				LB	228			
SH1	275	1.3415	Buffers size = 3		SH1	285	1.2500	Buffers size = 3	
SH2	265	1.2927			SH2	281	1.2325		
PBFFS	245	1.1951			PBFFS	266	1.1667		
Data set 1	PBFFS_SA	Ratio PBFFS_SA to LB	Ratio Best SH to PBFFS_SA	Ratio PBFFS to PBFFS_SA	Data set 2	PBFFS_SA	Ratio PBFFS_SA to LB	Ratio Best SH to PBFFS_SA	Ratio PBFFS to PBFFS_SA
Exp 1	237	1.1561	1.1181	1.0338	Exp 1	263	1.1535	1.0684	1.0114
Exp 2	234	1.1415	1.1325	1.0470	Exp 2	255	1.1184	1.1020	1.0431
Exp 3	234	1.1415	1.1325	1.0470	Exp 3	261	1.1447	1.0766	1.0192
Exp 4	228	1.1122	1.1623	1.0746	Exp 4	257	1.1272	1.0934	1.0350
Exp 5	234	1.1415	1.1325	1.0470	Exp 5	254	1.1140	1.1063	1.0472
Exp 6	236	1.1512	1.1229	1.0381	Exp 6	255	1.1184	1.1020	1.0431
Exp 7	232	1.1317	1.1422	1.0560	Exp 7	261	1.1447	1.0766	1.0192
Exp 8	232	1.1317	1.1422	1.0560	Exp 8	256	1.1228	1.0977	1.0391
Exp 9	238	1.1610	1.1134	1.0294	Exp 9	257	1.1272	1.0934	1.0350
Exp 10	234	1.1415	1.1325	1.0470	Exp 10	258	1.1316	1.0891	1.0310

Data set 3		Ratio to LB	Stage Configuration 3-1-2		Data set 4		Ratio to LB	Stage Configuration 3-1-2	
LB	219				LB	214			
SH1	269	1.2283	Buffers size = 3		SH1	256	1.1963	Buffers size = 3	
SH2	273	1.2466			SH2	268	1.2523		
PBFFS	255	1.1644			PBFFS	254	1.1869		
Data set 3	PBFFS_SA	Ratio PBFFS_SA to LB	Ratio Best SH to PBFFS_SA	Ratio PBFFS to PBFFS_SA	Data set 4	PBFFS_SA	Ratio PBFFS_SA to LB	Ratio Best SH to PBFFS_SA	Ratio PBFFS to PBFFS_SA
Exp 1	238	1.0868	1.1303	1.0714	Exp 1	240	1.1215	1.0667	1.0583
Exp 2	241	1.1005	1.1162	1.0581	Exp 2	241	1.1262	1.0622	1.0539
Exp 3	249	1.1370	1.0803	1.0241	Exp 3	240	1.1215	1.0667	1.0583
Exp 4	246	1.1233	1.0935	1.0366	Exp 4	242	1.1308	1.0579	1.0496
Exp 5	244	1.1142	1.1025	1.0451	Exp 5	242	1.1308	1.0579	1.0496
Exp 6	240	1.0959	1.1208	1.0625	Exp 6	244	1.1402	1.0492	1.0410
Exp 7	237	1.0822	1.1350	1.0759	Exp 7	239	1.1168	1.0711	1.0628
Exp 8	242	1.1050	1.1116	1.0537	Exp 8	242	1.1308	1.0579	1.0496
Exp 9	246	1.1233	1.0935	1.0366	Exp 9	241	1.1262	1.0622	1.0539
Exp 10	242	1.1050	1.1116	1.0537	Exp 10	245	1.1449	1.0449	1.0367

Data set 5		Ratio to LB	Stage Configuration 3-1-2		Data set 6		Ratio to LB	Stage Configuration 3-1-2	
LB	246				LB	231			
SH1	305	1.2398	Buffers size = 3		SH1	291	1.2597	Buffers size = 3	
SH2	303	1.2317			SH2	293	1.2684		
PBFFS	292	1.1870			PBFFS	279	1.2078		
Data set 5	PBFFS_SA	Ratio PBFFS_SA to LB	Ratio Best SH to PBFFS_SA	Ratio PBFFS to PBFFS_SA	Data set 6	PBFFS_SA	Ratio PBFFS_SA to LB	Ratio Best SH to PBFFS_SA	Ratio PBFFS to PBFFS_SA
Exp 1	283	1.1504	1.0707	1.0318	Exp 1	267	1.1558	1.0899	1.0449
Exp 2	281	1.1423	1.0783	1.0391	Exp 2	261	1.1299	1.1149	1.0690
Exp 3	283	1.1504	1.0707	1.0318	Exp 3	262	1.1342	1.1107	1.0649
Exp 4	284	1.1545	1.0669	1.0282	Exp 4	266	1.1515	1.0940	1.0489
Exp 5	282	1.1463	1.0745	1.0355	Exp 5	266	1.1515	1.0940	1.0489
Exp 6	285	1.1585	1.0632	1.0246	Exp 6	261	1.1299	1.1149	1.0690
Exp 7	280	1.1382	1.0821	1.0429	Exp 7	266	1.1515	1.0940	1.0489
Exp 8	280	1.1382	1.0821	1.0429	Exp 8	263	1.1385	1.1065	1.0608
Exp 9	281	1.1423	1.0783	1.0391	Exp 9	266	1.1515	1.0940	1.0489
Exp 10	285	1.1585	1.0632	1.0246	Exp 10	267	1.1558	1.0899	1.0449

Data set 7		Ratio to LB	Stage Configuration 3-1-2		Data set 8		Ratio to LB	Stage Configuration 3-1-2	
LB	224				LB	230			
SH1	280	1.2500	Buffers size = 3		SH1	299	1.3000	Buffers size = 3	
SH2	274	1.2232			SH2	290	1.2609		
PBFFS	263	1.1741			PBFFS	280	1.2174		
Data set 7	PBFFS_SA	Ratio PBFFS_SA to LB	Ratio Best SH to PBFFS_SA	Ratio PBFFS to PBFFS_SA	Data set 8	PBFFS_SA	Ratio PBFFS_SA to LB	Ratio Best SH to PBFFS_SA	Ratio PBFFS to PBFFS_SA
Exp 1	252	1.1250	1.0873	1.0437	Exp 1	270	1.1739	1.0741	1.0370
Exp 2	246	1.0982	1.1138	1.0691	Exp 2	268	1.1652	1.0821	1.0448
Exp 3	256	1.1429	1.0703	1.0273	Exp 3	270	1.1739	1.0741	1.0370
Exp 4	253	1.1295	1.0830	1.0395	Exp 4	269	1.1696	1.0781	1.0409
Exp 5	253	1.1295	1.0830	1.0395	Exp 5	272	1.1826	1.0662	1.0294
Exp 6	249	1.1116	1.1004	1.0562	Exp 6	272	1.1826	1.0662	1.0294
Exp 7	250	1.1161	1.0960	1.0520	Exp 7	270	1.1739	1.0741	1.0370
Exp 8	254	1.1339	1.0787	1.0354	Exp 8	268	1.1652	1.0821	1.0448
Exp 9	251	1.1205	1.0916	1.0478	Exp 9	273	1.1870	1.0623	1.0256
Exp 10	252	1.1250	1.0873	1.0437	Exp 10	269	1.1696	1.0781	1.0409

Data set 9		Ratio to LB	Stage Configuration 3-1-2		Data set 10		Ratio to LB	Stage Configuration 3-1-2	
LB	258				LB	199			
SH1	322	1.2481	Buffers size = 3		SH1	253	1.2714	Buffers size = 3	
SH2	324	1.2558			SH2	255	1.2814		
PBFFS	303	1.1744			PBFFS	241	1.2111		
Data set 9	PBFFS_SA	Ratio PBFFS_SA to LB	Ratio Best SH to PBFFS_SA	Ratio PBFFS to PBFFS_SA	Data set 10	PBFFS_SA	Ratio PBFFS_SA to LB	Ratio Best SH to PBFFS_SA	Ratio PBFFS to PBFFS_SA
Exp 1	287	1.1124	1.1220	1.0557	Exp 1	230	1.1558	1.1000	1.0478
Exp 2	293	1.1357	1.0990	1.0341	Exp 2	229	1.1508	1.1048	1.0524
Exp 3	295	1.1434	1.0915	1.0271	Exp 3	230	1.1558	1.1000	1.0478
Exp 4	293	1.1357	1.0990	1.0341	Exp 4	231	1.1608	1.0952	1.0433
Exp 5	299	1.1589	1.0769	1.0134	Exp 5	231	1.1608	1.0952	1.0433
Exp 6	291	1.1279	1.1065	1.0412	Exp 6	227	1.1407	1.1145	1.0617
Exp 7	293	1.1357	1.0990	1.0341	Exp 7	229	1.1508	1.1048	1.0524
Exp 8	293	1.1357	1.0990	1.0341	Exp 8	233	1.1709	1.0858	1.0343
Exp 9	289	1.1202	1.1142	1.0484	Exp 9	234	1.1759	1.0812	1.0299
Exp 10	296	1.1473	1.0878	1.0236	Exp 10	232	1.1658	1.0905	1.0388

Data set 1		Ratio to LB	Stage Configuration 2-1-2-1		Data set 2		Ratio to LB	Stage Configuration 2-1-2-1	
LB	206				LB	237			
SH1	280	1.3592	Buffers size = 3		SH1	310	1.3080	Buffers size = 3	
SH2	277	1.3447			SH2	298	1.2574		
RBFFS	268	1.3010			RBFFS	284	1.1983		
Data set 1	RBFFS_SA	Ratio RBFFS_SA to LB	Ratio Best SH to RBFFS_SA	Ratio RBFFS to RBFFS_SA	Data set 2	RBFFS_SA	Ratio RBFFS_SA to LB	Ratio Best SH to RBFFS_SA	Ratio RBFFS to RBFFS_SA
Exp 1	240	1.1650	1.1542	1.1167	Exp 1	258	1.0886	1.1550	1.1008
Exp 2	238	1.1553	1.1639	1.1261	Exp 2	266	1.1224	1.1203	1.0677
Exp 3	239	1.1602	1.1590	1.1213	Exp 3	264	1.1139	1.1288	1.0758
Exp 4	239	1.1602	1.1590	1.1213	Exp 4	263	1.1097	1.1331	1.0798
Exp 5	239	1.1602	1.1590	1.1213	Exp 5	263	1.1097	1.1331	1.0798
Exp 6	240	1.1650	1.1542	1.1167	Exp 6	267	1.1266	1.1161	1.0637
Exp 7	235	1.1408	1.1787	1.1404	Exp 7	265	1.1181	1.1245	1.0717
Exp 8	239	1.1602	1.1590	1.1213	Exp 8	260	1.0970	1.1462	1.0923
Exp 9	239	1.1602	1.1590	1.1213	Exp 9	265	1.1181	1.1245	1.0717
Exp 10	240	1.1650	1.1542	1.1167	Exp 10	266	1.1224	1.1203	1.0677

Data set 3		Ratio to LB	Stage Configuration 2-1-2-1		Data set 4		Ratio to LB	Stage Configuration 2-1-2-1	
LB	225				LB	214			
SH1	294	1.3067	Buffers size = 3		SH1	277	1.2944	Buffers size = 3	
SH2	296	1.3156			SH2	278	1.2991		
RBFFS	289	1.2844			RBFFS	279	1.3037		
Data set 3	RBFFS_SA	Ratio RBFFS_SA to LB	Ratio Best SH to RBFFS_SA	Ratio RBFFS to RBFFS_SA	Data set 4	RBFFS_SA	Ratio RBFFS_SA to LB	Ratio Best SH to RBFFS_SA	Ratio RBFFS to RBFFS_SA
Exp 1	261	1.1600	1.1264	1.1073	Exp 1	244	1.1402	1.1352	1.1434
Exp 2	251	1.1156	1.1713	1.1514	Exp 2	237	1.1075	1.1688	1.1772
Exp 3	257	1.1422	1.1440	1.1245	Exp 3	241	1.1262	1.1494	1.1577
Exp 4	256	1.1378	1.1484	1.1289	Exp 4	238	1.1121	1.1639	1.1723
Exp 5	255	1.1333	1.1529	1.1333	Exp 5	245	1.1449	1.1306	1.1388
Exp 6	261	1.1600	1.1264	1.1073	Exp 6	246	1.1495	1.1260	1.1341
Exp 7	254	1.1289	1.1575	1.1378	Exp 7	246	1.1495	1.1260	1.1341
Exp 8	251	1.1156	1.1713	1.1514	Exp 8	243	1.1355	1.1399	1.1481
Exp 9	256	1.1378	1.1484	1.1289	Exp 9	241	1.1262	1.1494	1.1577
Exp 10	256	1.1378	1.1484	1.1289	Exp 10	245	1.1449	1.1306	1.1388

Data set 5		Ratio to LB	Stage Configuration 2-1-2-1		Data set 6		Ratio to LB	Stage Configuration 2-1-2-1	
LB	248				LB	234			
SH1	312	1.2581	Buffers size = 3		SH1	307	1.3120	Buffers size = 3	
SH2	319	1.2863			SH2	294	1.2564		
RBFFS	315	1.2702			RBFFS	291	1.2436		
Data set 5	RBFFS_SA	Ratio RBFFS_SA to LB	Ratio Best SH to RBFFS_SA	Ratio RBFFS to RBFFS_SA	Data set 6	RBFFS_SA	Ratio RBFFS_SA to LB	Ratio Best SH to RBFFS_SA	Ratio RBFFS to RBFFS_SA
Exp 1	286	1.1532	1.0909	1.1014	Exp 1	271	1.1581	1.0849	1.0738
Exp 2	283	1.1411	1.1025	1.1131	Exp 2	272	1.1624	1.0809	1.0699
Exp 3	284	1.1452	1.0986	1.1092	Exp 3	268	1.1453	1.0970	1.0858
Exp 4	285	1.1492	1.0947	1.1053	Exp 4	274	1.1709	1.0730	1.0620
Exp 5	287	1.1573	1.0871	1.0976	Exp 5	267	1.1410	1.1011	1.0899
Exp 6	280	1.1290	1.1143	1.1250	Exp 6	267	1.1410	1.1011	1.0899
Exp 7	290	1.1694	1.0759	1.0862	Exp 7	276	1.1795	1.0652	1.0543
Exp 8	288	1.1613	1.0833	1.0938	Exp 8	265	1.1325	1.1094	1.0981
Exp 9	291	1.1734	1.0722	1.0825	Exp 9	270	1.1538	1.0889	1.0778
Exp 10	284	1.1452	1.0986	1.1092	Exp 10	267	1.1410	1.1011	1.0899

Data set 7		Ratio to LB	Stage Configuration 2-1-2-1		Data set 8		Ratio to LB	Stage Configuration 2-1-2-1	
LB	224				LB	239			
SH1	297	1.3259	Buffers size = 3		SH1	304	1.2720	Buffers size = 3	
SH2	288	1.2857			SH2	307	1.2845		
RBFFS	275	1.2277			RBFFS	295	1.2343		
Data set 7	RBFFS_SA	Ratio RBFFS_SA to LB	Ratio Best SH to RBFFS_SA	Ratio RBFFS to RBFFS_SA	Data set 8	RBFFS_SA	Ratio RBFFS_SA to LB	Ratio Best SH to RBFFS_SA	Ratio RBFFS to RBFFS_SA
Exp 1	260	1.1607	1.1077	1.0577	Exp 1	275	1.1506	1.1055	1.0727
Exp 2	261	1.1652	1.1034	1.0536	Exp 2	273	1.1423	1.1136	1.0806
Exp 3	262	1.1696	1.0992	1.0496	Exp 3	272	1.1381	1.1176	1.0846
Exp 4	255	1.1384	1.1294	1.0784	Exp 4	274	1.1464	1.1095	1.0766
Exp 5	265	1.1830	1.0868	1.0377	Exp 5	272	1.1381	1.1176	1.0846
Exp 6	261	1.1652	1.1034	1.0536	Exp 6	276	1.1548	1.1014	1.0688
Exp 7	261	1.1652	1.1034	1.0536	Exp 7	275	1.1506	1.1055	1.0727
Exp 8	263	1.1741	1.0951	1.0456	Exp 8	271	1.1339	1.1218	1.0886
Exp 9	260	1.1607	1.1077	1.0577	Exp 9	274	1.1464	1.1095	1.0766
Exp 10	260	1.1607	1.1077	1.0577	Exp 10	270	1.1297	1.1259	1.0926

Data set 9		Ratio to LB	Stage Configuration 2-1-2-1		Data set 10		Ratio to LB	Stage Configuration 2-1-2-1	
LB	262				LB	216			
SH1	327	1.2481	Buffers size = 3		SH1	276	1.2778	Buffers size = 3	
SH2	322	1.2290			SH2	285	1.3194		
RBFFS	323	1.2328			RBFFS	275	1.2731		
Data set 9	RBFFS_SA	Ratio RBFFS_SA to LB	Ratio Best SH to RBFFS_SA	Ratio RBFFS to RBFFS_SA	Data set 10	RBFFS_SA	Ratio RBFFS_SA to LB	Ratio Best SH to RBFFS_SA	Ratio RBFFS to RBFFS_SA
Exp 1	303	1.1565	1.0627	1.0660	Exp 1	242	1.1204	1.1405	1.1364
Exp 2	302	1.1527	1.0662	1.0695	Exp 2	242	1.1204	1.1405	1.1364
Exp 3	303	1.1565	1.0627	1.0660	Exp 3	245	1.1343	1.1265	1.1224
Exp 4	301	1.1489	1.0698	1.0731	Exp 4	247	1.1435	1.1174	1.1134
Exp 5	302	1.1527	1.0662	1.0695	Exp 5	245	1.1343	1.1265	1.1224
Exp 6	303	1.1565	1.0627	1.0660	Exp 6	250	1.1574	1.1040	1.1000
Exp 7	304	1.1603	1.0592	1.0625	Exp 7	245	1.1343	1.1265	1.1224
Exp 8	299	1.1412	1.0769	1.0803	Exp 8	240	1.1111	1.1500	1.1458
Exp 9	299	1.1412	1.0769	1.0803	Exp 9	242	1.1204	1.1405	1.1364
Exp 10	304	1.1603	1.0592	1.0625	Exp 10	244	1.1296	1.1311	1.1270

Data set 1		Ratio to LB	Stage Configuration 2-1-2-1		Data set 2		Ratio to LB	Stage Configuration 2-1-2-1	
LB	206				LB	237			
SH1	280	1.3592	Buffers size = 3		SH1	310	1.3080	Buffers size = 3	
SH2	277	1.3447			SH2	298	1.2574		
PBFFS	256	1.2427			PBFFS	280	1.1814		
Data set 1	PBFFS_SA	Ratio PBFFS_SA to LB	Ratio Best SH to PBFFS_SA	Ratio PBFFS to PBFFS_SA	Data set 2	PBFFS_SA	Ratio PBFFS_SA to LB	Ratio Best SH to PBFFS_SA	Ratio PBFFS to PBFFS_SA
Exp 1	251	1.2184	1.1036	1.0199	Exp 1	270	1.1392	1.1037	1.0370
Exp 2	250	1.2136	1.1080	1.0240	Exp 2	274	1.1561	1.0876	1.0219
Exp 3	252	1.2233	1.0992	1.0159	Exp 3	273	1.1519	1.0916	1.0256
Exp 4	254	1.2330	1.0906	1.0079	Exp 4	273	1.1519	1.0916	1.0256
Exp 5	246	1.1942	1.1260	1.0407	Exp 5	267	1.1266	1.1161	1.0487
Exp 6	251	1.2184	1.1036	1.0199	Exp 6	272	1.1477	1.0956	1.0294
Exp 7	252	1.2233	1.0992	1.0159	Exp 7	276	1.1646	1.0797	1.0145
Exp 8	252	1.2233	1.0992	1.0159	Exp 8	274	1.1561	1.0876	1.0219
Exp 9	250	1.2136	1.1080	1.0240	Exp 9	274	1.1561	1.0876	1.0219
Exp 10	251	1.2184	1.1036	1.0199	Exp 10	269	1.1350	1.1078	1.0409

Data set 3		Ratio to LB	Stage Configuration 2-1-2-1		Data set 4		Ratio to LB	Stage Configuration 2-1-2-1	
LB	225				LB	214			
SH1	294	1.3067	Buffers size = 3		SH1	277	1.2944	Buffers size = 3	
SH2	296	1.3156			SH2	278	1.2991		
PBFFS	280	1.2444			PBFFS	267	1.2477		
Data set 3	PBFFS_SA	Ratio PBFFS_SA to LB	Ratio Best SH to PBFFS_SA	Ratio PBFFS to PBFFS_SA	Data set 4	PBFFS_SA	Ratio PBFFS_SA to LB	Ratio Best SH to PBFFS_SA	Ratio PBFFS to PBFFS_SA
Exp 1	261	1.1600	1.1264	1.0728	Exp 1	258	1.2056	1.0736	1.0349
Exp 2	260	1.1556	1.1308	1.0769	Exp 2	245	1.1449	1.1306	1.0898
Exp 3	266	1.1822	1.1053	1.0526	Exp 3	255	1.1916	1.0863	1.0471
Exp 4	264	1.1733	1.1136	1.0606	Exp 4	254	1.1869	1.0906	1.0512
Exp 5	263	1.1689	1.1179	1.0646	Exp 5	256	1.1963	1.0820	1.0430
Exp 6	263	1.1689	1.1179	1.0646	Exp 6	256	1.1963	1.0820	1.0430
Exp 7	268	1.1911	1.0970	1.0448	Exp 7	254	1.1869	1.0906	1.0512
Exp 8	266	1.1822	1.1053	1.0526	Exp 8	256	1.1963	1.0820	1.0430
Exp 9	266	1.1822	1.1053	1.0526	Exp 9	257	1.2009	1.0778	1.0389
Exp 10	264	1.1733	1.1136	1.0606	Exp 10	255	1.1916	1.0863	1.0471

Data set 5		Ratio to LB	Stage Configuration 2-1-2-1		Data set 6		Ratio to LB	Stage Configuration 2-1-2-1	
LB	248				LB	234			
SH1	312	1.2581	Buffers size = 3		SH1	307	1.3120	Buffers size = 3	
SH2	319	1.2863			SH2	294	1.2564		
PBFFS	302	1.2177			PBFFS	282	1.2051		
Data set 5	PBFFS_SA	Ratio PBFFS_SA to LB	Ratio Best SH to PBFFS_SA	Ratio PBFFS to PBFFS_SA	Data set 6	PBFFS_SA	Ratio PBFFS_SA to LB	Ratio Best SH to PBFFS_SA	Ratio PBFFS to PBFFS_SA
Exp 1	297	1.1976	1.0505	1.0168	Exp 1	270	1.1538	1.0889	1.0444
Exp 2	300	1.2097	1.0400	1.0067	Exp 2	271	1.1581	1.0849	1.0406
Exp 3	293	1.1815	1.0648	1.0307	Exp 3	277	1.1838	1.0614	1.0181
Exp 4	294	1.1855	1.0612	1.0272	Exp 4	277	1.1838	1.0614	1.0181
Exp 5	290	1.1694	1.0759	1.0414	Exp 5	277	1.1838	1.0614	1.0181
Exp 6	293	1.1815	1.0648	1.0307	Exp 6	275	1.1752	1.0691	1.0255
Exp 7	301	1.2137	1.0365	1.0033	Exp 7	277	1.1838	1.0614	1.0181
Exp 8	291	1.1734	1.0722	1.0378	Exp 8	275	1.1752	1.0691	1.0255
Exp 9	291	1.1734	1.0722	1.0378	Exp 9	277	1.1838	1.0614	1.0181
Exp 10	292	1.1774	1.0685	1.0342	Exp 10	277	1.1838	1.0614	1.0181

Data set 7		Ratio to LB	Stage Configuration 2-1-2-1		Data set 8		Ratio to LB	Stage Configuration 2-1-2-1	
LB	224				LB	239			
SH1	297	1.3259	Buffers size = 3		SH1	304	1.2720	Buffers size = 3	
SH2	288	1.2857			SH2	307	1.2845		
PBFFS	276	1.2321			PBFFS	294	1.2301		
Data set 7	PBFFS_SA	Ratio PBFFS_SA to LB	Ratio Best SH to PBFFS_SA	Ratio PBFFS to PBFFS_SA	Data set 8	PBFFS_SA	Ratio PBFFS_SA to LB	Ratio Best SH to PBFFS_SA	Ratio PBFFS to PBFFS_SA
Exp 1	262	1.1696	1.0992	1.0534	Exp 1	283	1.1841	1.0742	1.0389
Exp 2	262	1.1696	1.0992	1.0534	Exp 2	279	1.1674	1.0896	1.0538
Exp 3	270	1.2054	1.0667	1.0222	Exp 3	278	1.1632	1.0935	1.0576
Exp 4	269	1.2009	1.0706	1.0260	Exp 4	281	1.1757	1.0819	1.0463
Exp 5	263	1.1741	1.0951	1.0494	Exp 5	289	1.2092	1.0519	1.0173
Exp 6	265	1.1830	1.0868	1.0415	Exp 6	283	1.1841	1.0742	1.0389
Exp 7	268	1.1964	1.0746	1.0299	Exp 7	283	1.1841	1.0742	1.0389
Exp 8	259	1.1563	1.1120	1.0656	Exp 8	284	1.1883	1.0704	1.0352
Exp 9	264	1.1786	1.0909	1.0455	Exp 9	279	1.1674	1.0896	1.0538
Exp 10	266	1.1875	1.0827	1.0376	Exp 10	289	1.2092	1.0519	1.0173

Data set 9		Ratio to LB	Stage Configuration 2-1-2-1		Data set 10		Ratio to LB	Stage Configuration 2-1-2-1	
LB	262				LB	216			
SH1	327	1.2481	Buffers size = 3		SH1	276	1.2778	Buffers size = 3	
SH2	322	1.2290			SH2	285	1.3194		
PBFFS	306	1.1679			PBFFS	267	1.2361		
Data set 9	PBFFS_SA	Ratio PBFFS_SA to LB	Ratio Best SH to PBFFS_SA	Ratio PBFFS to PBFFS_SA	Data set 10	PBFFS_SA	Ratio PBFFS_SA to LB	Ratio Best SH to PBFFS_SA	Ratio PBFFS to PBFFS_SA
Exp 1	298	1.1374	1.0805	1.0268	Exp 1	254	1.1759	1.0866	1.0512
Exp 2	296	1.1298	1.0878	1.0338	Exp 2	252	1.1667	1.0952	1.0595
Exp 3	298	1.1374	1.0805	1.0268	Exp 3	251	1.1620	1.0996	1.0637
Exp 4	300	1.1450	1.0733	1.0200	Exp 4	251	1.1620	1.0996	1.0637
Exp 5	296	1.1298	1.0878	1.0338	Exp 5	247	1.1435	1.1174	1.0810
Exp 6	298	1.1374	1.0805	1.0268	Exp 6	251	1.1620	1.0996	1.0637
Exp 7	300	1.1450	1.0733	1.0200	Exp 7	252	1.1667	1.0952	1.0595
Exp 8	298	1.1374	1.0805	1.0268	Exp 8	251	1.1620	1.0996	1.0637
Exp 9	295	1.1260	1.0915	1.0373	Exp 9	251	1.1620	1.0996	1.0637
Exp 10	301	1.1489	1.0698	1.0166	Exp 10	248	1.1481	1.1129	1.0766

Data set 1		Ratio to LB	Stage Configuration 3-3-2-2-1		Data set 2		Ratio to LB	Stage Configuration 3-3-2-2-1	
LB	214	-			LB	243	-		
SH1	313	1.4626	Buffers size = 4		SH1	334	1.3745	Buffers size = 4	
SH2	315	1.4720			SH2	348	1.4321		
RBFFS	320	1.4953			RBFFS	351	1.4444		
Data set 1	RBFFS_SA	Ratio RBFFS_SA to LB	Ratio Best SH to RBFFS_SA	Ratio RBFFS to RBFFS_SA	Data set 2	RBFFS_SA	Ratio RBFFS_SA to LB	Ratio Best SH to RBFFS_SA	Ratio RBFFS to RBFFS_SA
Exp 1	292	1.3645	1.0719	1.0959	Exp 1	328	1.3498	1.0183	1.0701
Exp 2	297	1.3879	1.0539	1.0774	Exp 2	328	1.3498	1.0183	1.0701
Exp 3	298	1.3925	1.0503	1.0738	Exp 3	332	1.3663	1.0060	1.0572
Exp 4	300	1.4019	1.0433	1.0667	Exp 4	328	1.3498	1.0183	1.0701
Exp 5	295	1.3785	1.0610	1.0847	Exp 5	324	1.3333	1.0309	1.0833
Exp 6	295	1.3785	1.0610	1.0847	Exp 6	334	1.3745	1.0000	1.0509
Exp 7	292	1.3645	1.0719	1.0959	Exp 7	330	1.3580	1.0121	1.0636
Exp 8	287	1.3411	1.0906	1.1150	Exp 8	328	1.3498	1.0183	1.0701
Exp 9	290	1.3551	1.0793	1.1034	Exp 9	329	1.3539	1.0152	1.0669
Exp 10	292	1.3645	1.0719	1.0959	Exp 10	326	1.3416	1.0245	1.0767

Data set 3		Ratio to LB	Stage Configuration 3-3-2-2-1		Data set 4		Ratio to LB	Stage Configuration 3-3-2-2-1	
LB	226	-			LB	222	-		
SH1	331	1.4646	Buffers size = 4		SH1	305	1.3739	Buffers size = 4	
SH2	333	1.4735			SH2	319	1.4369		
RBFFS	338	1.4956			RBFFS	329	1.4820		
Data set 3	RBFFS_SA	Ratio RBFFS_SA to LB	Ratio Best SH to RBFFS_SA	Ratio RBFFS to RBFFS_SA	Data set 4	RBFFS_SA	Ratio RBFFS_SA to LB	Ratio Best SH to RBFFS_SA	Ratio RBFFS to RBFFS_SA
Exp 1	301	1.3319	1.0997	1.1229	Exp 1	303	1.3649	1.0066	1.0858
Exp 2	311	1.3761	1.0643	1.0868	Exp 2	306	1.3784	0.9967	1.0752
Exp 3	313	1.3850	1.0575	1.0799	Exp 3	300	1.3514	1.0167	1.0967
Exp 4	307	1.3584	1.0782	1.1010	Exp 4	304	1.3694	1.0033	1.0822
Exp 5	309	1.3673	1.0712	1.0939	Exp 5	294	1.3243	1.0374	1.1190
Exp 6	306	1.3540	1.0817	1.1046	Exp 6	303	1.3649	1.0066	1.0858
Exp 7	297	1.3142	1.1145	1.1380	Exp 7	301	1.3559	1.0133	1.0930
Exp 8	305	1.3496	1.0852	1.1082	Exp 8	306	1.3784	0.9967	1.0752
Exp 9	305	1.3496	1.0852	1.1082	Exp 9	312	1.4054	0.9776	1.0545
Exp 10	309	1.3673	1.0712	1.0939	Exp 10	304	1.3694	1.0033	1.0822

Data set 5		Ratio to LB	Stage Configuration 3-3-2-2-1		Data set 6		Ratio to LB	Stage Configuration 3-3-2-2-1	
LB	264	-			LB	247			
SH1	363	1.3750	Buffers size = 4		SH1	348	1.4089	Buffers size = 4	
SH2	377	1.4280			SH2	353	1.4291		
RBFFS	376	1.4242			RBFFS	361	1.4615		
Data set 5	RBFFS_SA	Ratio RBFFS_SA to LB	Ratio Best SH to RBFFS_SA	Ratio RBFFS to RBFFS_SA	Data set 6	RBFFS_SA	Ratio RBFFS_SA to LB	Ratio Best SH to RBFFS_SA	Ratio RBFFS to RBFFS_SA
Exp 1	355	1.3447	1.0225	1.0592	Exp 1	339	1.3725	1.0265	1.0649
Exp 2	356	1.3485	1.0197	1.0562	Exp 2	339	1.3725	1.0265	1.0649
Exp 3	353	1.3371	1.0283	1.0652	Exp 3	334	1.3522	1.0419	1.0808
Exp 4	357	1.3523	1.0168	1.0532	Exp 4	335	1.3563	1.0388	1.0776
Exp 5	348	1.3182	1.0431	1.0805	Exp 5	333	1.3482	1.0450	1.0841
Exp 6	357	1.3523	1.0168	1.0532	Exp 6	331	1.3401	1.0514	1.0906
Exp 7	352	1.3333	1.0313	1.0682	Exp 7	331	1.3401	1.0514	1.0906
Exp 8	352	1.3333	1.0313	1.0682	Exp 8	331	1.3401	1.0514	1.0906
Exp 9	346	1.3106	1.0491	1.0867	Exp 9	332	1.3441	1.0482	1.0873
Exp 10	352	1.3333	1.0313	1.0682	Exp 10	328	1.3279	1.0610	1.1006

Data set 7		Ratio to LB	Stage Configuration 3-3-2-2-1		Data set 8		Ratio to LB	Stage Configuration 3-3-2-2-1	
LB	239				LB	243			
SH1	329	1.3766	Buffers size = 4		SH1	355	1.4609	Buffers size = 4	
SH2	352	1.4728			SH2	355	1.4609		
RBFFS	346	1.4477			RBFFS	365	1.5021		
Data set 7	RBFFS_SA	Ratio RBFFS_SA to LB	Ratio Best SH to RBFFS_SA	Ratio RBFFS to RBFFS_SA	Data set 8	RBFFS_SA	Ratio RBFFS_SA to LB	Ratio Best SH to RBFFS_SA	Ratio RBFFS to RBFFS_SA
Exp 1	316	1.3222	1.0411	1.0949	Exp 1	342	1.4074	1.0380	1.0673
Exp 2	316	1.3222	1.0411	1.0949	Exp 2	331	1.3621	1.0725	1.1027
Exp 3	317	1.3264	1.0379	1.0915	Exp 3	336	1.3827	1.0565	1.0863
Exp 4	312	1.3054	1.0545	1.1090	Exp 4	335	1.3786	1.0597	1.0896
Exp 5	320	1.3389	1.0281	1.0813	Exp 5	339	1.3951	1.0472	1.0767
Exp 6	318	1.3305	1.0346	1.0881	Exp 6	339	1.3951	1.0472	1.0767
Exp 7	317	1.3264	1.0379	1.0915	Exp 7	330	1.3580	1.0758	1.1061
Exp 8	320	1.3389	1.0281	1.0813	Exp 8	337	1.3868	1.0534	1.0831
Exp 9	318	1.3305	1.0346	1.0881	Exp 9	345	1.4198	1.0290	1.0580
Exp 10	313	1.3096	1.0511	1.1054	Exp 10	337	1.3868	1.0534	1.0831

Data set 9		Ratio to LB	Stage Configuration 3-3-2-2-1		Data set 10		Ratio to LB	Stage Configuration 3-3-2-2-1	
LB	268				LB	227			
SH1	378	1.4104	Buffers size = 4		SH1	311	1.3700	Buffers size = 4	
SH2	379	1.4142			SH2	339	1.4934		
RBFFS	384	1.4328			RBFFS	322	1.4185		
Data set 9	RBFFS_SA	Ratio RBFFS_SA to LB	Ratio Best SH to RBFFS_SA	Ratio RBFFS to RBFFS_SA	Data set 10	RBFFS_SA	Ratio RBFFS_SA to LB	Ratio Best SH to RBFFS_SA	Ratio RBFFS to RBFFS_SA
Exp 1	364	1.3582	1.0385	1.0549	Exp 1	302	1.3304	1.0298	1.0662
Exp 2	368	1.3731	1.0272	1.0435	Exp 2	301	1.3260	1.0332	1.0698
Exp 3	356	1.3284	1.0618	1.0787	Exp 3	297	1.3084	1.0471	1.0842
Exp 4	357	1.3321	1.0588	1.0756	Exp 4	304	1.3392	1.0230	1.0592
Exp 5	364	1.3582	1.0385	1.0549	Exp 5	303	1.3348	1.0264	1.0627
Exp 6	368	1.3731	1.0272	1.0435	Exp 6	300	1.3216	1.0367	1.0733
Exp 7	364	1.3582	1.0385	1.0549	Exp 7	300	1.3216	1.0367	1.0733
Exp 8	362	1.3507	1.0442	1.0608	Exp 8	295	1.2996	1.0542	1.0915
Exp 9	361	1.3470	1.0471	1.0637	Exp 9	295	1.2996	1.0542	1.0915
Exp 10	364	1.3582	1.0385	1.0549	Exp 10	299	1.3172	1.0401	1.0769

Data set 1		Ratio to LB	Stage Configuration 3-3-2-2-1		Data set 2		Ratio to LB	Stage Configuration 3-3-2-2-1	
LB	214	-			LB	243	-		
SH1	313	1.4626	Buffers size = 4		SH1	334	1.3745	Buffers size = 4	
SH2	315	1.4720			SH2	348	1.4321		
PBFFS	297	1.3879			PBFFS	328	1.3498		
Data set 1	PBFFS_SA	Ratio PBFFS_SA to LB	Ratio Best SH to PBFFS_SA	Ratio PBFFS to PBFFS_SA	Data set 2	PBFFS_SA	Ratio PBFFS_SA to LB	Ratio Best SH to PBFFS_SA	Ratio PBFFS to PBFFS_SA
Exp 1	287	1.3411	1.0906	1.0348	Exp 1	325	1.3374	1.0277	1.0092
Exp 2	287	1.3411	1.0906	1.0348	Exp 2	319	1.3128	1.0470	1.0282
Exp 3	288	1.3458	1.0868	1.0313	Exp 3	322	1.3251	1.0373	1.0186
Exp 4	285	1.3318	1.0982	1.0421	Exp 4	318	1.3086	1.0503	1.0314
Exp 5	286	1.3364	1.0944	1.0385	Exp 5	318	1.3086	1.0503	1.0314
Exp 6	286	1.3364	1.0944	1.0385	Exp 6	319	1.3128	1.0470	1.0282
Exp 7	285	1.3318	1.0982	1.0421	Exp 7	323	1.3292	1.0341	1.0155
Exp 8	291	1.3598	1.0756	1.0206	Exp 8	321	1.3210	1.0405	1.0218
Exp 9	289	1.3505	1.0830	1.0277	Exp 9	320	1.3169	1.0438	1.0250
Exp 10	289	1.3505	1.0830	1.0277	Exp 10	321	1.3210	1.0405	1.0218

Data set 3		Ratio to LB	Stage Configuration 3-3-2-2-1		Data set 4		Ratio to LB	Stage Configuration 3-3-2-2-1	
LB	226	-			LB	222	-		
SH1	331	1.4646	Buffers size = 4		SH1	305	1.3739	Buffers size = 4	
SH2	333	1.4735			SH2	319	1.4369		
PBFFS	326	1.4425			PBFFS	296	1.3333		
Data set 3	PBFFS_SA	Ratio PBFFS_SA to LB	Ratio Best SH to PBFFS_SA	Ratio PBFFS to PBFFS_SA	Data set 4	PBFFS_SA	Ratio PBFFS_SA to LB	Ratio Best SH to PBFFS_SA	Ratio PBFFS to PBFFS_SA
Exp 1	296	1.3097	1.1182	1.1014	Exp 1	294	1.3243	1.0374	1.0068
Exp 2	292	1.2920	1.1336	1.1164	Exp 2	295	1.3288	1.0339	1.0034
Exp 3	295	1.3053	1.1220	1.1051	Exp 3	290	1.3063	1.0517	1.0207
Exp 4	295	1.3053	1.1220	1.1051	Exp 4	292	1.3153	1.0445	1.0137
Exp 5	296	1.3097	1.1182	1.1014	Exp 5	294	1.3243	1.0374	1.0068
Exp 6	293	1.2965	1.1297	1.1126	Exp 6	293	1.3198	1.0410	1.0102
Exp 7	296	1.3097	1.1182	1.1014	Exp 7	294	1.3243	1.0374	1.0068
Exp 8	298	1.3186	1.1107	1.0940	Exp 8	296	1.3333	1.0304	1.0000
Exp 9	292	1.2920	1.1336	1.1164	Exp 9	294	1.3243	1.0374	1.0068
Exp 10	293	1.2965	1.1297	1.1126	Exp 10	290	1.3063	1.0517	1.0207

Data set 5		Ratio to LB	Stage Configuration 3-3-2-2-1		Data set 6		Ratio to LB	Stage Configuration 3-3-2-2-1	
LB	264	-			LB	247			
SH1	363	1.3750	Buffers size = 4		SH1	348	1.4089	Buffers size = 4	
SH2	377	1.4280			SH2	353	1.4291		
PBFFS	358	1.3561			PBFFS	330	1.3360		
Data set 5	PBFFS_SA	Ratio PBFFS_SA to LB	Ratio Best SH to PBFFS_SA	Ratio PBFFS to PBFFS_SA	Data set 6	PBFFS_SA	Ratio PBFFS_SA to LB	Ratio Best SH to PBFFS_SA	Ratio PBFFS to PBFFS_SA
Exp 1	339	1.2841	1.0708	1.0560	Exp 1	324	1.3117	1.0741	1.0185
Exp 2	345	1.3068	1.0522	1.0377	Exp 2	318	1.2874	1.0943	1.0377
Exp 3	343	1.2992	1.0583	1.0437	Exp 3	324	1.3117	1.0741	1.0185
Exp 4	340	1.2879	1.0676	1.0529	Exp 4	323	1.3077	1.0774	1.0217
Exp 5	340	1.2879	1.0676	1.0529	Exp 5	323	1.3077	1.0774	1.0217
Exp 6	342	1.2955	1.0614	1.0468	Exp 6	324	1.3117	1.0741	1.0185
Exp 7	343	1.2992	1.0583	1.0437	Exp 7	324	1.3117	1.0741	1.0185
Exp 8	347	1.3144	1.0461	1.0317	Exp 8	320	1.2955	1.0875	1.0313
Exp 9	345	1.3068	1.0522	1.0377	Exp 9	321	1.2996	1.0841	1.0280
Exp 10	336	1.2727	1.0804	1.0655	Exp 10	324	1.3117	1.0741	1.0185

Data set 7		Ratio to LB	Stage Configuration 3-3-2-2-1		Data set 8		Ratio to LB	Stage Configuration 3-3-2-2-1	
LB	239				LB	243			
SH1	329	1.3766	Buffers size = 4		SH1	355	1.4609	Buffers size = 4	
SH2	352	1.4728			SH2	355	1.4609		
PBFFS	323	1.3515			PBFFS	342	1.4074		
Data set 7	PBFFS_SA	Ratio PBFFS_SA to LB	Ratio Best SH to PBFFS_SA	Ratio PBFFS to PBFFS_SA	Data set 8	PBFFS_SA	Ratio PBFFS_SA to LB	Ratio Best SH to PBFFS_SA	Ratio PBFFS to PBFFS_SA
Exp 1	311	1.3013	1.0579	1.0386	Exp 1	326	1.3416	1.0890	1.0491
Exp 2	308	1.2887	1.0682	1.0487	Exp 2	331	1.3621	1.0725	1.0332
Exp 3	309	1.2929	1.0647	1.0453	Exp 3	330	1.3580	1.0758	1.0364
Exp 4	307	1.2845	1.0717	1.0521	Exp 4	324	1.3333	1.0957	1.0556
Exp 5	308	1.2887	1.0682	1.0487	Exp 5	325	1.3374	1.0923	1.0523
Exp 6	307	1.2845	1.0717	1.0521	Exp 6	328	1.3498	1.0823	1.0427
Exp 7	302	1.2636	1.0894	1.0695	Exp 7	329	1.3539	1.0790	1.0395
Exp 8	313	1.3096	1.0511	1.0319	Exp 8	329	1.3539	1.0790	1.0395
Exp 9	306	1.2803	1.0752	1.0556	Exp 9	332	1.3663	1.0693	1.0301
Exp 10	310	1.2971	1.0613	1.0419	Exp 10	334	1.3745	1.0629	1.0240

Data set 9		Ratio to LB	Stage Configuration 3-3-2-2-1		Data set 10		Ratio to LB	Stage Configuration 3-3-2-2-1	
LB	268				LB	227			
SH1	378	1.4104	Buffers size = 4		SH1	311	1.3700	Buffers size = 4	
SH2	379	1.4142			SH2	339	1.4934		
PBFFS	356	1.3284			PBFFS	302	1.3304		
Data set 9	PBFFS_SA	Ratio PBFFS_SA to LB	Ratio Best SH to PBFFS_SA	Ratio PBFFS to PBFFS_SA	Data set 10	PBFFS_SA	Ratio PBFFS_SA to LB	Ratio Best SH to PBFFS_SA	Ratio PBFFS to PBFFS_SA
Exp 1	350	1.3060	1.0800	1.0171	Exp 1	290	1.2775	1.0724	1.0414
Exp 2	353	1.3172	1.0708	1.0085	Exp 2	293	1.2907	1.0614	1.0307
Exp 3	351	1.3097	1.0769	1.0142	Exp 3	293	1.2907	1.0614	1.0307
Exp 4	351	1.3097	1.0769	1.0142	Exp 4	289	1.2731	1.0761	1.0450
Exp 5	350	1.3060	1.0800	1.0171	Exp 5	290	1.2775	1.0724	1.0414
Exp 6	346	1.2910	1.0925	1.0289	Exp 6	290	1.2775	1.0724	1.0414
Exp 7	349	1.3022	1.0831	1.0201	Exp 7	293	1.2907	1.0614	1.0307
Exp 8	345	1.2873	1.0957	1.0319	Exp 8	288	1.2687	1.0799	1.0486
Exp 9	354	1.3209	1.0678	1.0056	Exp 9	293	1.2907	1.0614	1.0307
Exp 10	352	1.3134	1.0739	1.0114	Exp 10	291	1.2819	1.0687	1.0378