Abstract

MANNINAGARAJAN, PADMANABHAN. Rolling Horizon Plant Scheduling.
(Under the direction of Kristin A. Thoney)

Scheduling is critical in all industries as it helps to reduce delays in job completion by effectively using all available resources. The Virtual Factory is one of the many job shop-scheduling systems for scheduling large problems. The Virtual Factory is an iterative simulation based procedure that has been found to provide near-optimal solutions to industrial-sized problems. As the current version of the Virtual Factory has more than 40 classes, understanding it might be a time-consuming task even for an experienced C++ programmer. To make it easier for the user to perform experiments, a Visual Basic interface is developed.

The Virtual Factory has primarily been tested under transient conditions in which the plant is run until it is empty. In industry, each day jobs are released, the status of the plant is downloaded, and scheduling is performed. The best schedule is implemented until the plant is scheduled again. To analyze the potential performance of the Virtual Factory in industry, it is tested in a rolling horizon setting. Experiments with various parameters show that the Virtual Factory also performs well in these circumstances.

Rolling Horizon Plant Scheduling

by

PADMANABHAN MANNINAGARAJAN

A thesis submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Master of Science

TEXTILE MANAGEMENT TECHNOLOGY

Raleigh

2002

APPROVED BY

Thom J. Hodgson

Jeffrey A. Joines

Kristin A. Thoney
Chair of Advisory Committee

Dedication

I would like to dedicate this thesis to my parents, Nagarajan and Savithri. I will always be thankful for their constant encouragement and support all along, without which I would not have reached these heights.

Biography

Padmanabhan Manninagarajan was born on October 1, 1978, in Salem, Tamil Nadu, India. He attended Bala Bharathi Matriculation Higher Secondary School and completed his elementary and secondary studies in 1995. He joined P.S.G College of Technology in Coimbatore, India in 1995 and graduated with a B.Tech in Textile Technology in May 1999.

Padmanabhan enrolled at North Carolina State University in Fall 2000 to continue his studies to obtain a Masters in Textile Management and Technology.

Padmanabhan is the son of Nagarajan and Savithri.

Acknowledgements

I would like to express my sincere thanks to Dr. Kristin Thoney for her enthusiastic guidance and encouragement for this project and throughout the course of this study. She has all along provided valuable suggestions and constructive ideas and has assisted me a great deal in writing up this thesis.

I would also like to express my gratitude to the members of my advisory committee, Dr. Thom Hodgson, and Dr. Jeffrey Joines.

I would also like to thank all my friends for their help and support.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Production scheduling is important in all industries, particularly in the textile industry. As the textile industry has become progressively more global, competition is more fierce. To survive, U.S. companies must adopt technology to become more efficient. Production scheduling tools help create better schedules that have the potential to reduce overtime by better utilizing machines, even in the cases of textile and apparel products that have erratic demand. They can also help companies better satisfy their customers by providing more on-time orders.

There are many tools available for scheduling large problems. The Virtual Factory is one such tool that has been found to provide near-optimal solutions to industrial-sized problems in seconds. The Virtual Factory is an iterative simulation-based procedure, whose objective is to minimize the maximum lateness, L_{max}. To determine the effectiveness of its solutions, results are compared to a simple lower bound calculation.

The Virtual Factory is written in C++, and the current version has grown to include over 40 classes. Even for an experienced C++ programmer, understanding the complete program is a time-consuming undertaking. Just learning how to run it with the desired options can also be difficult. To remedy this situation, a Visual Basic

interface will be developed. Consequently, little training will be needed to be able to perform experimentation with the existing options.

The Virtual Factory, as the majority of the job shop scheduling algorithms found in the literature, has been tested exclusively under transient circumstances. In industry, though, running a plant until it is empty is rare. Instead, plants usually contain many different orders, with new orders arriving as older ones are completed. Scheduling is often performed on some regular basis, i.e. everyday. The best schedule is implemented until the plant is rescheduled. Thus, scheduling occurs on a rolling horizon basis. To test how well the Virtual Factory would perform in industry, it therefore will be tested under these circumstances.

In Chapter 2, a literature review for this thesis is given. Chapter 3 provides the details about the Visual Basic interface for the Virtual Factory. In Chapter 4, the results of testing the Virtual Factory on a rolling horizon basis are analyzed. Conclusions and future research are found in Chapter 5.

Chapter 2

Literature Review

2.1 The Virtual Factory

2.1.1 Introduction

The idea for this simulation-based job shop-scheduling algorithm was first proposed by Lawrence and Morton [17] and Vepsalainen and Morton [30]. Hodgson et al. [12] further developed it and named it the Virtual Factory. The Virtual Factory consists both of a scheduling algorithm and a lower bound. In general, it produces very good results.

2.1.2 Scheduling Procedure

Let d_i be the due date of job i and p_{ij} be the processing time of job i on machine j. Then the slack of job i on machine m is calculated as

$$Slack_{i,m} = d_i - \sum_{j \epsilon m^+} p_{ij},$$

where $m+$ is the set of all operations subsequent to machine m on job i's routing. Slack represents the latest possible time that a job can finish on a machine and still satisfy its final due date. As this does not include queuing time, slack did not perform well as a dispatching rule in early experiments found in the scheduling literature.

To remedy this situation, a revised slack value that incorporates queuing times is used as the sequencing rule in the Virtual Factory. Queuing times are recorded for

each job at each machine it visits in one iteration of the simulation and used in the next iteration. The revised slack for job *i* on machine *m* is computed as

$$Slack'_{i,m} = d_i - \sum_{j \epsilon m^{+}} p_{ij} - \sum_{j \epsilon m^{++}} q_{ij} ,$$

where m_i^{++} is the set of all subsequent operations to machine *m* on the routing sheet for job *i,* except the immediate subsequent operation. The simulation is run until the lower bound is achieved or a specified number of iterations is reached, and the best solution is saved.

2.1.3 Lower Bound

Hodgson et al. [12] chose to evaluate the quality of the schedules produced by the Virtual Factory through comparison to a lower bound. The lower bound is calculated by decomposing the job shop problem into individual one machine problems. To do this, an earliest start time and a latest finish time were calculated for each machine on each job's route. Let r_i be the release time of job *i.* Then the earliest possible start time for a job *i* on machine *m* is,

$$ES_{i,m} = r_i + \sum_{j \epsilon m^{-}} p_{ij} ,$$

where m^{-} is the set of all operations preceding machine *m* on job *i*'s routing sheet. The latest finish time for each job *i* on machine *m* is

$$LF_{i,m} = d_i - \sum_{j \epsilon m^{+}} p_{ij} ,$$

where m^+ is the set of all operations following machine m on the routing sheet of job i.

The lower bound for the job shop problem ($N/M/L_{max}$) is obtained by solving the $N/1/ L_{max} \mid r_i$ problem on each machine m by considering $LF_{i,m}$ as the effective due date for job i on machine m and $ES_{i,m}$ as the release time (r_i) for job i on machine m. Since $N/1/ L_{max} \mid r_i$ is NP-hard, a relaxation suggested by Baker and Su [3] is used. The relaxation is to allow preemption of a job in process whenever one with a more imminent due date becomes available.

The overall lower bound, $LB\ (L_{max})$, is computed as

$$LB\ (L_{max}) = \max_{m=1,M} \{LB_m(L_{max})\},$$

where $LB_m\ (L_{max})$ is the lower bound for machine m. The power of this lower bound is that there are M chances to get a tight bound.

2.1.4 Experimental Results

The Virtual Factory has provided optimal results when tested with two data sets from a large furniture manufacturing plant. The Virtual Factory was also tested on data generated randomly with the procedure suggested by Demirkol et al. [11]. The test results for a particular problem specified with the number of jobs, number of machines, number of operations per job, and processing time range, showed that the due date range had considerable effect on the performance of the Virtual Factory. When the difference between L_{max} and LB was plotted against the due date range, Hodgson et al. [13] observed that the difference was zero for low due date ranges and

the difference increased up to a specific value and remained at that value as the due date range was increased to larger values.

Figure 2.1 shows a graph with the difference between L_{max} and LB plotted against the due date range for a problem with 1000 jobs, 100 machines, 7 operations and processing time uniformly distributed between 1 and 200. Each point on the graph represents the average of solutions to 10 problems. The difference between L_{max} and LB represents the maximum by which the Virtual Factory solution could exceed the optimal solution.

Hodgson et al. [13] found that the difference was roughly equal to the expected queuing time of the L_{max} job. The L_{max} job should be processed immediately on the first machine on its route. For each subsequent machine on the job's route, usually another job will be in process when the L_{max} job arrives. On average, the job on the machine will be half way through processing and its processing time will be equal to the average processing time. Thus the expected queuing time of the L_{max} job is equal to $(O_{ps} - 1)(\overline{P}/2)$, where O_{ps} is the number of operations per job and $\overline{P}$ is the average processing time per job. For the problem shown in Figure 2.1, the expected queuing time is (7-1)(100.5/2) = 301.5, which is similar to the difference seen in the graph.

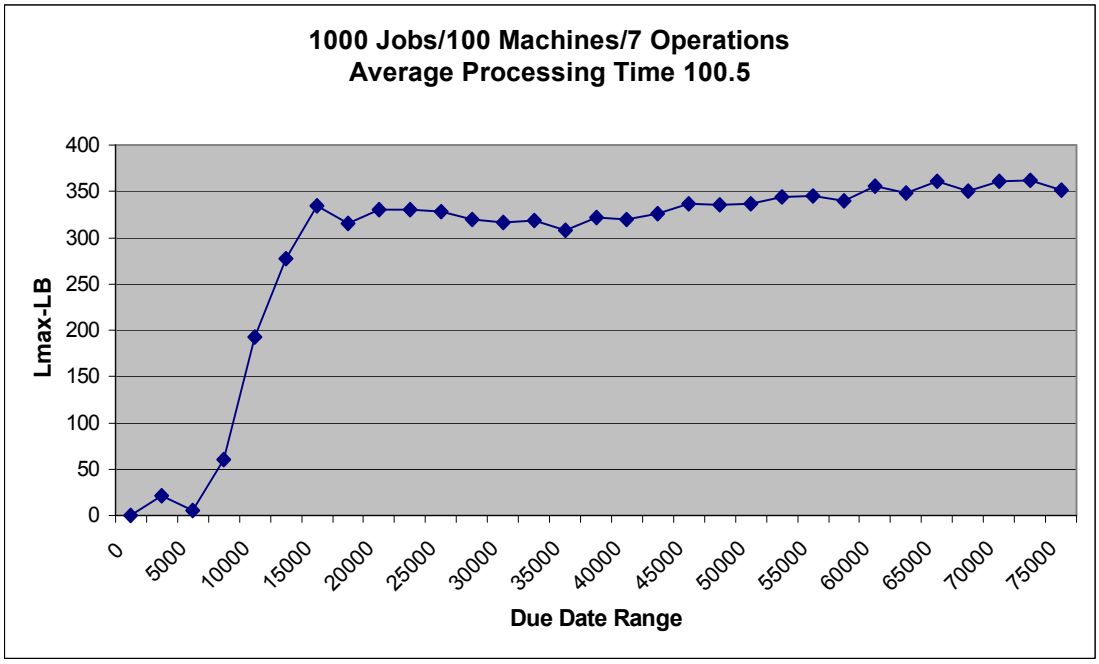

Figure 2.1: Virtual Factory: Performance vs. Due Date Range

2.2 Extensions of the Virtual Factory

2.2.1 Hot Job Acceleration

Hodgson et al. [13] proposed an extension of their scheduling heuristic by identifying critical jobs, i.e. those jobs whose lateness is equal to or close to L_{max}. They observed that a critical job might be delayed by a non-critical job already in process when the critical job arrives. Hodgson et al. tried to improve the schedule by inserting idle time into a machine schedule just before the arrival of a critical job if a non-critical job would otherwise prevent the critical job from starting immediately. By accelerating critical jobs throughout the system, in general, L_{max} decreased. This can be seen in Figure 2.2, which shows a comparison of the original Virtual Factory solution and the solution with the accelerating hot jobs procedure. For larger due date

ranges, it clearly shows that the difference between the L_{max} and LB has been reduced considerably in the accelerated version when compared with the original Virtual Factory solution.

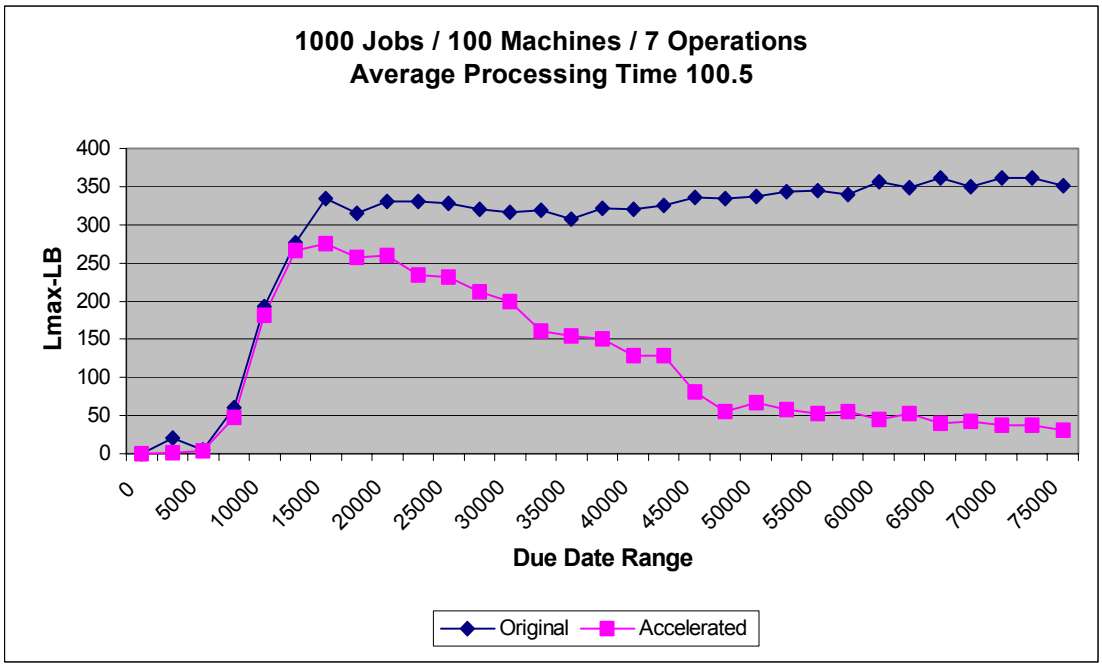

Figure 2.2: Comparison of Original vs. Accelerated Versions of Virtual Factory

2.2.2 Alternative Process Plans

Weintraub et al. [31] extended the methodology used by Hodgson et al. [12] by developing a procedure for scheduling jobs with alternative processes to minimize the manufacturing costs and at the same time satisfy due dates. A tabu search was used to evaluate the alternative process plans, and those plans were selected based upon their lower bound. The process plans were modified with alternative routings, operations and sequences. From the experiments, they found that there were substantial differences in performance between scheduling with and scheduling without

alternatives. Also scheduling with alternatives could identify optimal or near optimal schedules that minimize manufacturing costs and satisfy due dates. They found that scheduling with alternative operations resulted in the largest schedule improvement, and scheduling with alternative sequencing resulted in the smallest schedule improvement.

2.2.3 Determining Job Release Times

Zozom et al. [33] developed two heuristics for deciding when to release jobs to the shop floor. From the experimental results, both methods were effective at lowering work in process (WIP). In addition, the solutions that the heuristic provided were close to a lower bound computed on WIP.

2.2.4 Multi-Factory Scenarios

Thoney et al. [28] developed a detailed scheduling procedure for multi-factory scenarios, including inter-plant transportation. This required incorporating transportation into the Virtual Factory. Vehicles are batch processors, and their characteristics are different from that of conventional processors. In a batch processor, a number of jobs start and end processing together. Because of these special characteristics, initial experiments calculating revised slack for batch processors in the same manner as for conventional processors did not perform well. Consequently, a new revised slack calculation had to be computed for batch processors. In addition, a lower bound on batch processors had to be developed.

Thoney et al. considered the following four multi-factory scenarios: Two Factories in Series, Three Factories in Series, Two Factories Feeding One, and One

Factory Feeding Two. Experiments showed that performance for these scenarios was, in general, quite good. In transportation-constrained scenarios, the performance declined slightly. This was determined to be the results of transient effects occurring at the beginning and ending of the simulations. It was suggested that using rolling horizon scheduling might eliminate these transient effects.

2.2.5 Simulated Annealing

Schultz et al. [23] developed a simulated annealing procedure to be used as a post processing procedure with the Virtual Factory. Experiments were run using the problems generated by Demirkol et al. [11] and also with some industrial-sized problems used by Hodgson et al. [13]. Schultz et al. found that the simulated annealing procedure coupled with the Virtual Factory was an effective approach to improving solution quality. For the 160-benchmark problems generated by Demirkol, the best-known solution was obtained or bettered for 141 problems in 120 minutes. For 116 of these problems, the solution was improved. For the industrial-sized problems, this procedure provided significant improvement to Virtual Factory for additional run times of 1 to 5 minutes.

Figure 2.3 shows the comparison of the Virtual Factory solution with hot job acceleration and the simulated annealing solutions with different additional run times (1, 5 and 10 minutes). The difference between L_{max} and LB has been reduced greatly with the simulated annealing procedure, and it can also be seen that an increase in run time reduces the difference further.

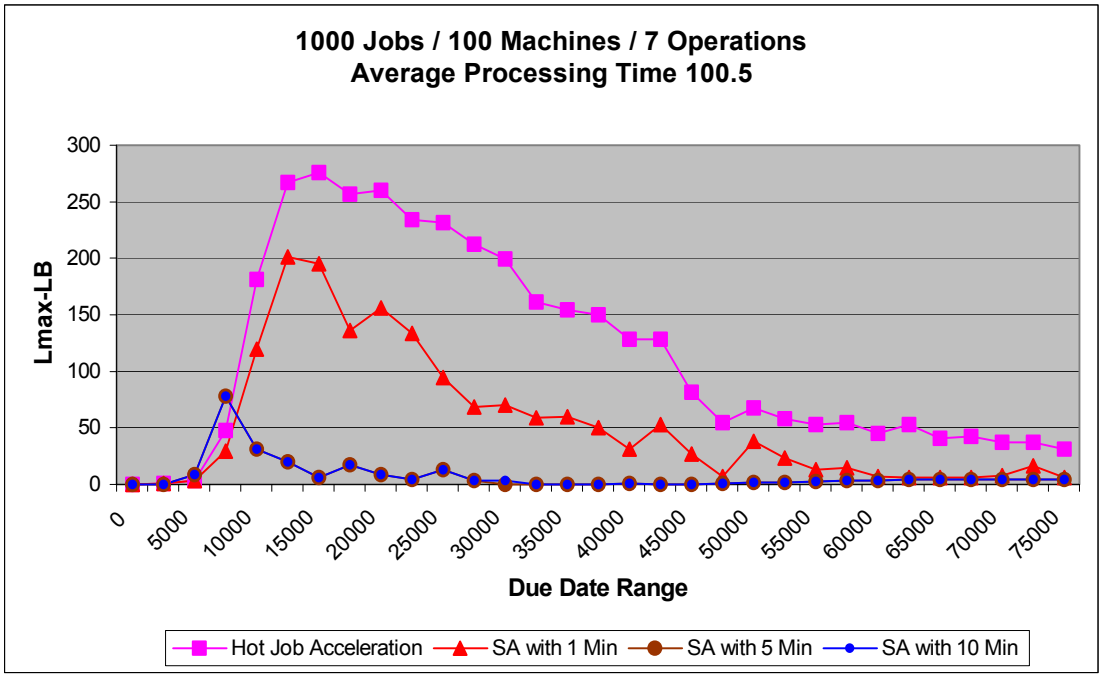

Figure 2.3: Comparison of Virtual Factory with Hot Job Acceleration vs. Procedure with Simulated Annealing

2.3 Rolling Horizon Scheduling

Previous work that treats rolling horizon-scheduling problems is presented in this section. In most of the earlier studies, lot sizing problems were applied in a rolling horizon setting and their performances were analyzed. Several conditions that were considered for rolling horizon schedules include deterministic demand, single machine and parallel machine problems, and probabilistic demand. Among the various factors that were studied are the length of the planning horizon, method of freezing the schedule, stability of the schedule, and cost performance.

2.3.1 Lot Sizing Problems

Baker [1] conducted an experimental study to test the effectiveness of schedules obtained from a finite-horizon planning model using limited information about future demand and applying those schedules on a rolling horizon basis. He considered the effects of the length of the forecast window, the cost structure, and the demand pattern. He found that the rolling schedules produced low-cost results. He also found that the rolling schedule's efficiency depended on the length of the forecast horizon.

Lundin and Morton [18] developed planning horizon procedures for the dynamic lot size model. They observed that the length of the planning horizon must be at least five EOQ cycles to ensure a solution within one percent of optimality.

Blackburn and Millen [4] compared the performance of the following four lot-sizing methods for single level assembly systems by implementing rolling schedules: part period cost balancing, Silver-Meal heuristic, Wagner-Whitin algorithm, and modified Silver-Meal algorithm. They found that the Silver-Meal heuristic provided better cost effectiveness than the rest.

Carlson et al. [9] extended the results obtained by Baker and came up with solutions for different conditions in which $N < T$, $N = mT$, and $N > T$ where, N is the horizon length, T is the length of the natural cycle, and m is an integer.

Chand [10] modified the dynamic lot size algorithm of Wagner and Whitin for rolling horizon environments and obtained better-cost performance than the Wagner-Whitin algorithm and the Silver-Meal heuristic.

Wemmerlöv and Whybark [32] studied the performance of fourteen different single stage lot-sizing procedures with probabilistic demands in a rolling schedule and ranked them by conducting experiments. They found that demand uncertainty changed the solutions to a larger extent than the constant demand solutions.

Russell and Urban [22] studied the effects of forecast length and accuracy in horizon extensions. They also conducted experiments to compare the Wagner-Whitin algorithm with horizon extensions with the Silver-Meal heuristic. They found that the Wagner-Whitin algorithm with horizon extensions performed better than Silver-Meal heuristic for large and moderate values of the planning horizons.

Matta and Guignard [19] developed a production-scheduling model that finds low-cost solutions to a mixed-integer programming formulation of the production lot-sizing problem. They measured the quality of the production schedules by comparing them to a lower bound found using lagrangian relaxation. They concluded that the total savings in annual production cost is reduced when more periods are added to the planning horizon.

Simpson [24] evaluated nine multiple level planning heuristics. He analyzed the relation of rolling horizon results to fixed horizon results in a deterministic demand environment. He used the relative cost ratio as the primary performance measure. He computed a tight lower bound on the lowest possible cost schedule to calculate the relative cost ratio. Of all the nine algorithms studied, he found that the Non-sequential Incremental Part Period Algorithm provided the lowest cost schedules under all conditions, except extremely short planning horizons.

Stadler [27] developed a modified model of the single level lot-sizing problem in which lot-sizing decisions consider demand forecasts beyond a given planning horizon. He solved the model by the Wagner and Whitin algorithm. He compared the results with four known heuristics and found that the modified model performed well with relatively little additional cost.

2.3.2 Single and Parallel Machine Problems

Ovacik and Uzsoy [20] developed rolling horizon heuristics to minimize maximum lateness on a single machine with sequence dependent setup times. The procedure solved several small sub-problems to optimality using a branch and bound procedure. A part of those solutions were implemented. They found that these procedures performed far better than the best dispatching rules.

Ovacik and Uzsoy [21] extended their procedure to include parallel machines and showed that the performance was also better than dispatching rules.

2.3.3 Probabilistic Conditions

Bookbinder and H'ng [5] developed a production-planning procedure for a rolling horizon setting with probabilistic demand. They compared their procedure with Silver's procedure (a procedure to determine the timing and sizes of replenishments for probabilistic demand with normally distributed forecast errors) in terms of the cost performance, percentage of demand shortage per period, and percentage of periods with stock-outs. They found that their procedure provided better-cost performance than that of Silver's procedure, while Silver's procedure yielded better results for the percentage of demand shortage per period and percentage of periods with stock-outs.

They also found that the production plan was better when there was more information, even if that information included some uncertainty.

Baker and Peterson [2] developed a framework for assessing the cost performance of rolling schedules. They analytically studied a quadratic-cost model for the effects of factors such as the length of the planning horizon, the uncertainty in demand forecasts, the amplitude of demand fluctuations in seasonal cases, and imposition of a terminal condition.

Kleindorfer and Kenreuther [14] developed a method to relate the stochastic planning problems to the planning procedures and information system within the industry. They also described how their procedure could be used for specifying stochastic horizons for aggregate planning problems in the industry.

2.3.4 Aggregate Production Schedules

Venkataraman and Smith [29] developed a master production scheduling model that considers the disaggregation of aggregate plans to a rolling horizon master production schedule with minimum batch-size restrictions for a fixed routing, batch production, process industry environment. Their model included multiple products, multiple production lines, capacity limitations, inventory requirements, and seasonal demands.

2.3.5 Production Smoothing Problem

Kunreuther and Morton [15] [16] developed algorithms to find horizons for production and workforce smoothing problem with deterministic demands by

considering such factors as holding costs, overtime, lost sales, simple subcontracting, under time, and backlogging.

2.3.6 Fixed Interval Scheduling

Fixed interval scheduling is characterized by production periods that are evenly spaced over time. Campbell [6] studied fixed interval scheduling in a rolling horizon framework by using the concept of time fencing. He also studied three different methods of finding safety stock in this context.

2.3.7 Freezing the Master Production Schedule

Sridharan et al. [25] [26] discussed the measurement of instability in the Master Production Schedule (MPS). They related the instability to three decision variables in managing the MPS in a rolling horizon environment. The decision variables were the method used to freeze the MPS (period-based freezing and parts-based freezing), the proportion of the MPS frozen, and the length of the planning horizon for the MPS.

Chapter 3

Visual Basic Interface to the Virtual Factory

3.1 Introduction

The current academic version of the Virtual Factory is written in C++ and contains more than 40 classes. Understanding that many classes can be a time-consuming task, even for an experienced C++ programmer. In addition, just running the desired experiment is not straightforward because of the large number of critical parameters that are hidden in the massive amount of code. To overcome this situation, a Visual Basic (VB) program was developed that serves as a front end to the VF. With the VB program, only minimal training is required to learn how to perform experiments. Essentially this creates two types of users: those that can only run existing experiments and those that can also modify the C++ code to perform experiments of which the VF is not yet capable.

3.2 Flowchart

Figure 3.1 shows a flowchart of the Visual Basic interface. Based on the factory configuration, simulation type, and algorithm that is selected, there are specific parameters to input. The One Factory scenario is the most developed, in that it allows the user to select from more simulation types and algorithms than for the multi-factory scenarios. Adding these capabilities to the multi-factory problems is the subject of future work.

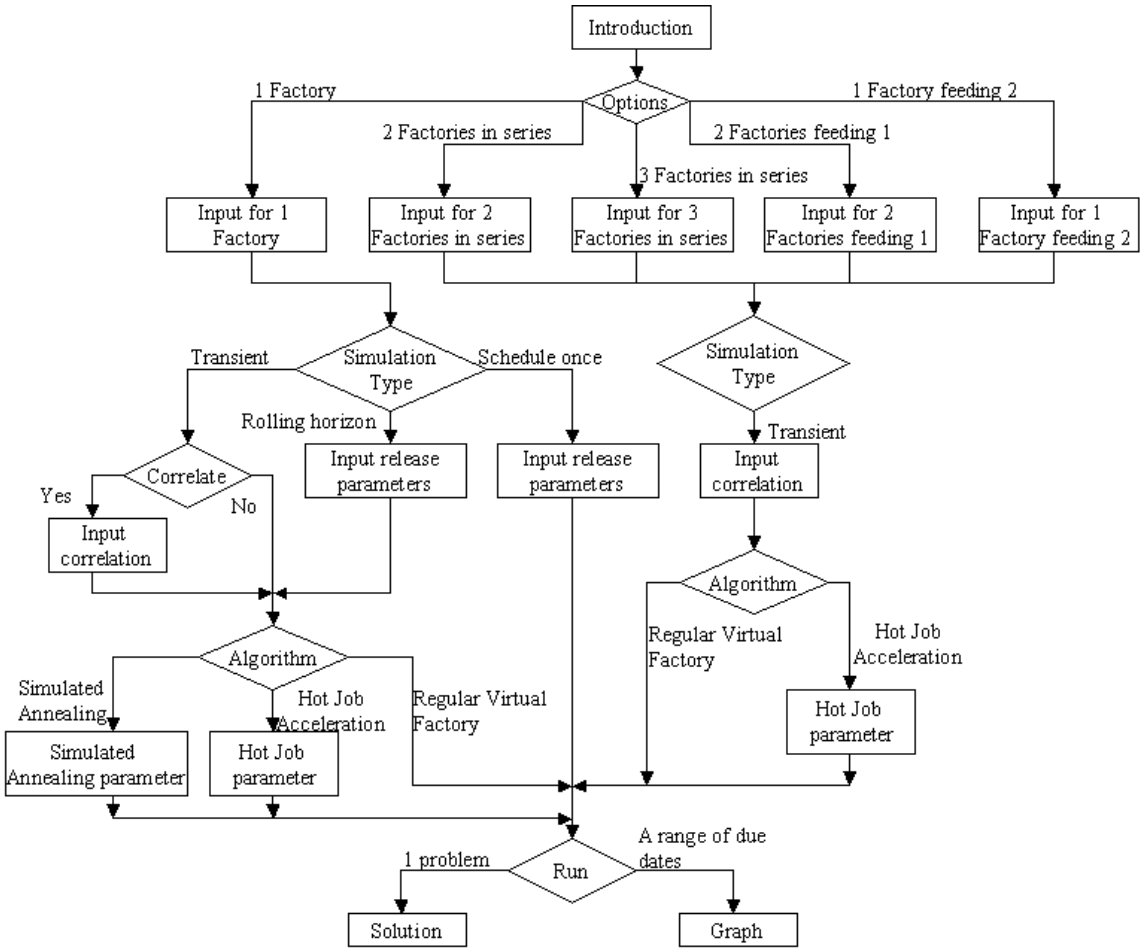

Figure 3.1: Flowchart for Visual Basic Interface

3.3 Factory Configurations

Figure 3.2 is the introduction form of the scheduling system. From here the user proceeds to Figure 3.3, where he or she is given a list of different factory configurations from which to choose.

Figure 3.2: Introduction form

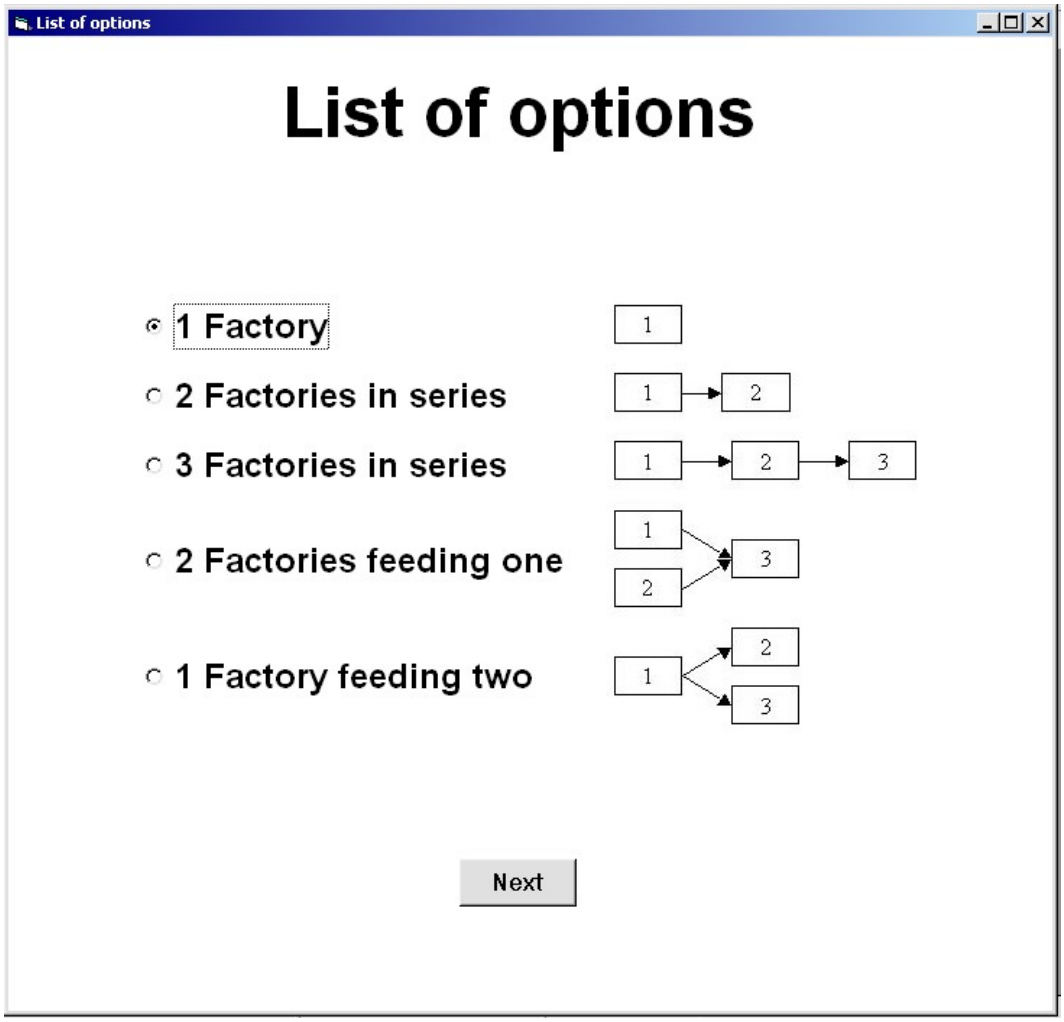

Figure 3.3: List of options

3.3.1 One Factory

The One Factory scenario randomly generates a job shop in which each job visits a subset of the machines in the factory, with the stipulation that each job visits no machine more than once. All other scenarios are generated with this stipulation. The input parameters for the One Factory scenario are shown in Figure 3.4. The processing times in this scenario, as in all others, are set at a default of being uniformly distributed between 1 and 200. These values were chosen since they were used in previous VF experimentation.

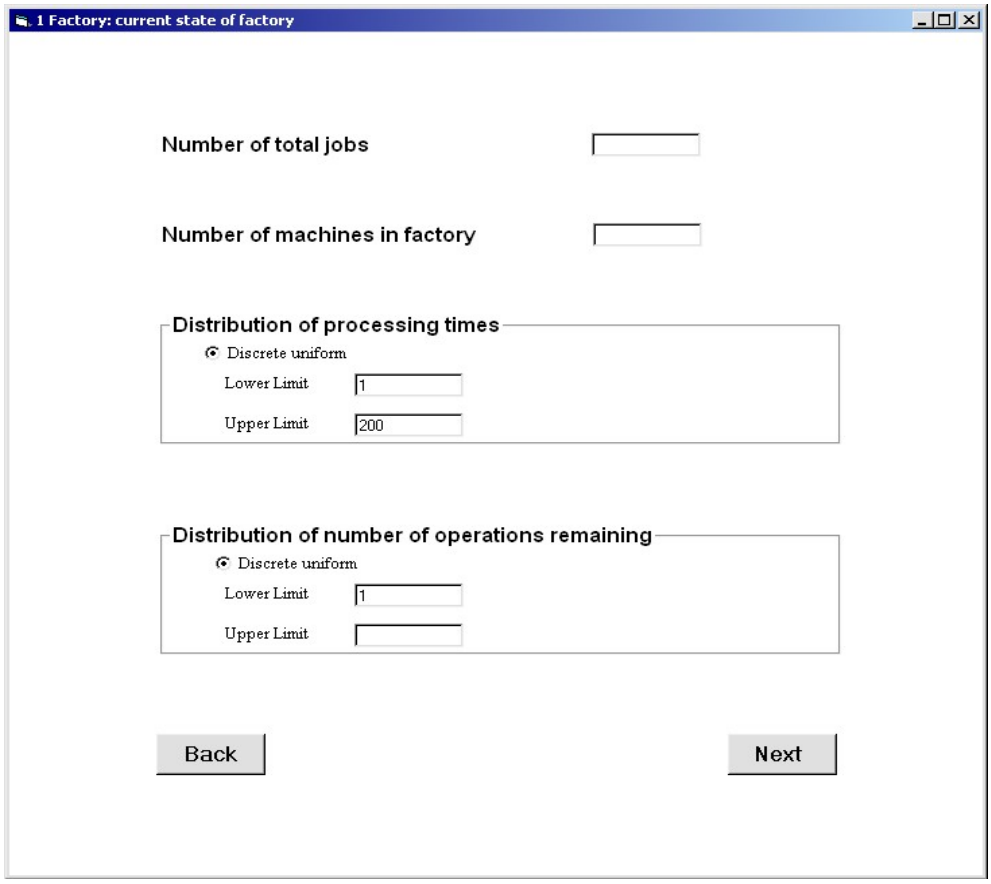

Figure 3.4: One factory parameters

3.3.2 Two Factories in Series

The Two Factories in Series scenario randomly generates a problem in which all jobs starting in Factory 1 will be transported to Factory 2 where they will also be processed there. Jobs starting on the truck or in Factory 2 will only be processed in Factory 2. The input parameters for the Two Factories in Series scenario are shown in Figure 3.5.

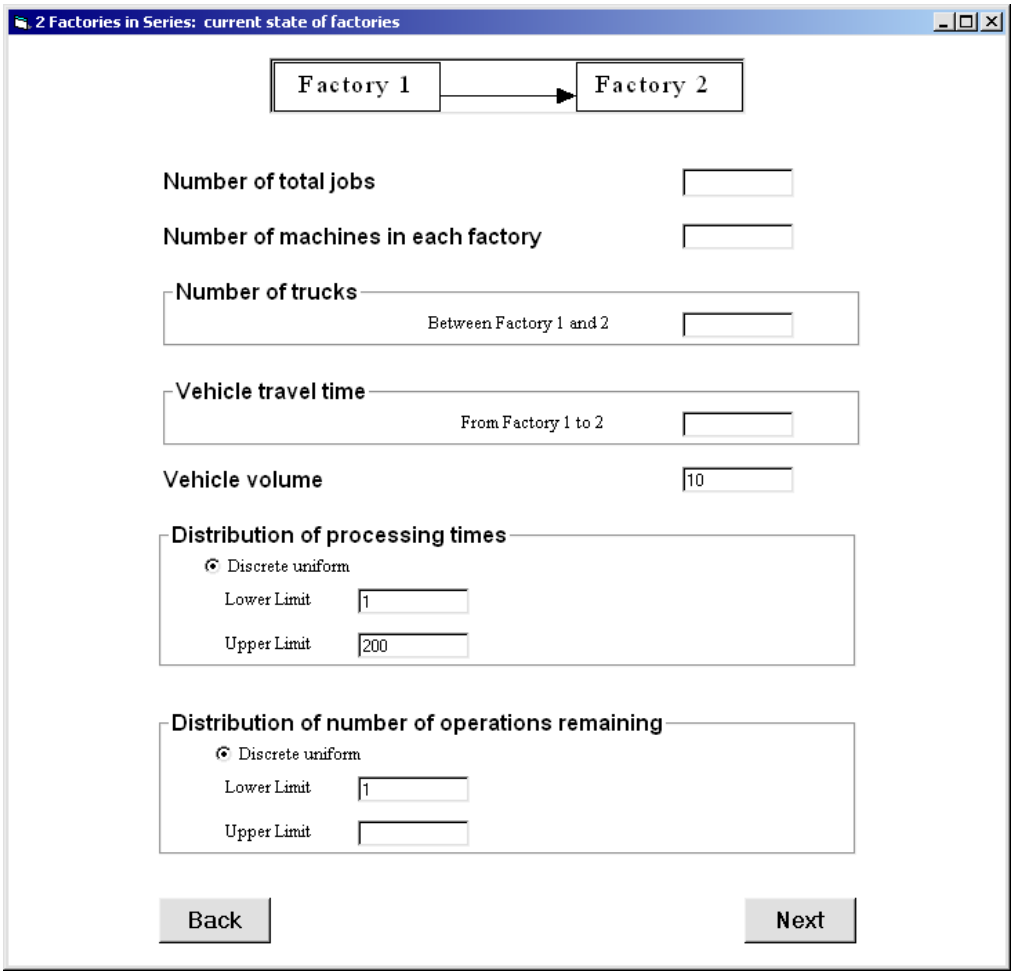

Figure 3.5: Two factories in series parameters

As is true for all multi-factory scenarios in this system, the maximum number of operations in each factory is equal, and truck transportation is considered a machine operation. The number of operations remaining, O_{ps}, for each job is uniformly distributed between 1 and an upper limit, U_L, that must be odd. Let M be the maximum number of operations in each factory. Then, $M = [(U_L - 1) / 2]$. *If* $O_{ps} > M+1$, the job will be processed on $O_{ps}-(M+1)$ machines in Factory 1, transported by truck to Factory 2, and processed on M machines in Factory 2. If $O_{ps}=M+1$, the job will be transported by truck to Factory 2 and processed on M machines. If $O_{ps} < M$, the job will be processed on O_{ps} machines in Factory 2.

The truck volume assumes that each job has unit volume and therefore equals the number of jobs that a single truck can carry from one factory to another at the same time. Truck volume is initialized to 10 because, again, most of the previous VF experiments used this value. Both of these observations hold for trucks in all multi-factory scenarios in this program.

3.3.3 Three Factories in Series

The Three Factories in Series option randomly generates a problem where jobs that begin in Factory 1 are processed there, transported via truck to Factory 2, processed in Factory 2, transported to Factory 3, and processed in Factory 3. Jobs that begin on the truck between Factory 1 and 2 or in Factory 2 are processed in Factory 2, transported to Factory 3 and processed in Factory 3. Jobs that begin on the truck

between Factory 2 and 3 or in Factory 3 are only processed Factory 3. The input parameters for the Three Factories in Series scenario are shown in Figure 3.6.

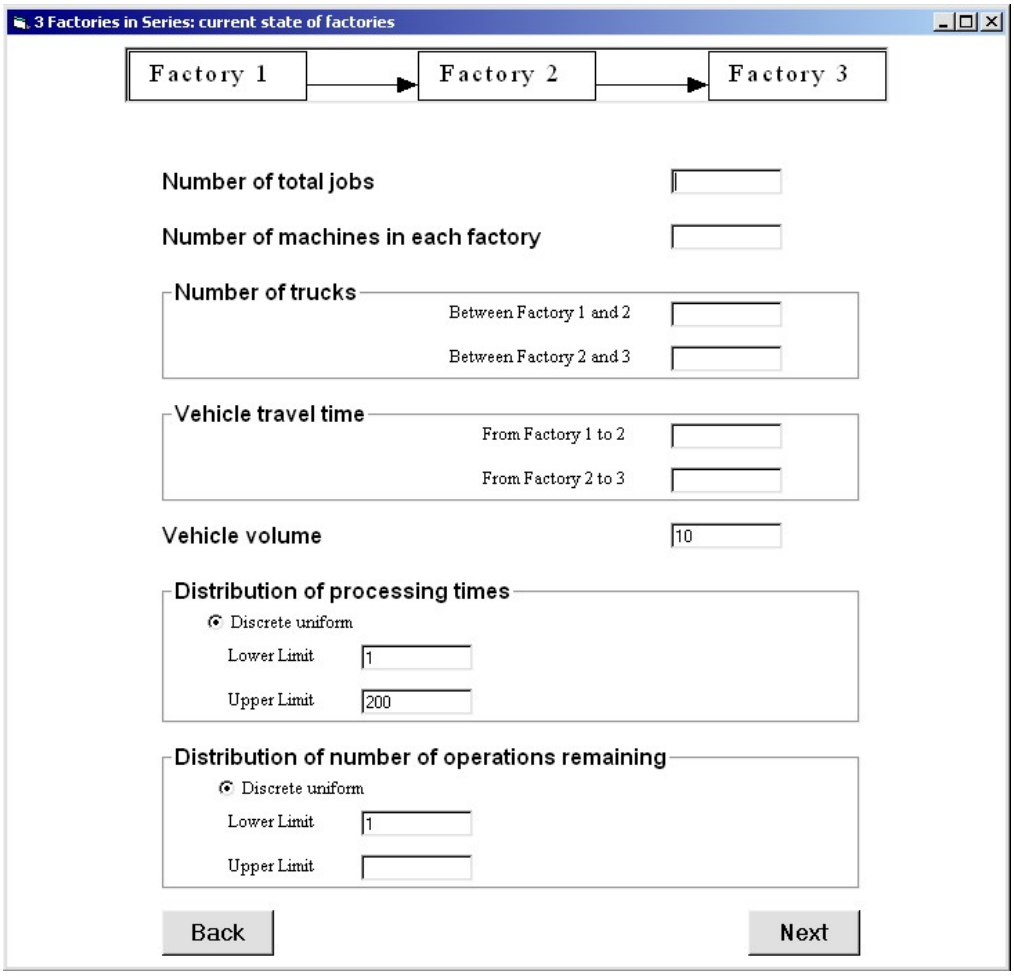

Figure 3.6: Three factories in series parameters

Let $M=[(U_L-2)/3]$. Consequently, U_L-2 must be a multiple of 3. *If* $O_{ps} > 2M+2$, the job is processed on $O_{ps} -(2M+2)$ machines in Factory 1, transported by truck to Factory 2, processed on M machines in Factory 2, transported by truck to Factory 3, and processed on M machines in Factory 3. If $O_{ps} =2M+2$, the job is transported by truck to Factory 2, processed on M machines in Factory 2, transported by truck to

Factory 3, and processed on M machines. If $(M+1) < O_{ps} < (2M+2)$, the job is processed on $O_{ps} - (M+1)$ machines in Factory 2, transported by truck to Factory 3, and processed on M machines in Factory 3. If $O_{ps} = M+1$, the job is transported by truck to Factory 3 and processed on M machines. If $O_{ps} \leq M$, the job is processed on O_{ps} machines in Factory 3.

3.3.4 Two Factories Feeding One

The Two Factories Feeding One scenario randomly generates a problem in which a job from Factory 1 and a job from Factory 2 (either starting in these factories or on a truck) are assembled together in Factory 3. A specific job from Factory 1 must be assembled with a specific job from Factory 2. This pairing is determined upon problem generation. The input parameters for the Two Factories Feeding One scenario can be seen in Figure 3.7.

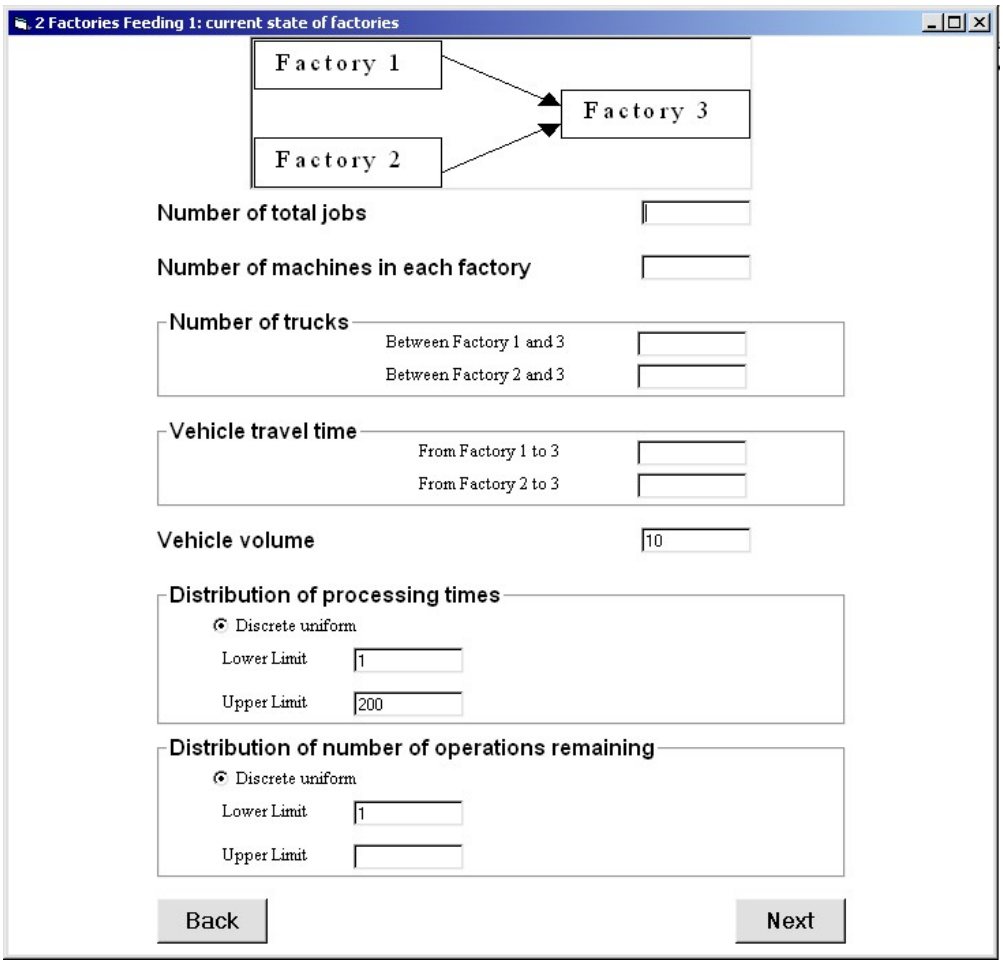

Figure 3.7: Two factories feeding one parameters

Let $M = (U_L - 1) / 2$. Thus, U_L must be an odd number. If $O_{ps} > M+1$, 3 jobs are generated. The first job will be processed on $O_{ps} - (M+1)$ machines in Factory 1 and transported by truck to Factory 3. The second job will be processed on $O_{ps} - (M+1)$ machines in Factory 2 and transported by Truck to Factory 3. The third job will be the assembly of 1 and 2 and it will be processed on M machines in Factory 3. If $O_{ps} = M+1$, 3 jobs are also generated. The first job will be transported by truck from Factory 1 to Factory 3. The second job will be transported by truck from Factory 2 to

Factory 3. The third job will be the assembly of 1 and 2 and will be processed on M machines in Factory 3. If $O_{ps} < M+1$, 1 job is generated, and it will be processed on M machines in Factory 3.

3.3.5 One Factory Feeding Two

The One Factory Feeding Two scenario randomly generates a problem where jobs staring in Factory 1 are split into two jobs, one of which is further processed in Factory 2 and the other in Factory 3. The input parameters for the One Factory Feeding Two scenario are shown in Figure 3.8.

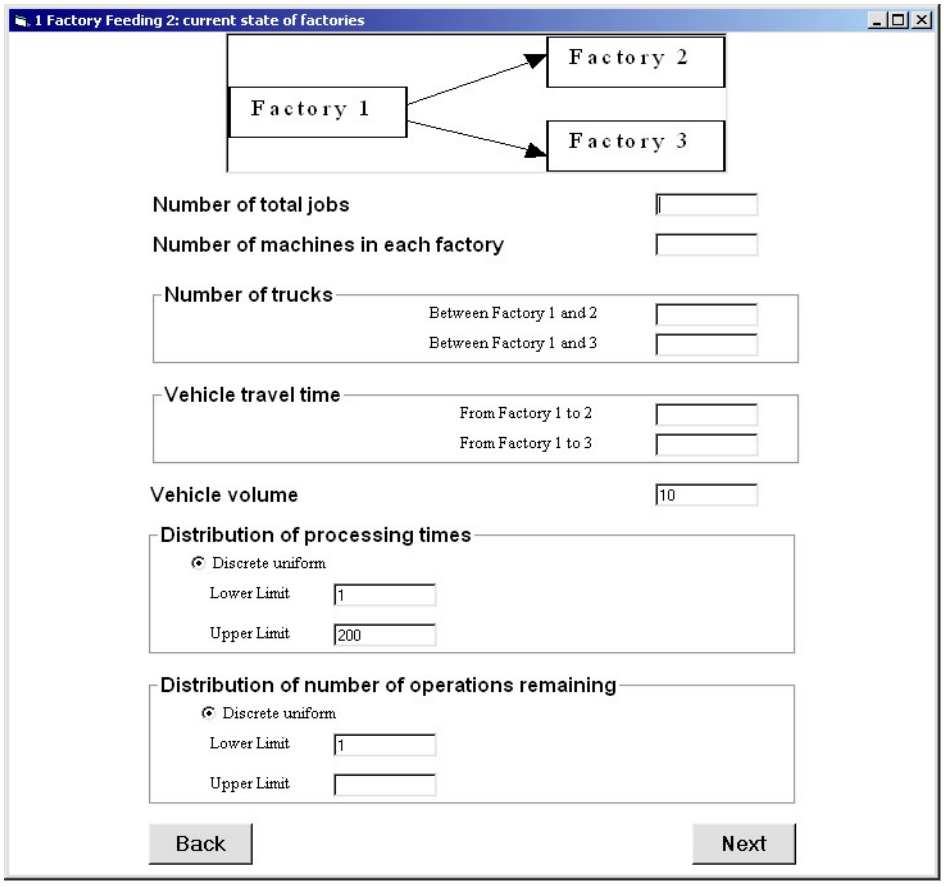

Figure 3.8: One factory feeding two parameters

Let $M = (U_L - 1) / 2$. Consequently, U_L must be an odd number. If $O_{ps} > (M+1)$, 3 jobs will be generated. The first job will be processed on $O_{ps} - (M+1)$ machines in Factory 1. This job is split into the second and third job. The second job will be transported by truck to Factory 2 and processed on M machines. The third job will be transported by truck to Factory 3 and processed on M machines. If $O_{ps} = (M+1)$, 2 jobs will be generated. The first job will be transported by truck to Factory 2 and processed on M machines. The second job will be transported to Factory 3 and processed on M machines. If $O_{ps} < M$, 2 jobs will be generated. The first and second will be processed on M machines in Factory 2 and Factory 3, respectively.

3.4 Simulation Type

The choices of simulation types can be seen in Figure 3.9.

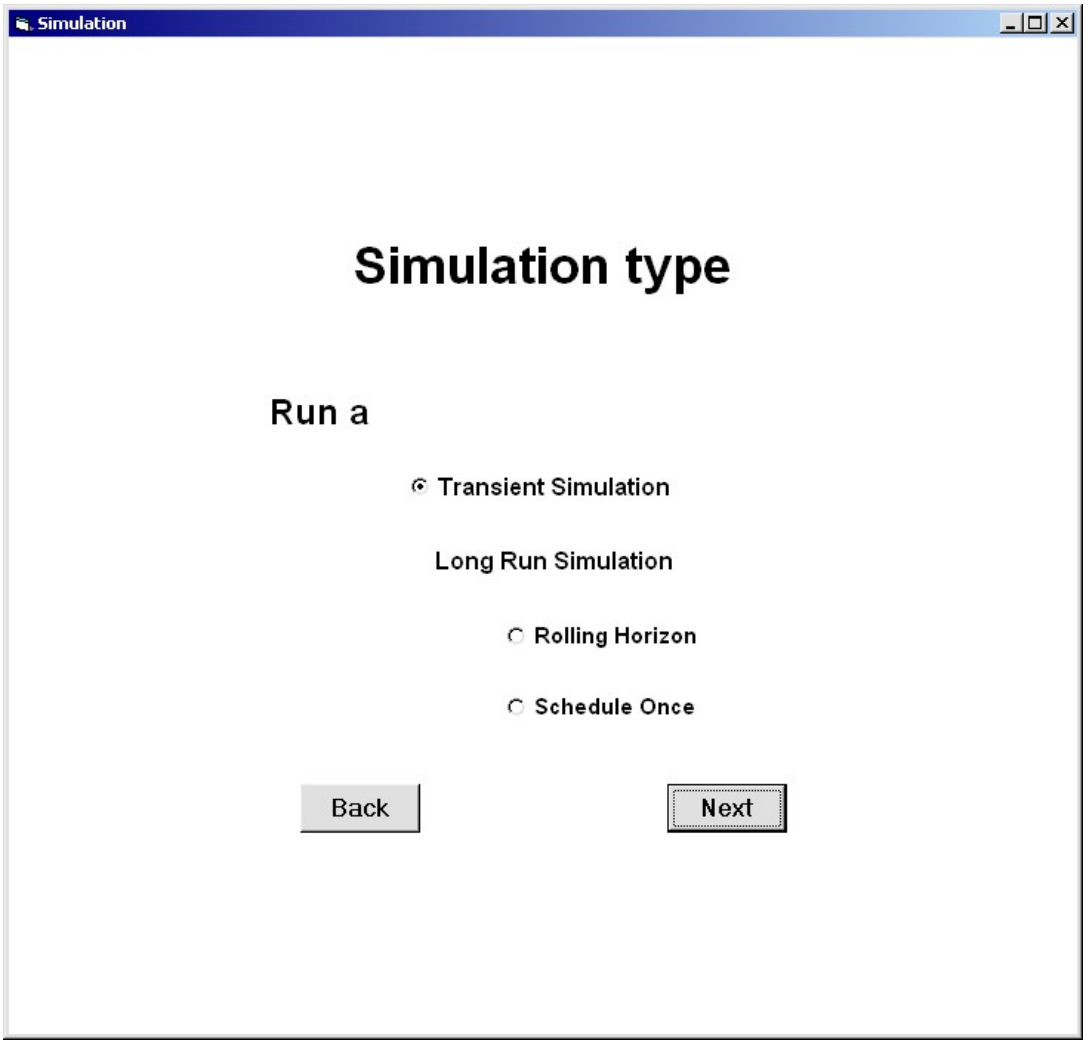

Figure 3.9: Simulation type

3.4.1 Transient simulation

A transient scheduling simulation is defined in this thesis to be a method in which a factory download is randomly generated, and the factory is simulated until all

the jobs are complete, and the factory is empty. This reflects the traditional method of evaluating job shop scheduling algorithms.

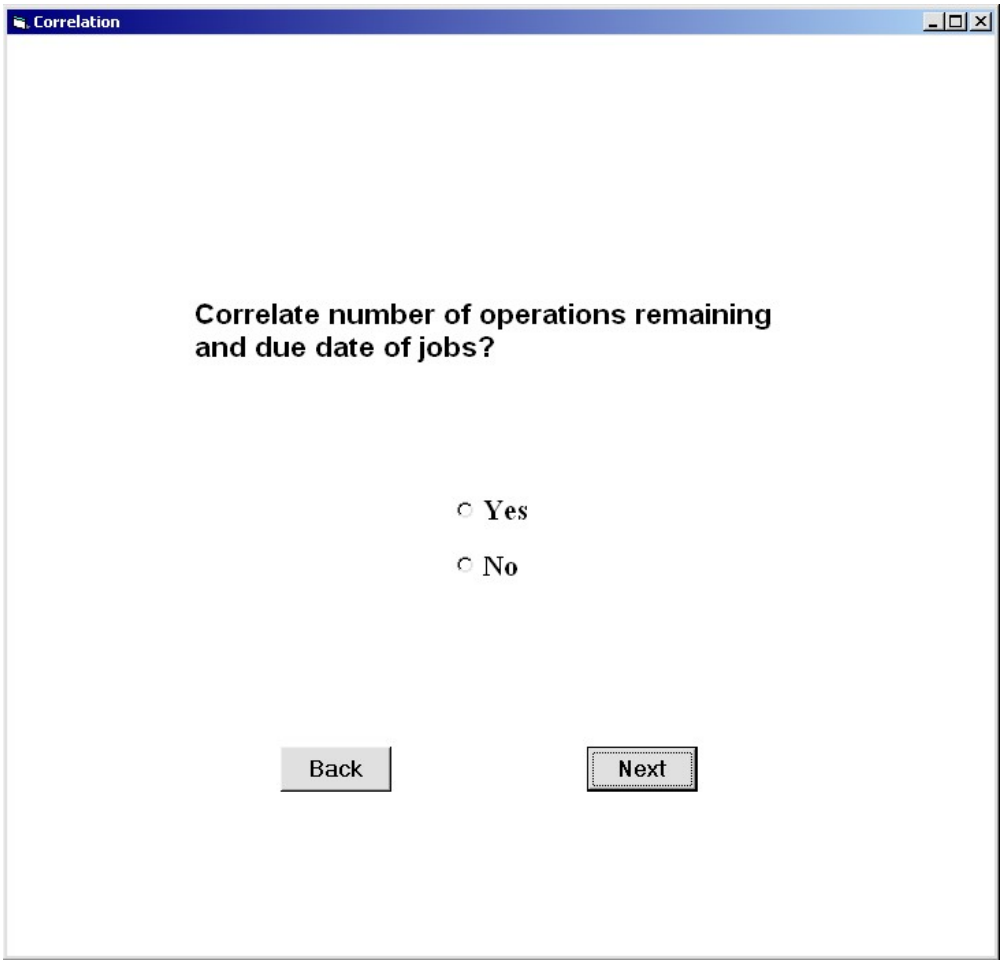

Figure 3.10: Correlation

For the transient simulation, the user is given the option of correlating the number of remaining operations and the due date, as seen in Figure 3.10. This gives the problem more realistic characteristics. One would expect that the jobs that are almost finished have imminent due dates, while jobs that have just started processing have due dates farther in the future. If the user decides to choose to correlate the

number of operations remaining with the due date, then Figure 3.11 allows him or her to input the correlation. This correlation should be nonnegative because in a shop that is well run, the average correlation between the number of operations remaining and the due date should not be negative. The method used to correlate the number of operations remaining with the due date is that of Cario et al [7, 8]. In this method, the correlation must strictly be less than one.

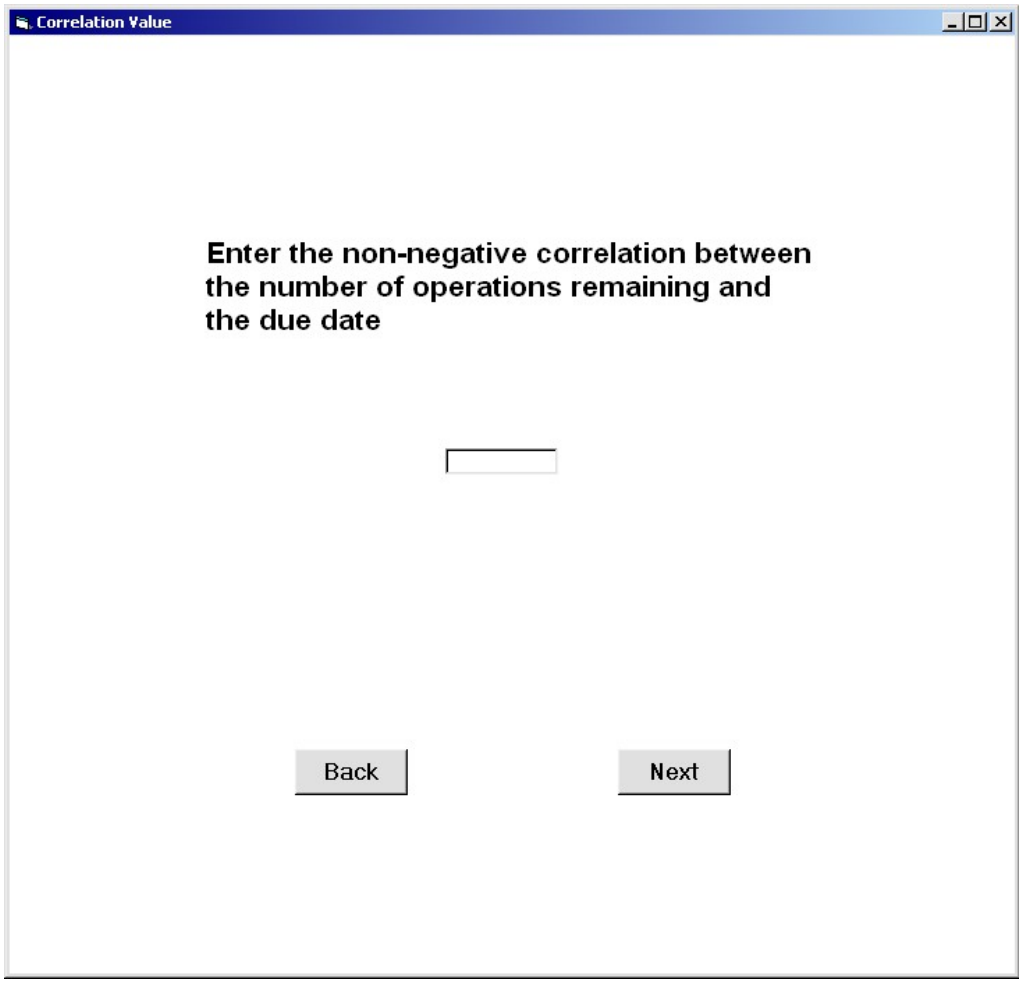

Figure 3.11: Correlation value

3.4.2 Long Run Simulation

A long run scheduling simulation is defined in this thesis as a simulation, which not only simulates the random factory download but also releases jobs into the system on a periodic basis and terminates after a large number of periods. This is a much better approximation of what happens in industry as compared to the transient simulation (particularly in the rolling horizon case), because in industry, factories are not often emptied.

The additional parameters for a long run simulation are found in Figure 3.12. These parameters relate to those jobs that are not currently in the factory download, but, instead, will be released later. The number of warm-up days refers to the number of days the simulation will be run before clearing the statistics. The purpose of this is to allow the user to collect only steady state information. The effect of changing the number of days in warm-up is discussed in detail in Section 4.4.3.

3.4.2.1 Rolling Horizon Simulation

Background on rolling horizon scheduling was discussed in Section 2.3. A rolling horizon scheduling simulation will be defined in this thesis as a long run simulation in which in each period jobs are released, the current status of the factory is downloaded, scheduling is performed, and the best schedule is implemented until more jobs are released and the factory is rescheduled. A more complete description of this procedure applied to the VF as well as computational experiments are located in Section 4.2.

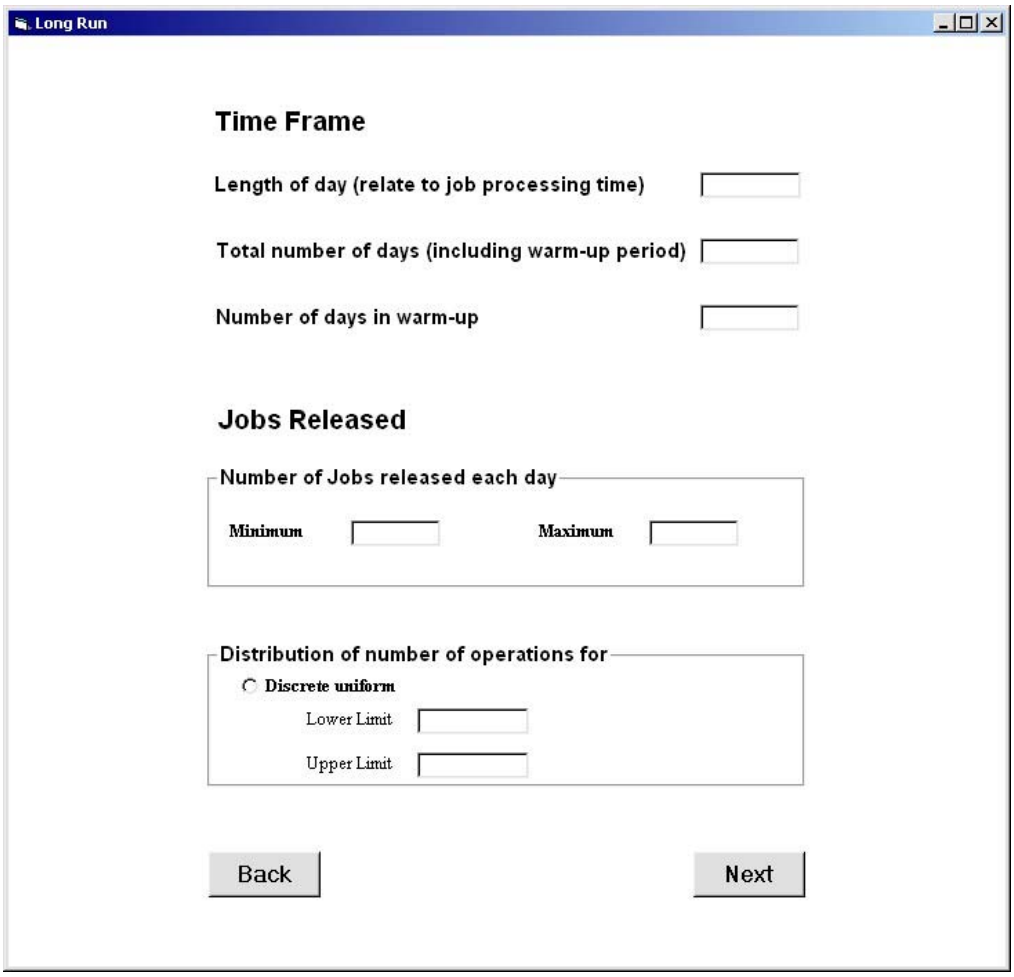

Figure 3.12: Long run parameters

3.4.2.2 Schedule Once

This option was created to test whether the VF performed better on a rolling horizon basis or in one large simulation where all job releases in the time horizon are known in advance. More information on this procedure is found in Section 4.4.7.

3.5 Algorithm

The choice of algorithms is shown in Figure 3.13.

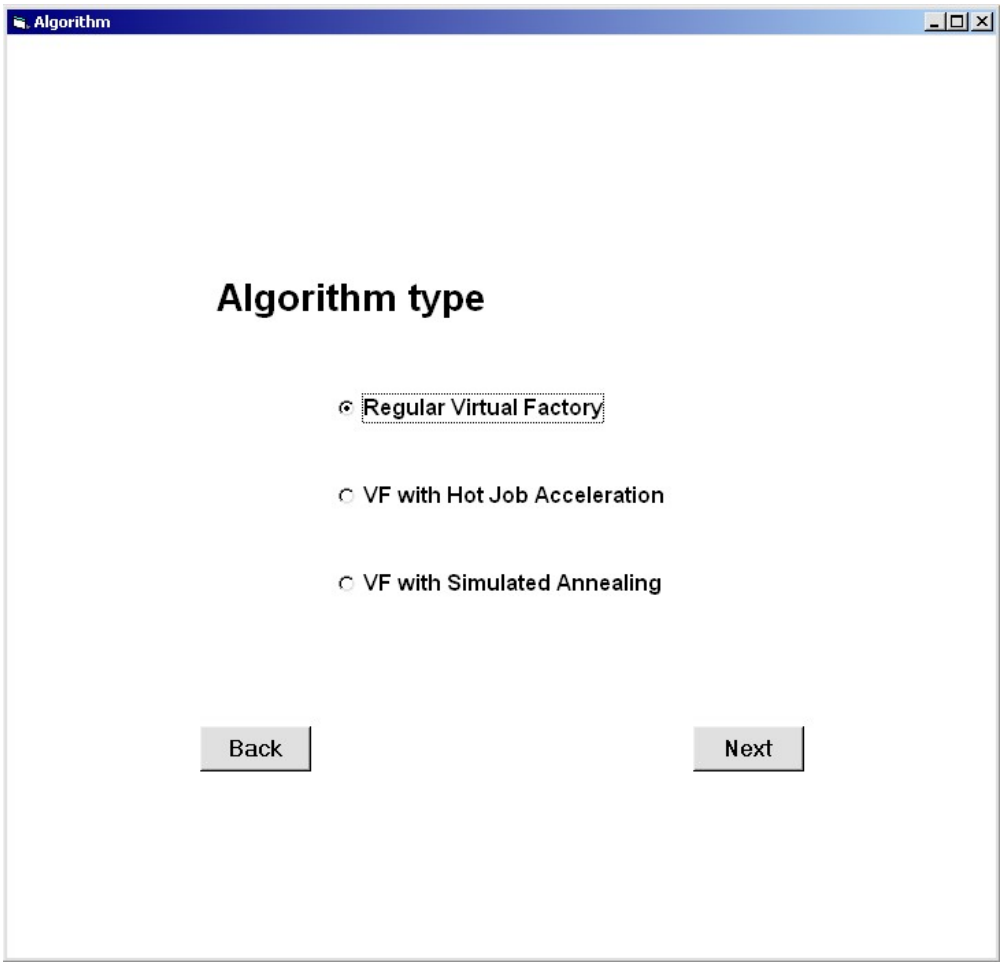

Figure 3.13: Type of algorithm

3.5.1 Regular Virtual Factory

The regular VF is the original algorithm explained in Section 2.1.

3.5.2 Hot Job Acceleration

In this algorithm, critical or "hot" jobs, whose lateness is equal to or close to L_{max}, are given priority over other jobs. This is accomplished by inserting idle time

into a machine schedule if a non-critical job would be in process on a machine when a critical job arrives. The critical job begins processing immediately upon arrival at the machine, while the non-critical job is kept in the queue until the critical job is complete. This method is explained in more detail in Section 2.2.1. Computational results for using this procedure in a rolling horizon setting can be found in Section 4.4.8.

The parameters for this option can be seen in Figure 3.14. When the number of consecutive worse solutions occurs or the maximum number of hot jobs is explored, the algorithm will terminate.

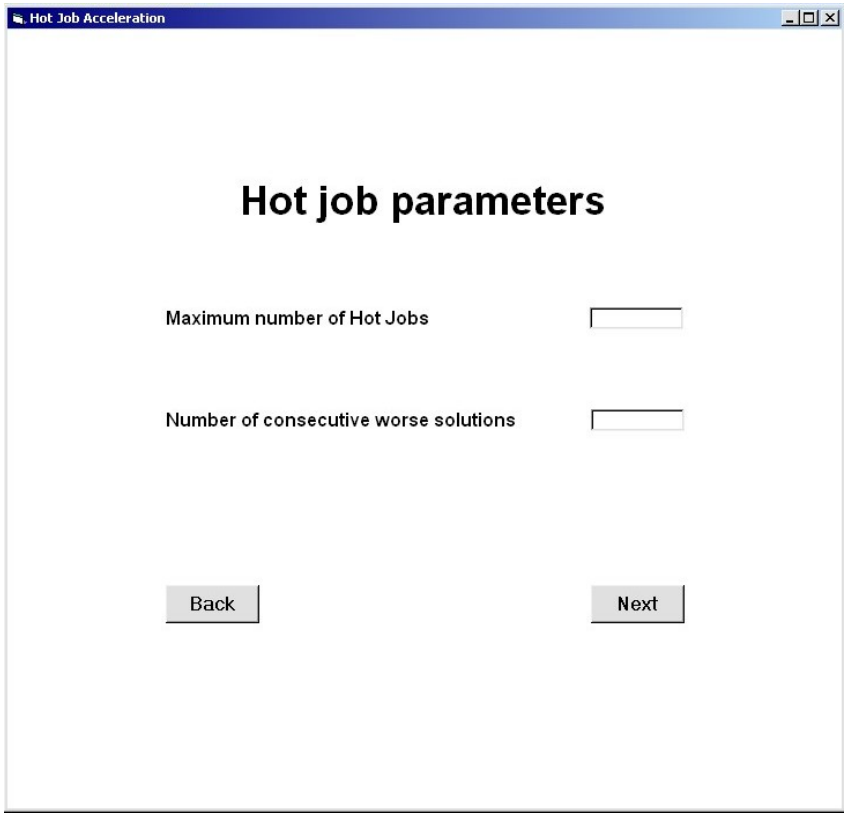

Figure 3.14: Hot job parameters

3.5.3 Simulated Annealing

This option uses the VF to obtain an initial solution and then uses a simulated annealing algorithm to investigate switching jobs on the critical path. This procedure was explained in Section 2.2.5. Computation results for using this procedure in a rolling horizon setting can be found in Section 4.4.9.

The parameters for this algorithm are found in Figure 3.15. Note that a time limit is used to control how long the simulated annealing algorithm is run. If this procedure is being run for a transient simulation, the hot job parameters also need to be entered. The reason for this is that the simulated annealing procedure should be started with the best-known solution. In the case of the transient simulation, this is the hot job solution. As will be shown in Section 4.4.8, hot job acceleration does not work particularly well in the rolling horizons situation. Therefore, the original VF algorithm usually yields the best solution. Consequently, the hot job parameters are hidden when the user reaches Figure 3.15, if he or she has selected the rolling horizon option.

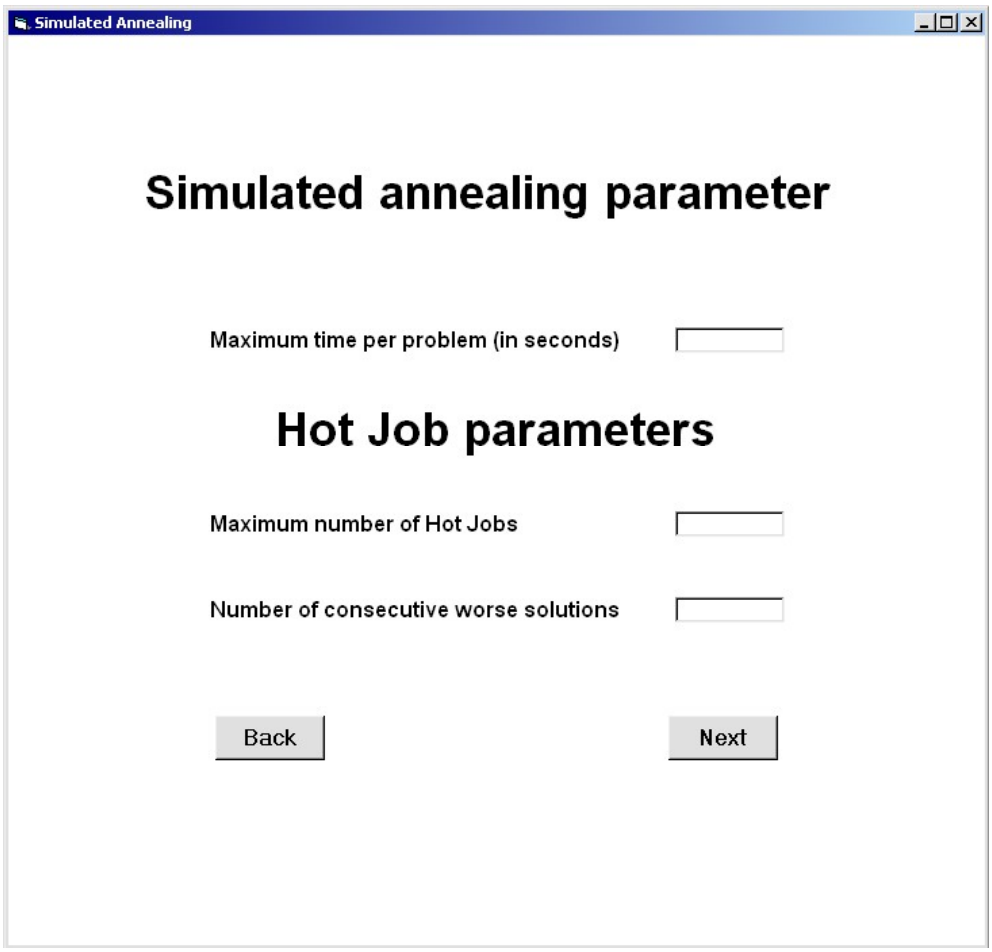

Figure 3.15: Simulated annealing parameter

3.6 Running Problems

As seen in Figure 3.16, there are two different options of running the simulation. The first option is to run a single experiment with a specific due date range. Random number stream 1 is used by default. An example of the output of this option is shown in Figure 3.17. The random numbers are generated by LCG.

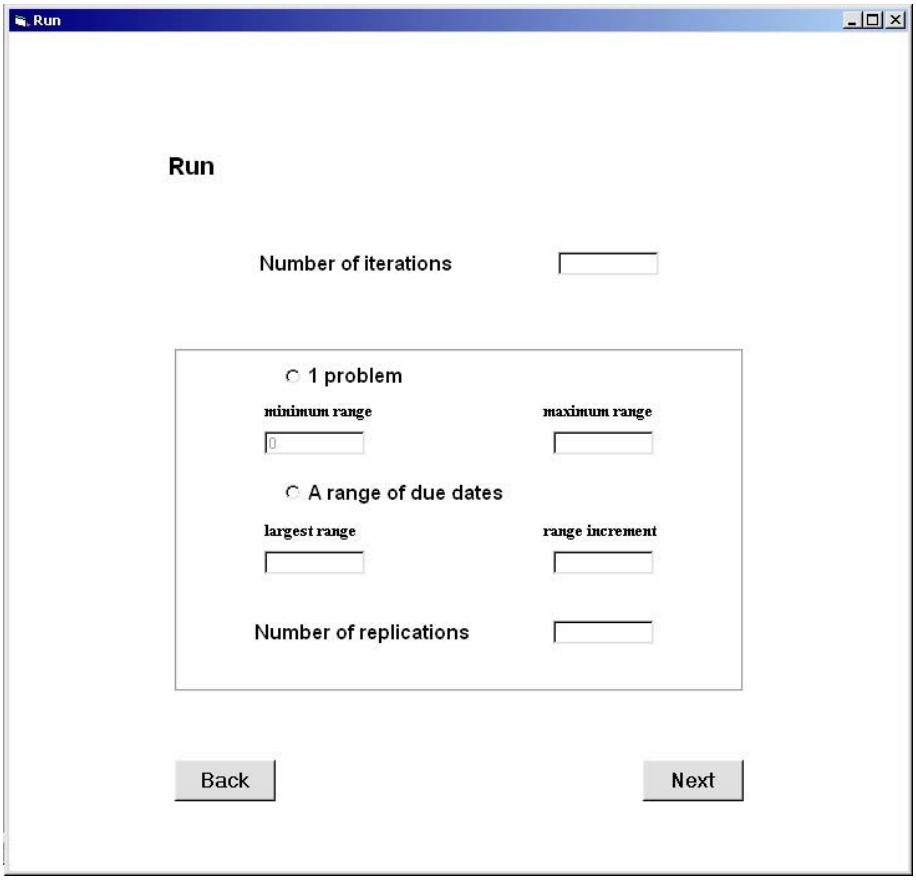

Figure 3.16: Run screen

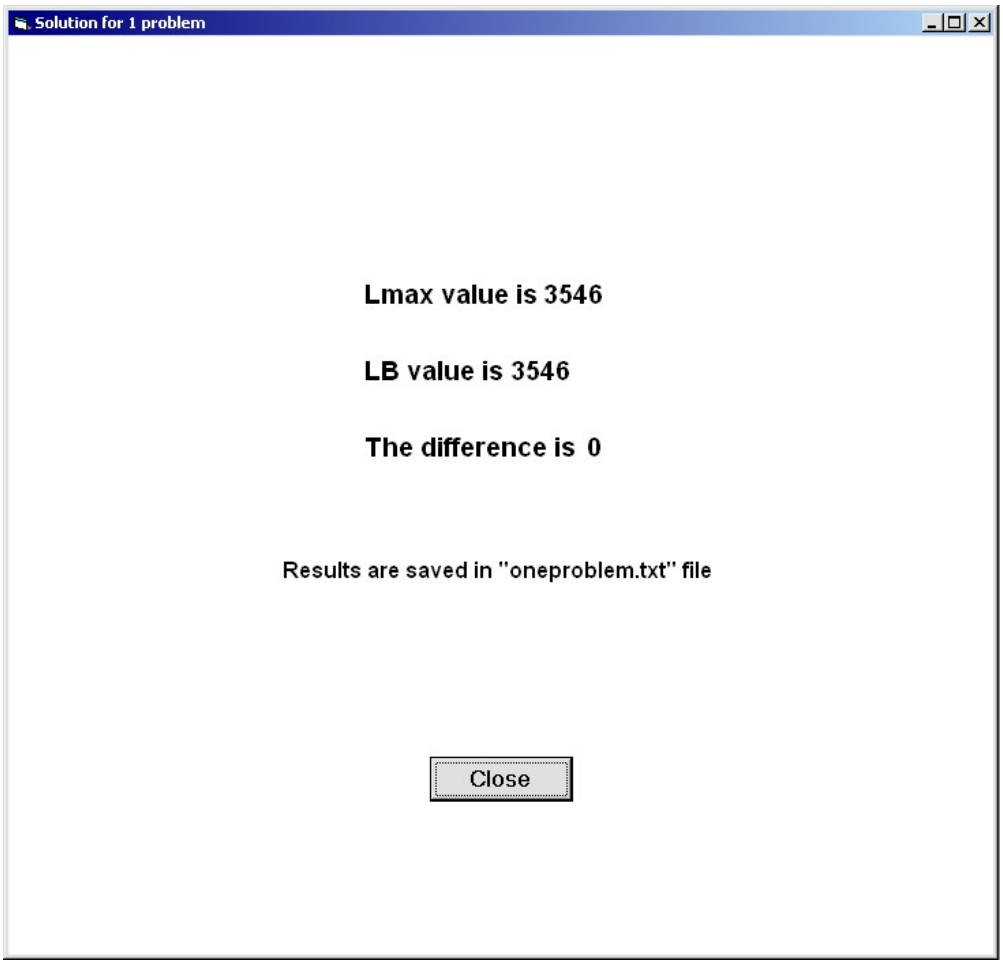

Figure 3.17: 1 problem output screen

The second option is to run multiple simulations, varying the due date range between 0 and some maximum value in fixed increments. The output for this option is a graph, an example of which can be seen in Figure 3.18. The system automatically writes the results to an excel spreadsheet as well as generates the graph from the results. The random number streams used are the first *R*, where *R* equals the number of replications.

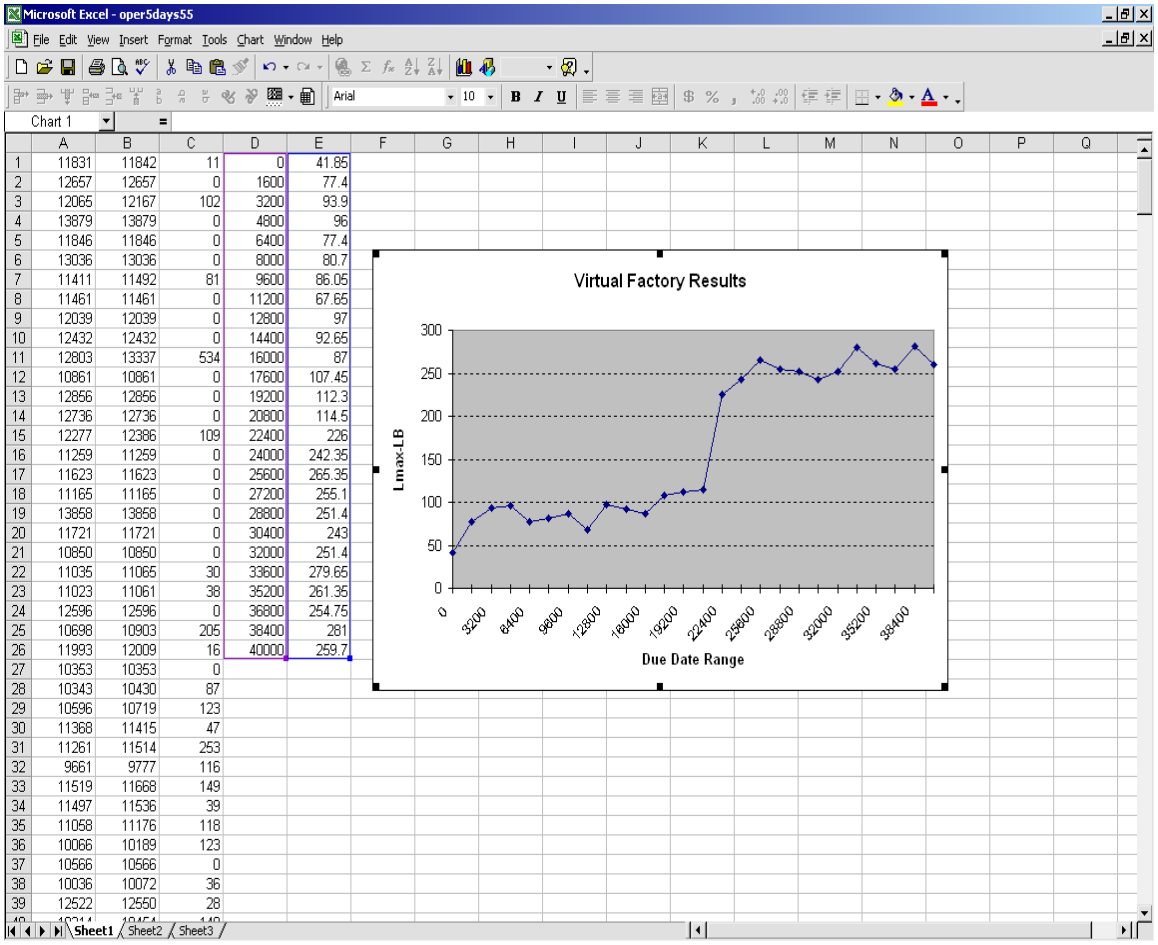

Figure 3.18: Graph for range of due dates-example

Chapter 4

Rolling Horizon Scheduling

4.1 Introduction

Until now, the Virtual Factory has been tested only under transient settings by running a plant until all operations are complete and the plant is empty. However, what happens in an actual industrial setting is different. In industry, jobs are released and scheduling is performed on a regular basis, perhaps every day. Before new job arrivals can be scheduled, they are entered into the plant's ERP or MRP system. Then, the current status of the plant is downloaded. Scheduling is performed, and the best schedule is implemented until the next time the plant is scheduled. This process is repeated over and over again. To more accurately determine the performance of the Virtual Factory in real scenarios, it will be tested on a rolling horizon basis.

The following definitions are required for this chapter:

t - Current time in days
LB - Lower bound
r_j - Release time of job j
c_j - Completion time of job j
d_j - Due date of job j
N - Total number of jobs
N_s - Total number of jobs starting in factory
M - Total number of machines
U_L - Upper limit of uniform distribution for number of operations
J_R - Number of jobs released each day
R_O - Number of operations for jobs released
D_L - Length of a day
T - Total number of days
w - Number of days in warm-up period

WIP - Work in process (number of days)
i - Number of iterations

4.2 Rolling Horizon Procedure

4.2.1 Scheduling algorithm

The algorithm for the rolling horizon scheduling procedure is given as follows:

1. Initialize $t = 0$

    1.1. If $t = w + 1$, compute LB

    1.2. Release jobs whose $r_j = t$

    1.3. Run the Virtual Factory i iterations

    1.4. Implement the first day of the best schedule

    1.5. $t = t + 1$

    1.6. Continue from 1.1. until $t = T$

2. Run the remainder of the best schedule of the last day (T) until all jobs are finished

3. Initialize j =1

    3.1. If $c_j > w$, determine if job j is the L_{max} job

    3.2. $j = j + 1$

    3.3. Continue from 3.1. until $j = N$

Step 1 initializes the beginning of the first day as time 0. If in step 1.1 the time is one day past the warm-up period, the lower bound is computed. In step 1.2, the jobs with release time equal to the current time enter the factory. No jobs are released on

the beginning of the first day since these jobs are assumed to be already in the factory. Step 1.3 runs the original VF procedure, usually for 100 iterations in the experimentation in this thesis. In step 1.4, the first day of the best schedule is implemented. The rest of the schedule is discarded, except on the last day. At the end of the day, there may be jobs that are still in process. Each of these jobs is put back in the machine's queue, and the job's processing time is set equal to the remaining processing time. Steps 1.5 and 1.6 ensure that steps 1.1 through 1.4 are run for each day until the total number of days is reached. In step 2, the best schedule of the last day is run until all jobs are finished. This ensures that the scheduling procedure does not sacrifice the remaining jobs in the factory to yield a good schedule for the jobs that complete processing since all jobs finish. Step 3 initializes the counter, j, equal to the first job. In Step 3.1, only jobs that are completed after the warm-up period are included in the L_{max} calculation to eliminate transient effects dependent on initial factory conditions. Steps 3.2 and 3.3 ensure that the lateness for each job completed after the warm-up period is compared to the current maximum lateness.

4.2.2 Lower Bound

The lower bound for the rolling horizon schedule is computed after the warm-up period. The *LB* calculation includes both jobs that are currently in the factory after the warm-up period, with their remaining operations and processing times, and also those jobs that will be released later, during the complete horizon of the simulation. The *LB* is computed in the same manner as for the original VF. Therefore, even

though there are multiple runs of the VF engine for the rolling horizon scheduling procedure, there is only one *LB* calculation.

4.3 Problem Generation

For testing the Virtual Factory on a rolling horizon basis, two different problems were generated, a 5 operation problem and a 7 operation problem. The parameters for these problems are given below in Table 4.1.

Table 4.1: 5 Operation and 7 operation problem parameters

	N_s	M	U_L	J_R	R_O	D_L	T	w	WIP	i
5 Operation Problem	1258	50	5	151	5	1600	100	10	5	100
7 Operation Problem	2082	75	7	165	7	1600	100	10	7	100

The values of J_R, N_s, and w were determined based on the other parameters of the problems. To compute J_R, the number of jobs that balances the input into the factory with the output from the factory needed to be found. This was approximated by dividing the average number of operations that can be processed daily by the amount of *WIP* in the factory. To find the average number of operations that can be processed each day in the factory, the number of machines, M, was multiplied by the average number of operations that a single machine can process in a day, $\overline{Mops}$. $\overline{Mops}$ can be computed by dividing the day length, D_L, by the average processing time, $\overline{P}$. Consequently,

$$\overline{Mops} = \frac{DL}{\overline{P}}$$

and

$$JR \approx \frac{(M)(\overline{Mops})}{WIP}.$$

Since the processing times for both problems are uniformly distributed between 1 and 200, $\overline{P} \approx 100$ and thus $\overline{Mops} \approx 1600/100=16$. (Assuming an 8 hour work day, the average processing time is 0.5 hours). Therefore, for the 5 operation problem, $J_R \approx (50)(16)/5=160$, and for the 7 operation problem, $J_R \approx (75)(16)/7=170$. These values tend to overestimate J_R since they assume that there is never any idle time on the machines. Therefore, experimentation was performed to determine the actual values of J_R, starting with the computed values. J_R was found to be 151 for the 5 operation problem and 165 for the 7 operation problem.

N_s was computed to achieve the desired amount of *WIP*. Since the problems have been designed so that factory input is approximately equal to the factory output, the number of operations that will be completed each day is approximately $(J_R)(R_O)$. If *WIP* days of work in process is desired, then the total number of operations that should start in the factory is $(J_R)(R_O)(WIP)$. Each job that starts in the factory has an average of $\overline{Ops}$ operations, where $\overline{Ops} = (U_L+1)/2$. Consequently,

$$N_s \approx \frac{(JR)(RO)(WIP)}{\overline{Ops}}$$

jobs should start in the factory. For the 5 operation problem, $N_s \approx [(151)(5)(5)]/3 \approx 1258$. For the 7 operation problem, the computed J_R was mistakenly used to calculate N_s. Therefore for the 7 operation problem, $N_s \approx [(170)(7)(7)]/4 \approx 2082$. This

is just slightly higher than the value that would have been obtained if the actual J_R was used and should not significantly effect results.

The length of the warm-up period was chosen to eliminate potential transient effects caused by the initial jobs in the factory. The warm-up period in days, *w*, was set equal to 10 since this is significantly larger than the *WIP* in either of the problems.

4.4 Experimentation

Each experiment was run with a maximum due date range of 25 days. A due date range, *DDR*, means that each job, *j*, is randomly generated a discrete uniform due date between r_j and $r_j + DDR$, where $r_j=0$ for jobs initially in the factory. Note that all jobs are released at the beginning of a day, whereas the due date for a job could occur at any time during the day. For each due date range, 20 replications were run and the average difference between L_{max} and *LB* was calculated. Recall that this difference is the maximum by which the simulation solution could exceed the optimal solution. A positive difference between L_{max} and *LB* could be the result of a non-optimal schedule, a weak *LB*, or a combination of both.

4.4.1 Base Cases

Results of the 5 operation problem can be seen in Figure 4.1. The average L_{max} - *LB* is approximately in the range of 0.9 to 0.18 days. For the first 14 due date ranges, the average difference does not exceed 0.15 days. There is a slight increase in the differences for due date ranges beyond 13 days. These differences are quite small considering that 90 days of factory performance were included in these statistics, with

the lateness of over (90)(151) = 13,590 jobs taken into account. This indicates that the scheduling procedure is performing well.

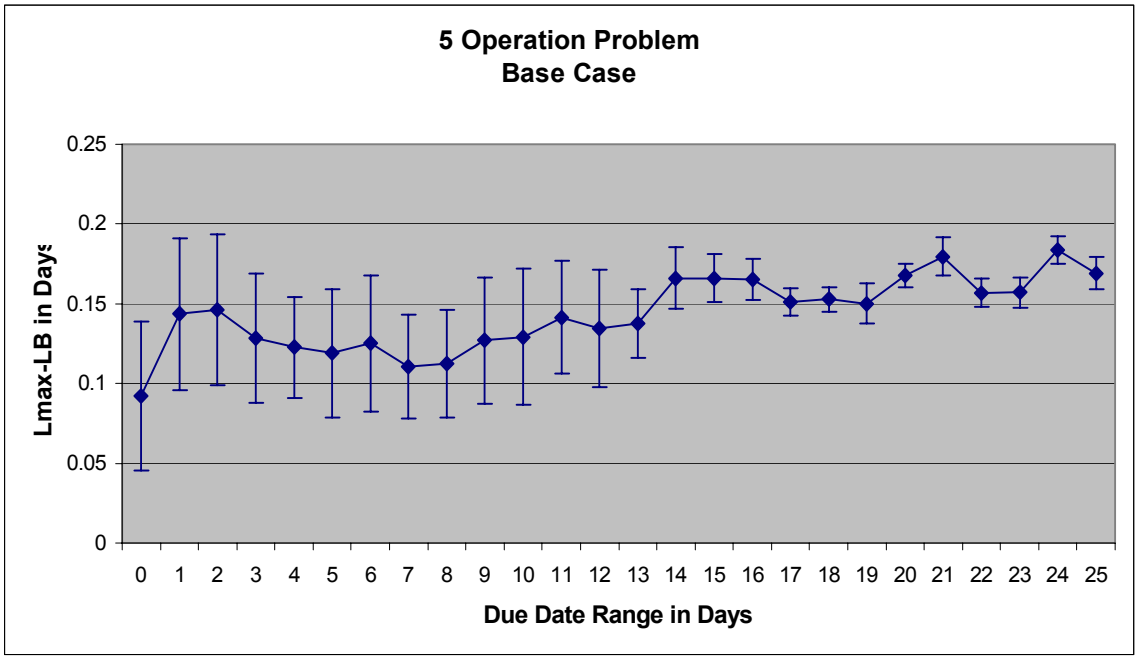

Figure 4.1: 5 operation problem-base case

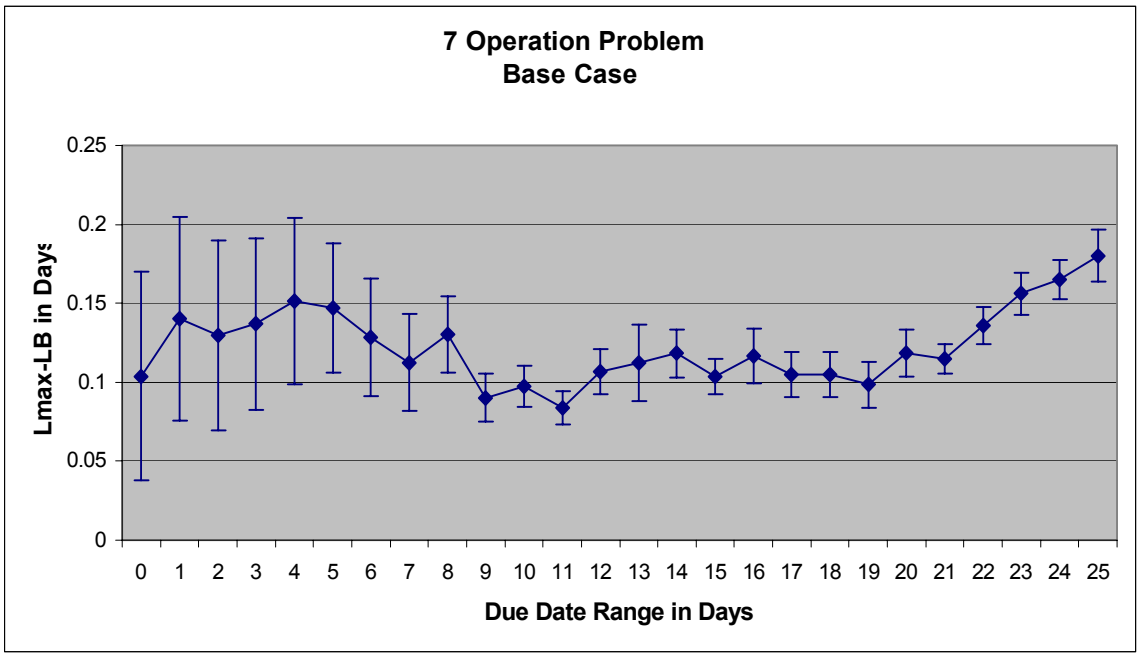

Figure 4.2: 7 operation problem-base case

Figure 4.2 shows the results for the 7 operation problem. The average L_{max}- LB has slight variations until a due date range of 22 days. Then it increases gradually. The magnitudes of L_{max}- LB are similar to those in the 5 operation problem.

4.4.2 Varying the Number of Days

To determine the effect of the total number of days that are scheduled on the quality of the scheduling solutions, each problem was run for 55 days and 190 days with the same 10 day warm-up. This allows the scheduling solutions to be observed when the total number of days after the warm-up period is half as many and twice as many as in the base cases.

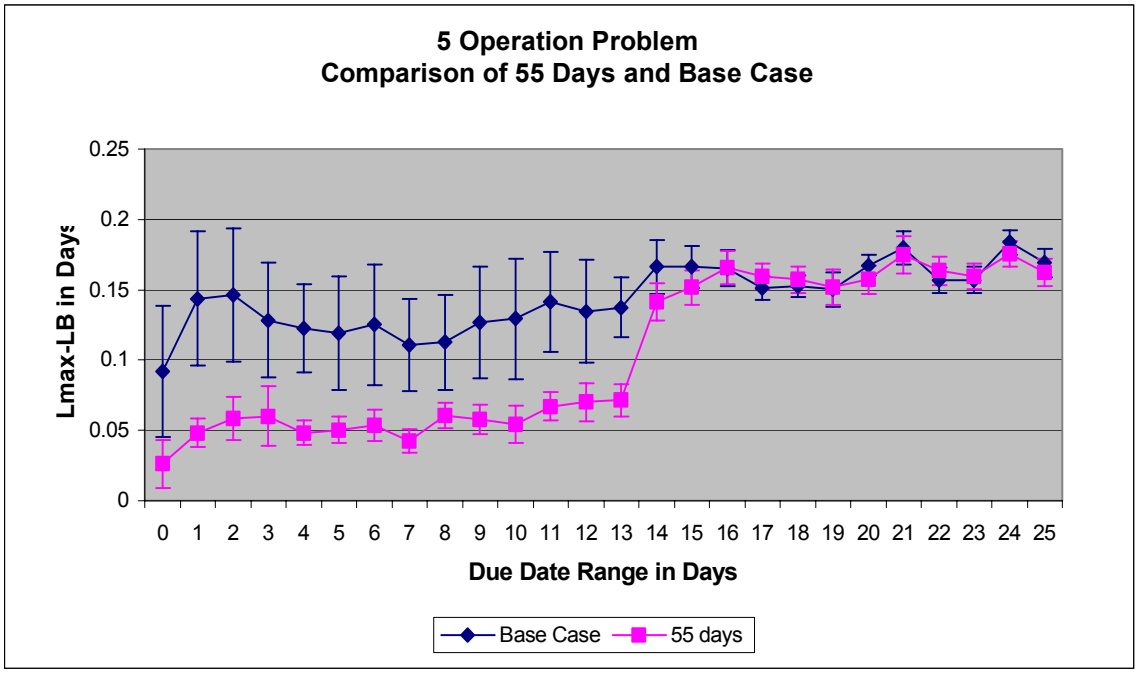

Figure 4.3: 5 operation problem-55 days

The results of the 5 operation and 7 operation problems with 55 days are shown in Figure 4.3 and Figure 4.4, respectively. For the 5 operation problem, the average L_{max}- LB value is low for due date ranges up to 13 days. It increases suddenly at 14 days and then the results are similar to that of the base case thereafter. This indicates that for large due date ranges, the differences between L_{max} and LB do not change much on average, between 55 and 100 days, but they do increase significantly for small due date ranges. For the 7 operation problem, the average L_{max}- LB value is low for due date range up to 8 days and then it is follows a similar pattern as that of the base case.

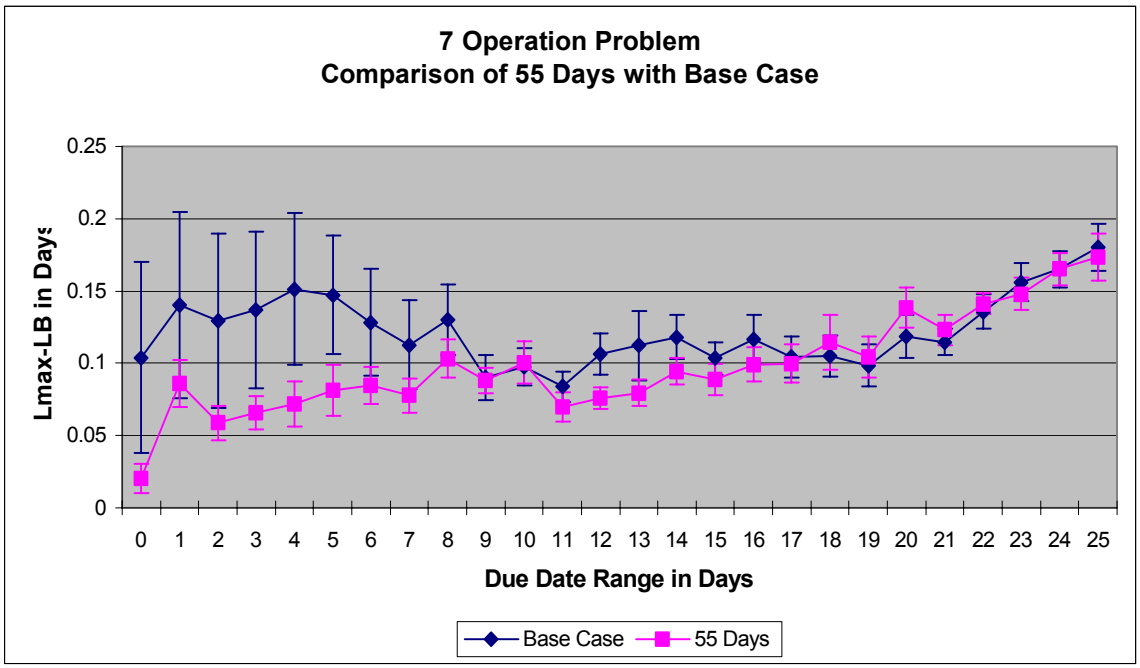

Figure 4.4: 7 operation problem-55 days

The results of the 5 operation problem run with 190 days is compared with the base case and is shown in Figure 4.5. The average L_{max}- LB is a little higher than the base case value for all the due date ranges.

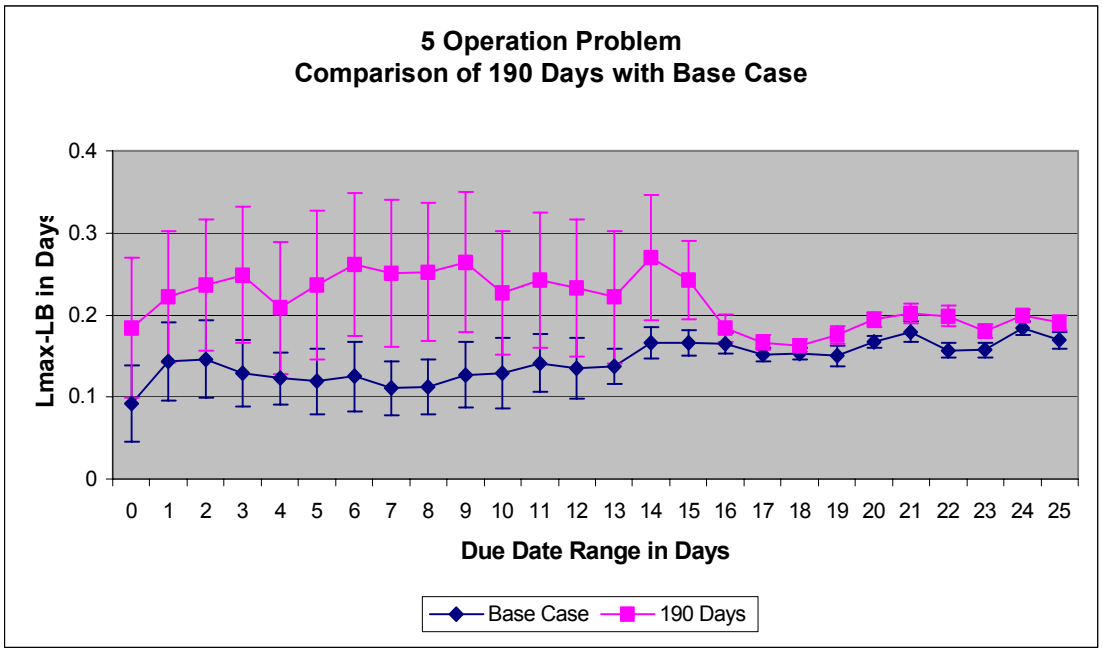

Figure 4.5: 5 operation problem-190 days

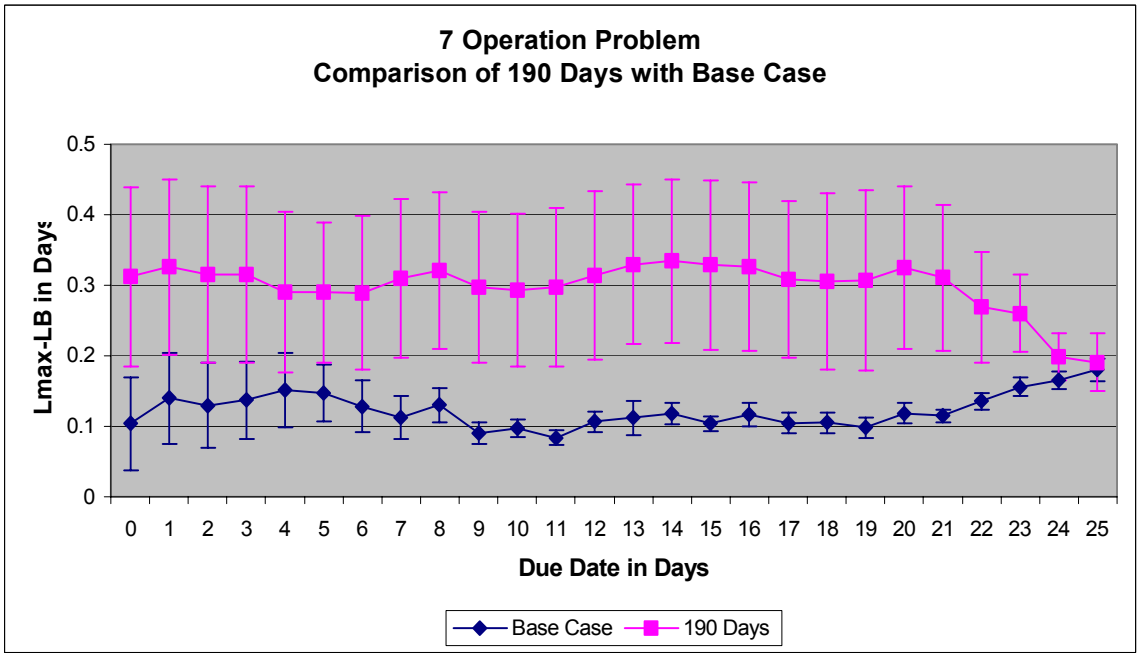

Figure 4.6: 7 operation problem-190 days

Figure 4.6 shows the comparison of the base case with 190 days run length for the 7 operation problem. The average L_{max}- LB is slightly higher than the base case until due date range of 21 days and then decreases to that of the base case value. Despite the fact that performance of the LB is not good as compared to the case of 100 days, the average L_{max}- LB is around 0.3 days, which is still quite good.

4.4.3 Increasing the Warm-up Period

The problems were run with an increased warm-up period to test if the transient effects were, indeed, eliminated.

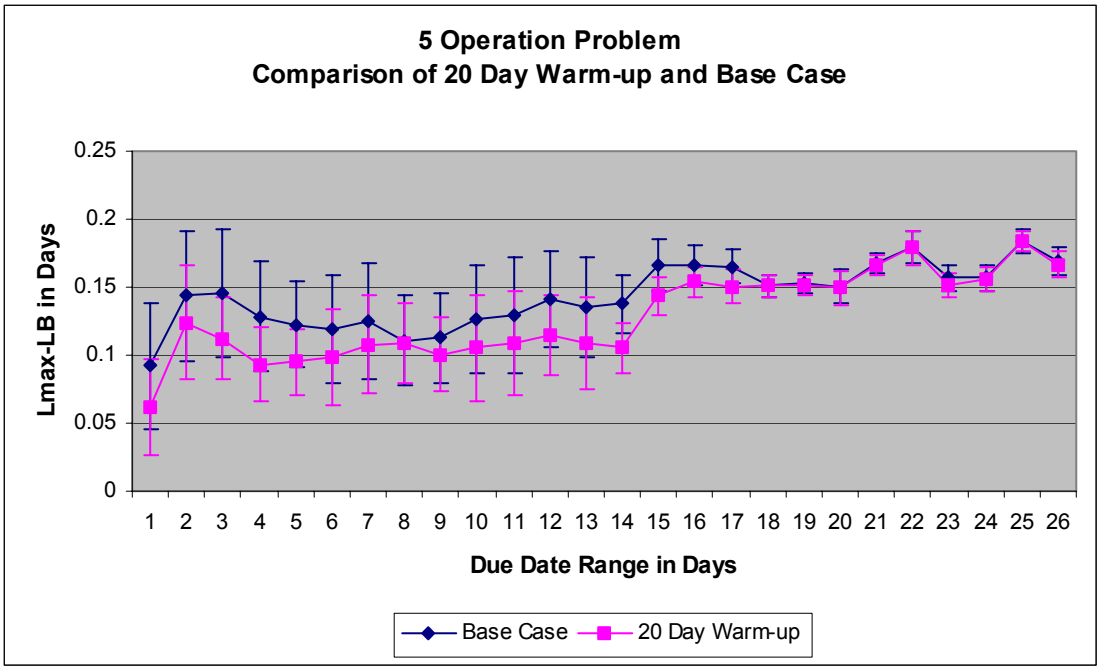

Figure 4.7: 5 operation problem-20 day warm-up

Figure 4.7 and Figure 4.8 show the comparison of increasing the warm-up period with the base case for the 5 operation problem and 7 operation problem, respectively. It is difficult to determine if the differences indicate that there are some

transient effects remaining when a warm-up period of 10 days is used or if this is a result of the slight decrease in performance since the horizon length is longer with 10 days of warm-up, evidenced in Section 4.4.2.

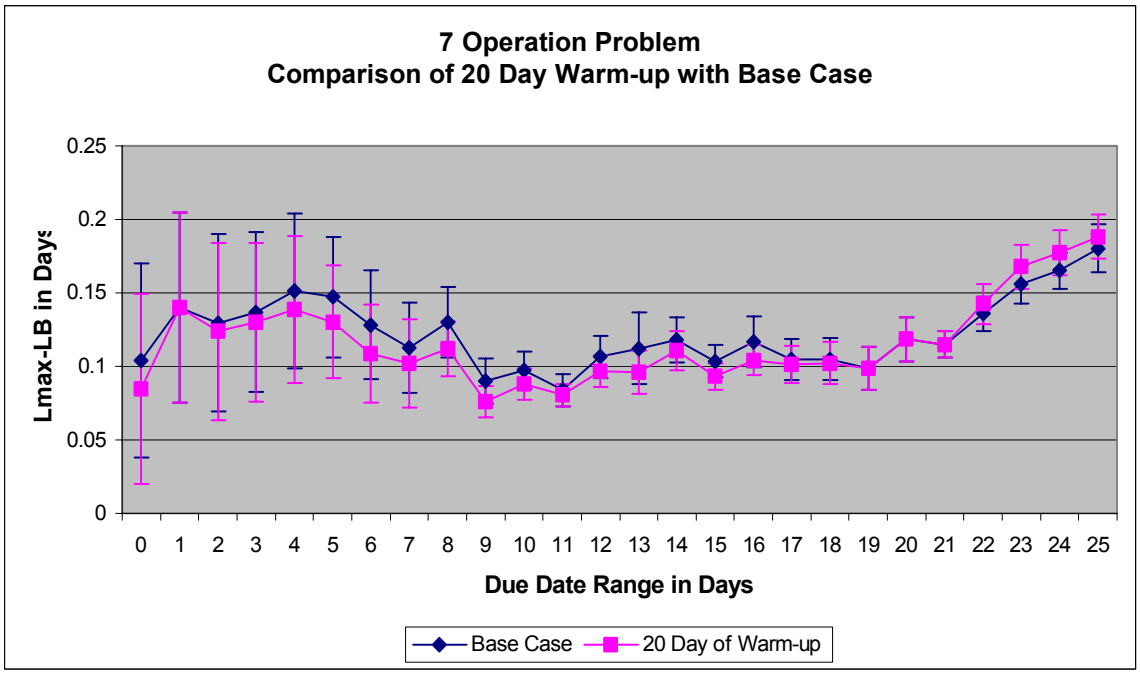

Figure 4.8: 7 operation problem-20 day warm-up

4.4.4 Varying the Number of Jobs Released

In industry, it would be uncommon for a factory to release exactly the same amount of jobs each day. Thus, to see the impact that varying the number of jobs released each day has on the ability of the VF to provide good schedules, experiments were carried out for both problems in which the average number of jobs released was approximately equal to the number released in the base cases. For the 5 operation problem, the number of jobs released each day was uniformly distributed between 145 and 155. For the 7 operation problem, the number of jobs released each day was varied

uniformly between 160 and 170. Figures 4.9 and 4.10 show the 5 operation and 7 operation problem, respectively. There is little difference between the base cases and the corresponding cases where the number of jobs released was varied.

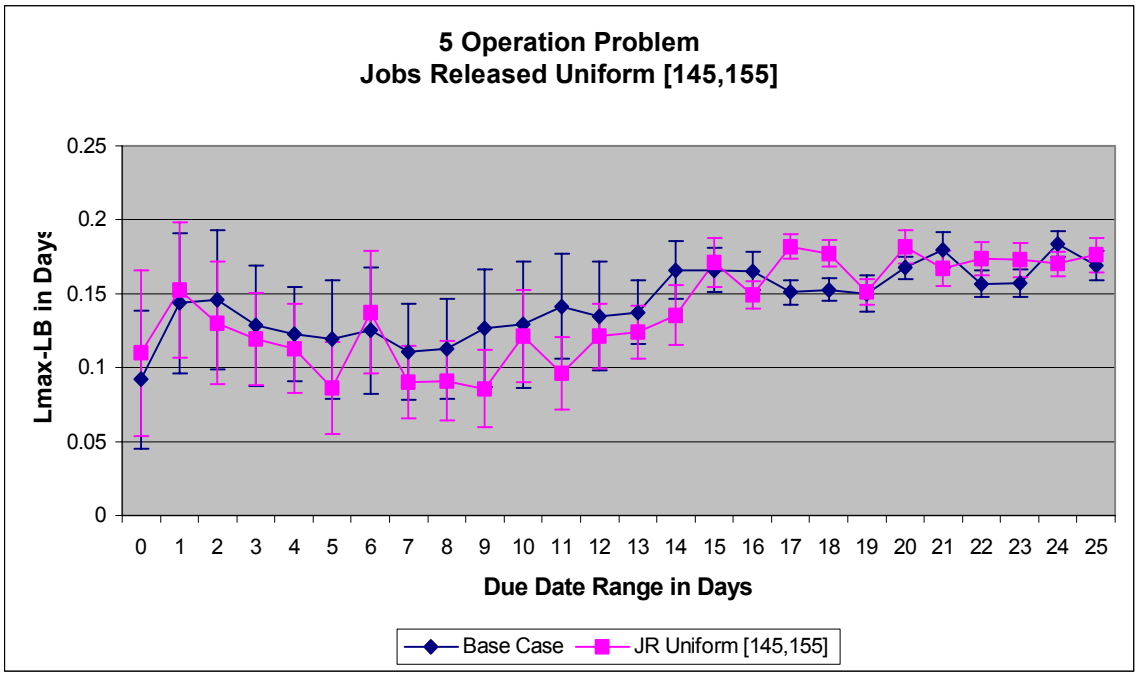

Figure 4.9: 5 operation problem-jobs released uniform [145,155]

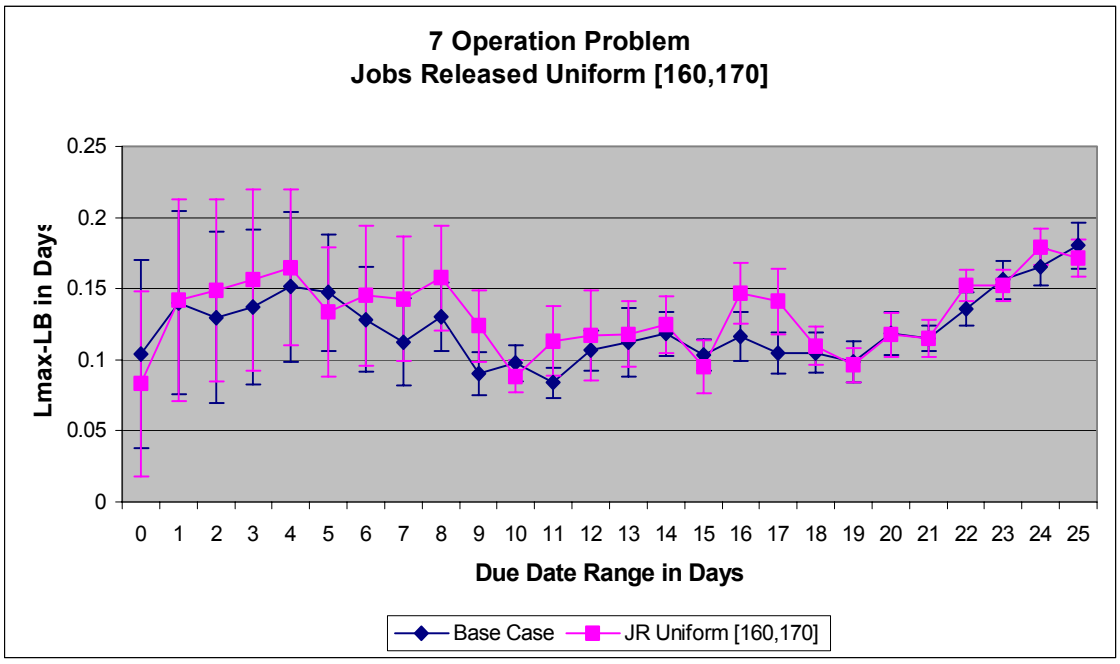

Figure 4.10: 7 operation problem-jobs released uniform [160,170]

4.4.5 Varying the Number of Operations

Releasing jobs with varying number of operations is also a typical occurrence in industry that has the potential to affect the performance of a scheduling algorithm. Therefore, this parameter has been varied, setting the average number of operations equal to the number of operations used in the base cases. For the 5 operation problem, the number of operations remaining for the jobs released each day was varied uniformly between 3 and 7. Figure 4.11 shows the comparison of changing the number of operations with the base case. When compared with the base case, varying the number of operations sometimes yields slightly better results for low due date ranges and slightly worse results for high due date ranges.

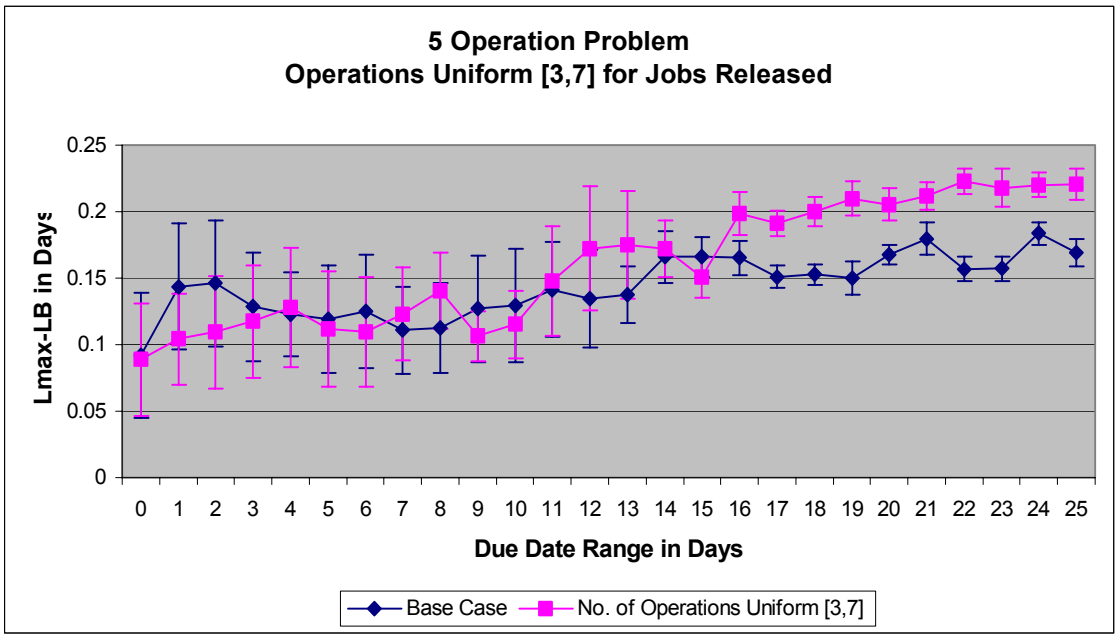

Figure 4.11: 5 operation problem-number of operations uniform [3,7]

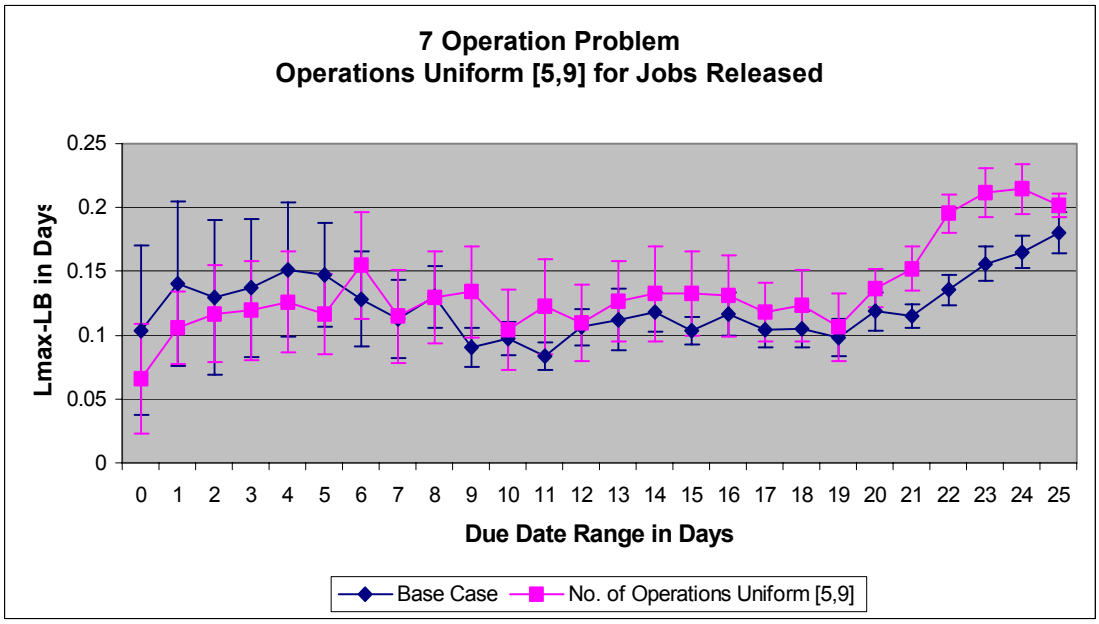

Figure 4.12: 7 operation problem-number of operations uniform [5,9]

For the 7 operation problem the number of operations remaining was varied uniformly between 5 and 9. Figure 4.12 shows the comparison of changing the number of operations for the 7 operation problem. A similar pattern to that found in the 5 operation problem is seen.

4.4.6 Varying the Number of Jobs Released and the Number of Operations

Since both releasing different numbers of jobs per period and releasing jobs with varying number of operations is common in industry, these variations should also be tested simultaneously.

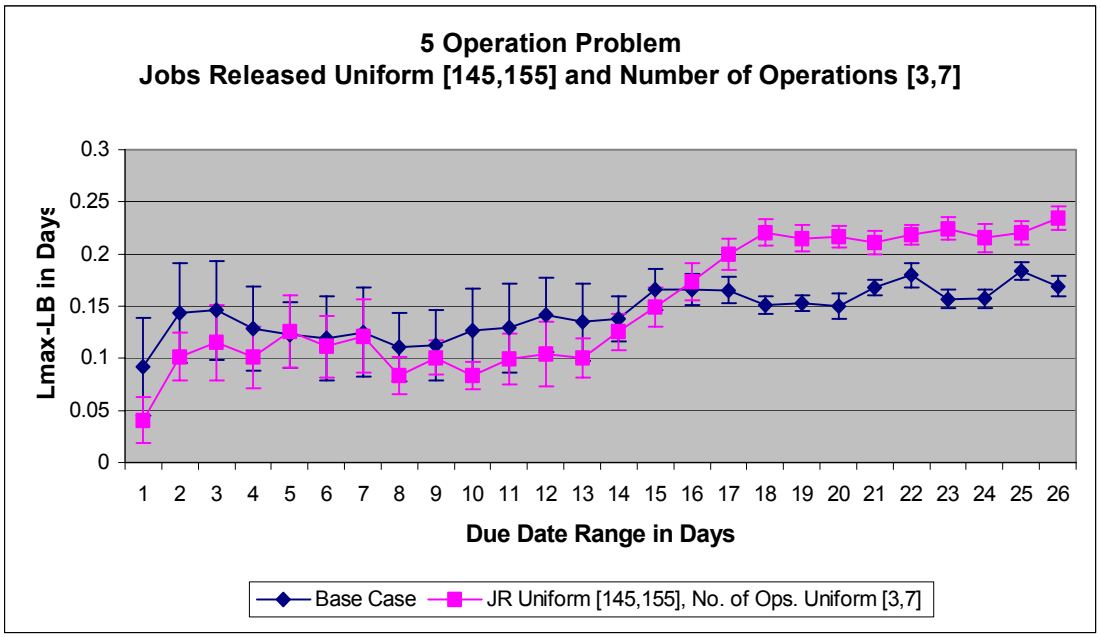

Figure 4.13: 5 operation problem-number of jobs released uniform [145,155] and number of operations uniform [3,7]

Figure 4.13 shows the comparison of changing both the number of jobs released and the number of operations remaining with the base case for the 5 operation problem. The results shows that changing both the number of jobs released and also the number of operations remaining yielded better results up to a due date range of 15 days and inferior results after that. Figure 4.14 show that the 7 operation problem results are similar to that of the 5 operation problem.

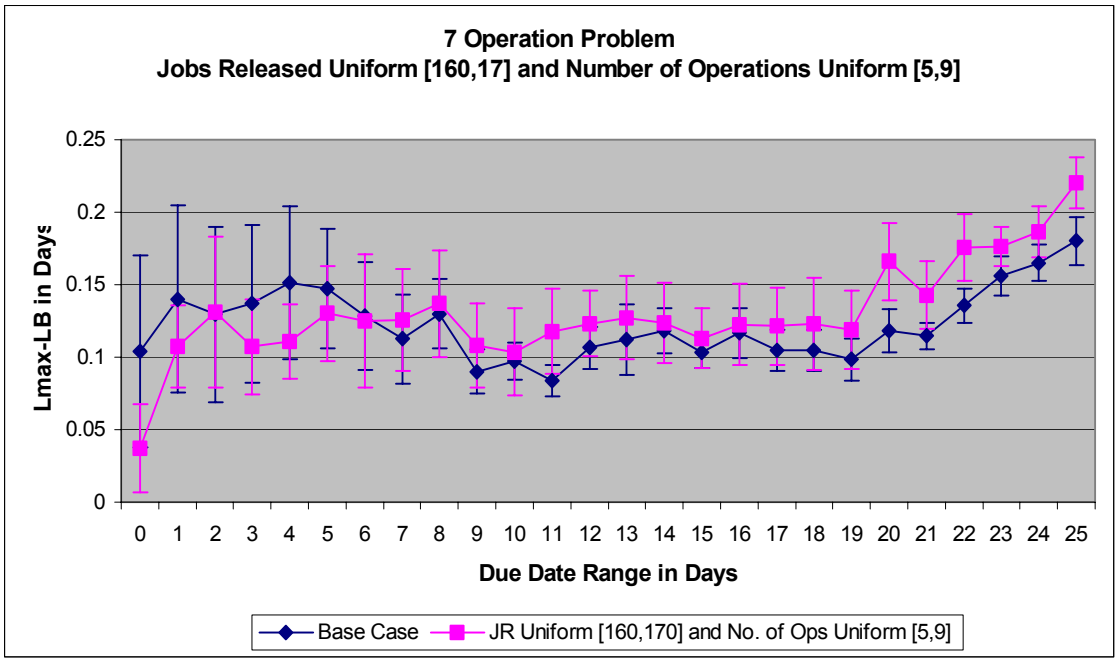

Figure 4.14: 7 operation problem-number of jobs released uniform [160,170] and number of operations uniform [5,9]

4.4.7 Effect of the Scheduling Frequency

To find out if using a rolling horizon approach to scheduling is detrimental to the quality of scheduling solutions when compared to knowing all the information about jobs releases in advance, the rolling horizon solutions were compared to

scheduling the total horizon once. Figures 4.15 and 4.16 show these comparisons for the 5 operation and 7 operation problems, respectively. Initially the scheduling once procedure was run for 100 iterations. Since it did not perform as well as the rolling horizon algorithm, it was conjectured that perhaps the comparison was not fair because the total number of iterations of the VF run during the rolling horizon simulation was much larger. Therefore, the number of iterations was increased to 1000. This yields exactly the same solutions as when using only 100 iterations. This experimentation shows that the VF actually works somewhat better when all upcoming job releases are not known in advance.

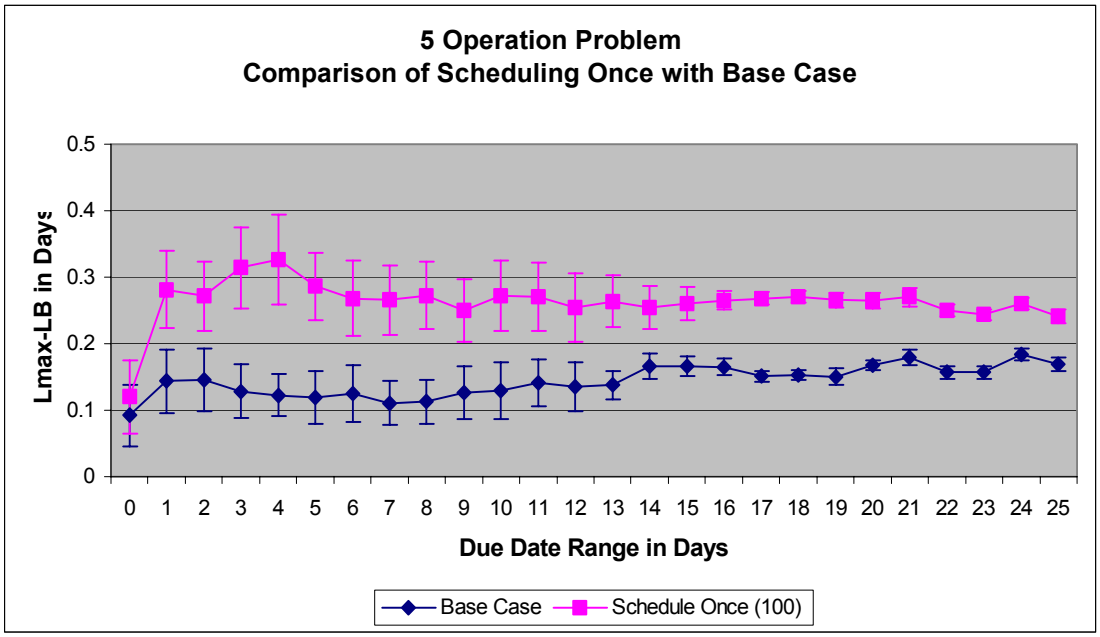

Figure 4.15: 5 operation problem-comparison of scheduling once with base case

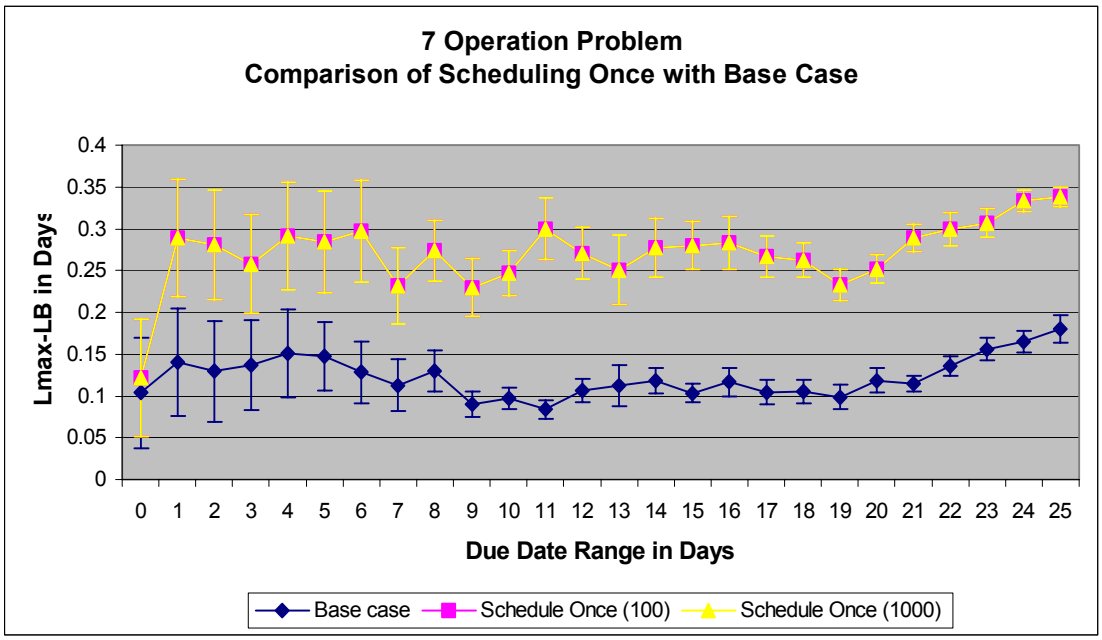

Figure 4.16: 7 operation problem-comparison of scheduling once with base case

4.4.8 Hot Job Acceleration

Hot job acceleration improved the performance of transient scheduling scenarios by narrowing the gap between the scheduling solution and the lower bound. Thus, its potential for improving the rolling horizon scheduling solutions was explored. Figures 4.17 and 4.18 compare the base cases with the results of hot job acceleration for the 5 operation and 7 operation problem, respectively. Hot job acceleration appears to only help slightly in the very high due date ranges for the 5 operation problem. In addition, there are due date ranges for both problems in which hot job acceleration does much worse than the original VF algorithm.

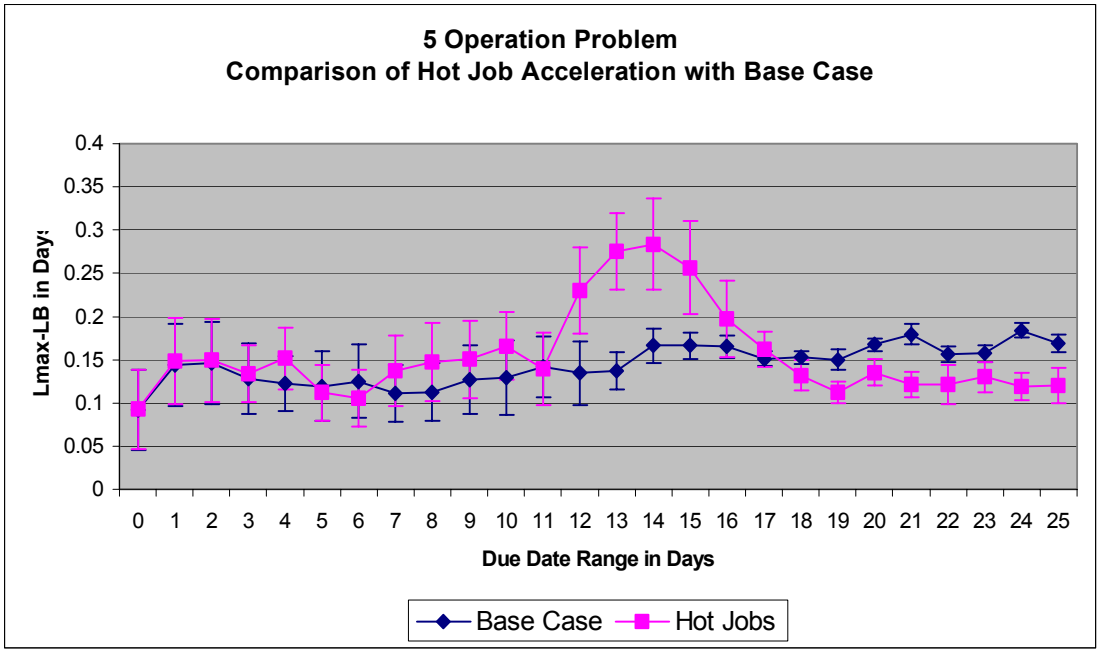

Figure 4.17: 5 operation problem-comparison of hot job acceleration with base case

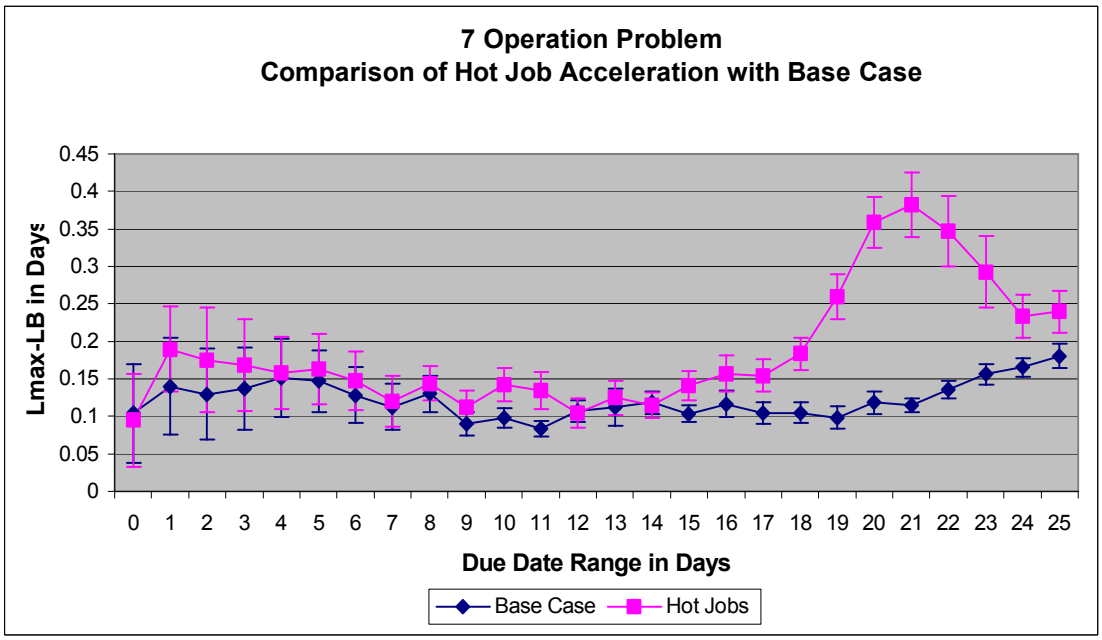

Figure 4.18: 7 operation problem-comparison of hot job acceleration with base case

4.4.9 Simulated Annealing

Using simulated annealing as a post-processing algorithm improved the scheduling solutions of transient situations. Therefore, this method will be applied to the rolling horizon scenarios. For transient scheduling, the hot job acceleration solution was used as a starting point for the simulated annealing algorithm. But since, in general, hot job acceleration does not improve the rolling horizon scheduling solutions, the original VF solution will be used as the starting point for the simulated annealing procedure with rolling horizon scheduling. The simulated annealing procedure was applied to every schedule each day.

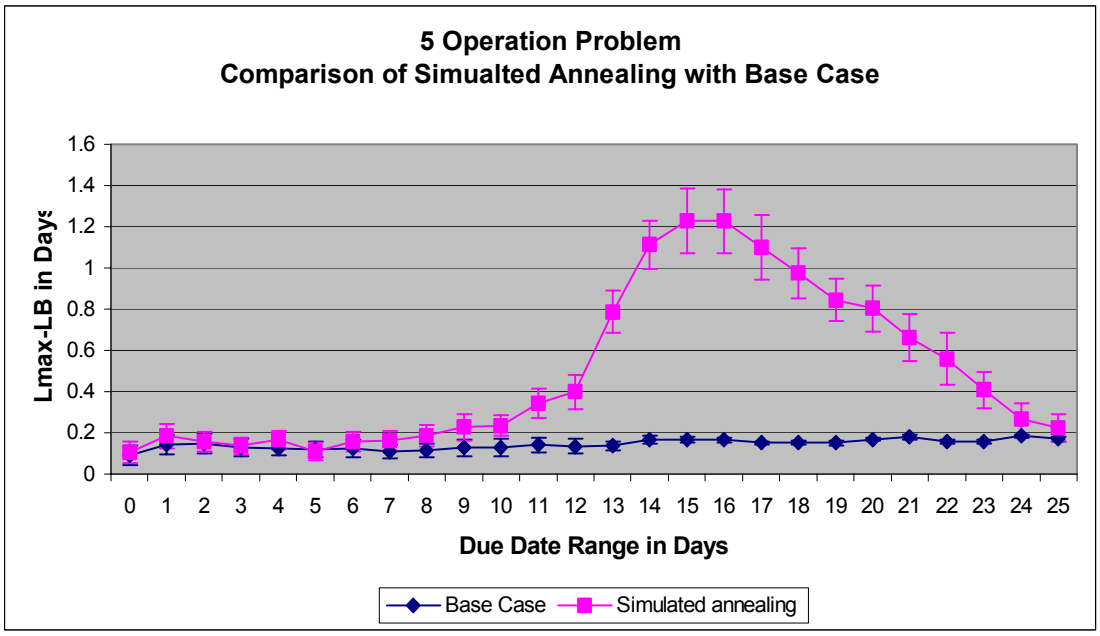

Figure 4.19: 5 operation problem-simulated annealing

Figure 4.19 and Figure 4.20 show the comparison of simulated annealing results with the base case for the 5 operation and 7 operation problems, respectively.

Simulated annealing did not provide better results. For a large number of due date ranges, the results were worse than the base case.

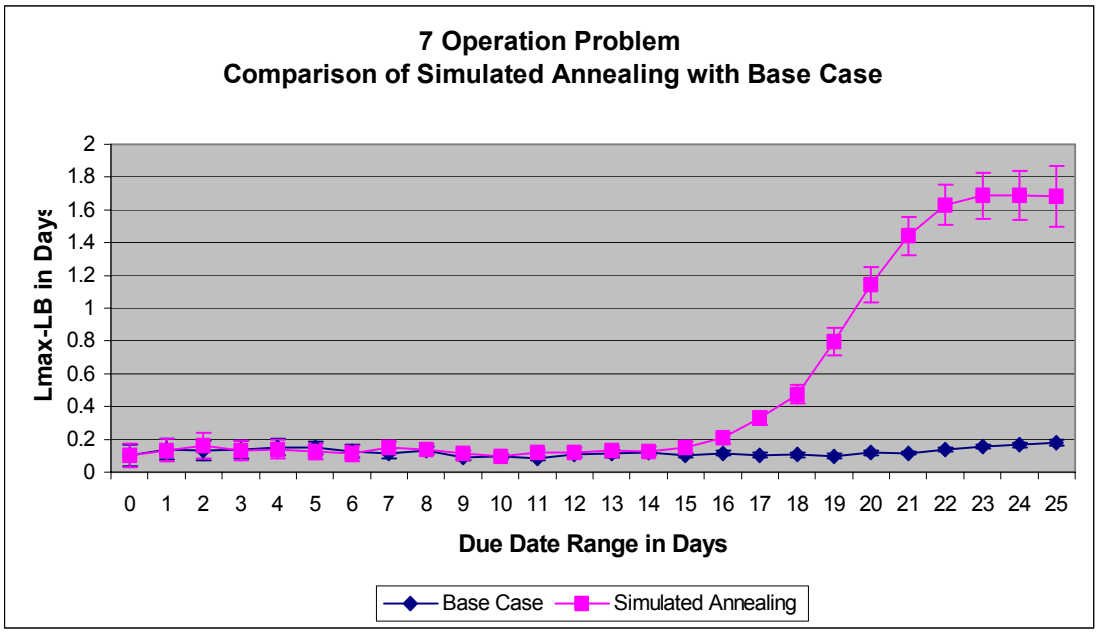

Figure 4.20: 7 operation problem-simulated annealing

Chapter 5

Conclusions and Future Research

5.1 Conclusions

In this thesis, the Virtual Factory was used as a tool for scheduling large job shop problems. As the Virtual Factory contains more than 40 classes, understanding and also learning to conduct experiments with it is a very time consuming process for the user, especially if he/she is not already familiar with C++. To enable the user to more quickly be able to accurately run desired experiments, a Visual Basic interface was developed. A detailed description of this interface was presented.

A method to test the Virtual Factory on a rolling horizon basis was also developed, both in terms of a scheduling algorithm and a lower bound. The Virtual Factory was shown to perform well in a rolling horizon setting under a variety of different conditions. Unfortunately, the hot job acceleration and simulated annealing procedures did not improve performance as they did in the transient experiments. The slight increase in the difference between L_{max} and LB as the total horizon length was increased could be the result of deterioration in the quality of the scheduling solutions or the lower bound. In any case, the differences in all cases are quite small with respect to the total horizon length and the number of jobs completed.

5.2 Future Research

Future experimentation will be concentrated primarily in three different areas. First, data has recently been received from a medium-sized U.S. apparel manufacturer. This manufacturer uses the progressive bundle system, and his factory is set up as a job shop. By applying the rolling horizon version of the Virtual Factory to the data from this plant, an assessment of how well the Virtual Factory would perform as a scheduling tool in the apparel industry can be obtained.

Second, the rolling horizon methodology will be used to further evaluate the multi-factory scenarios described in Thoney et al [19]. Evaluating these scenarios in a rolling horizon setting is especially important to eliminate the many transient effects found in the initial experimentation. This will allow us to more accurately quantify how well the Virtual Factory schedules supply chains. Furthermore, industrial data from fiber and textile manufacturers can also be gathered and implemented into the Virtual Factory. The end result would be the ability to test the scheduling performance of the Virtual Factory on the Integrated Textile Complex as a whole.

Lastly, stochastic processing times could be used in the implementation of the best schedule in the rolling horizon version of the Virtual Factory. The Virtual Factory schedules plants using deterministic processing times. Up to now, the actual processing times for the jobs have been assumed to be deterministic in the rolling horizon implementation of the Virtual Factory. But in industry, individual processing times are likely to follow some probability distribution. By using the mean processing time to schedule each job with the Virtual Factory and then generating a random

variable for the actual processing time in the implementation of the best schedule, additional insight can be gained into the potential performance of the Virtual Factory in industry.

Bibliography

[1] Baker, K.R. (1977). "An experimental study of the effectiveness of rolling schedules in production planning", *Decision Sciences*, Vol. 9, pp 19-27.

[2] Baker, K.R., and Peterson, D.W. (1979). " An analytic framework for evaluating rolling schedules", *Management Science,* Vol. 25, No. 4, pp 341-351.

[3] Baker, K.R. and Su, Z. (1974). "Sequencing with due-dates and early start times to minimize maximum tardiness", *Naval Research Logistics Quarterly,* Vol. 21, No. 1, pp 171-176.

[4] Blackburn, J.D., and Millen, R.A. (1980). "Heuristic lot-sizing performance in a rolling schedule environment", *Decision Sciences,* Vol.11, pp 691-701.

[5] Bookbinder, J.H., and H'ng, B.T. (1986). "Rolling horizon production planning for probabilistic time-varying demands", *International Journal of Production Research,* Vol. 24, No. 6, pp 1439-1458.

[6] Campbell, G.M. (1992). "Master production scheduling under rolling planning horizons with fixed order intervals", *Decision Sciences,* Vol. 23, No. 2, pp 312-331.

[7] Cario, M.C., Nelson, B.L., Roberts, S.D., and Wilson, J.R. (1999). "Modeling and generating random vectors with arbitrary marginal

distributions and correlation matrix," Technical Report, Department of Industrial Engineering and Management Science, Northwestern University.

[8] Cario, M.C. and Nelson, B.L. (1996). "Autoregressive to anything: Time series input processes for simulation", *Operations Research Letters 19,* pp 51-58.

[9] Carlson, R.C., Beckman, S.L., and Kropp, D.H. (1982). "The effectiveness of extending the horizon with fixed order intervals", *Decision Sciences,* Vol. 12, pp 129-146.

[10] Chand, S. (1982). "A note on dynamic lot-sizing in a rolling horizon environment", *Decision Sciences,* Vol. 13, pp 113-119.

[11] Demirkol, E., Mehta, S., and Uzsoy, R. (1998). "Benchmarks for shop scheduling problems", *European Journal of Operational Research*, Vol. 109, pp 137-141.

[12] Hodgson, T.J., Cormier, D., Weintraub, A.J. and Zozom, A. (1998). "Satisfying due dates in large job shops*", Management Science*, Vol. 44, No. 10, pp 1442-1446.

[13] Hodgson, T.J., King, R.E., Thoney, K.A., Stanislaw, N., Weintraub, A.J., and Zozom, A. (2000). "On satisfying due-dates in large job shops: idle time insertion", *IIE Transactions*, Vol. 32, pp 177-180.

[14] Kleindorfer, P., and Kunreuther, H. (1978). "Stochastic horizons for the aggregate planning problem", *Management Science,* Vol. 24, No. 5, pp 485-497.

[15] Kunreuther, H.C., and Morton, T.E. (1973). "Planning horizons for production smoothing with deterministic demands, I", *Management Science,* Vol. 20, No. 1, pp 110-125.

[16] Kunreuther, H.C., and Morton, T.E. (1974). "Planning horizons for production smoothing with deterministic demands, II", *Management Science,* Vol. 20, No. 7, pp 1037-1046.

[17] Lawrence, S.R. and Morton, T.E. (1986). "Patriarch: Hierarchical production scheduling", *National Bureau of Standards Special Publication*, Vol. 724, pp 87-97.

[18] Lundin, R.A., and Morton, T.E. (1975). "Planning horizons for the dynamic lot size model: Zabel vs. protective procedures and computational results", *Operations Research,* Vol. 23, No. 4, pp 711-734.

[19] Matta, R.D., and Guignard, M. (1995). "The performance of rolling production schedules in a process industry", *IIE Transactions,* Vol. 27, pp 564-573.

[20] Ovacik, I.M., and Uzsoy, R. (1994b). "Rolling horizon algorithms for a single machine dynamic scheduling problem with sequence dependent setup times", *International Journal of Production Research,* Vol. 32, pp 1243-1263.

[21] Ovacik, I.M., and Uzsoy, R. (1995). "Rolling horizon procedures for dynamic parallel machine scheduling with sequence-dependent setup

times", *International Journal of Production Research,* Vol. 33, No. 11, pp 3173-3192.

[22] Russell, R.A., and Urban, T.L. (1993). "Horizon extension for rolling production schedules: Length and accuracy requirements", *International Journal of Production Economics,* Vol. 29, pp 111-122.

[23] Schultz, S.R., Hodgson, T.J., King, R.E. and Thoney, K.A. "On minimizing L_{max} for the job shop scheduling problem", submitted to *IIE Transactions*.

[24] Simpson, N.C. (1999). "Multiple level production planning in rolling horizon assembly environments", *European Journal of Operational Research,* Vol. 114, pp 15-28.

[25] Sridharan, V., Berry, W.L., and Udhayabhanu, V. (1988). "Freezing the master production schedule under rolling horizon plans", *Management Science,* Vol. 33, No. 9, pp 1137-1149.

[26] Sridharan, V., Berry, W.L., and Udhayabhanu, V. (1988). "Measuring master production schedule stability under rolling planning horizons", *Management Science,* Vol. 19, No. 2, pp 147-166.

[27] Stadler, H. (2000). "Improved rolling schedules for the dynamic single level lot-sizing problem", *Management Science,* Vol. 46, No. 2, pp 318-326.

[28] Thoney, K.A., Hodgson, T.J., King, R.E., Taner, M.R., and Wilson, A.D. "Satisfying due-dates in large multi-factory supply chains", accepted by *IIE Transactions*.

[29] Venkataraman, R. and Smith, S.B. (1996). "Disaggregation to a rolling horizon master production schedule with minimum batch-size production restrictions", *International Journal of Production Research,* Vol. 34, No. 6, pp 1517-1537.

[30] Vepsalainen, A.P.J. and Morton, T.E. (1988). "Improving local priority rules with global lead-time estimates: A simulation study", *Journal of Manufacturing and Operations Management,* Vol.1, pp 102-118.

[31] Weintraub, A.J., Cormier, D., Hodgson, T.J., King, R.E., Wilson, J.R. and Zozom, A. (1999). "Scheduling with alternatives: A link between process planning and scheduling", *IIE Transactions,* Vol.31, No.11, pp 1093-1102.

[32] Wemmerlöv, U., and Whybark, D.C. (1984). "Lot-sizing uncertainty in a rolling schedule environment", *International Journal of Production Research,* Vol. 22, No. 3, pp 467-484.

[33] Zozom, A., Hodgson, T.J., King, R.E., Weintraub, A.J. and Cormier, D. "Determining release times to guarantee due-date performance", submitted to *IIE Transactions*.