Abstract

Davenport, Catherine Elizabeth. Optimization in Job Shop Scheduling Using Alternative Routes. (Under the direction of Dr. Russell King.)

The ability of a production system to complete orders on time is a critical measure of customer service. While there is typically a preferred routing for a job through the processing machines, often an alternative route is available that can be used to avoid bottleneck operations and improve due date performance. In this paper a heuristic approach is given to dynamically select routing alternatives for a set of jobs to be processed in a job shop. The approach is coupled with a job shop scheduling algorithm developed by Hodgson *et al.* (1998, 2000) to minimize the latest job (L_{max}).

Optimization in Job Shop Scheduling Using Alternative Routes

by
Catherine Elizabeth Davenport

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

Operations Research

Raleigh, NC

December 2002

APPROVED BY:

_____	_____
Dr. R. E. King	Dr. Kristin Thoney
Chair of Advisory Committee	Co-Chair of Advisory Committee
_____	_____
Dr. Scott Schultz	Dr. Thom Hodgson
Co-Chair of Advisory Committee	Minor representative

Biography

I was born on October 1, 1978 in Columbia, SC. During elementary school, my family moved to Blountville, TN. After graduating as valedictorian of Sullivan Central High School, I attended the University of Tennessee in Knoxville, where I received a Bachelor of Science Degree in Industrial Engineering. During that time, I married my high school sweetheart, Tim Davenport.

After graduation in Knoxville, Tim and I decided to move to Raleigh, NC to pursue graduate degrees at North Carolina State University. I will graduate in December 2002 with a Masters of Science degree in Operations Research.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Problem Definition

This research focuses on scheduling jobs in a job shop in order to satisfy customer due dates. Each job has a standard process plan, i.e. ordered set of operations, that must be performed on specified machines. However, an alternative process plan may be used in order to improve due date performance. In general, this alternative plan may take one of three different forms. First, some operation in the process route may be performed on an alternative machine or set of machines. Second, an entirely different route may be used. Finally, the ordering of consecutive operations may be switched. This paper will focus on alternative process plans that involve an alternative machine or set of machines to replace the bottleneck operation. The purpose of this research is to determine the appropriate set of routings to use for a given set of jobs and sequencing the operations on all machines in order to minimize the latest job (L_{max}).

The objective of minimizing L_{max} is supported in Raman (1995) who notes that the surveys of Panwalkar *et al.* (1973) and Smith *et al.* (1986) reported that 'operating managers consider meeting due-dates as their most important objective, and other scheduling criteria are considered only after the best schedule for meeting job due-dates has been determined'. In the application that prompted this research, the customer was the final assembly process. If one part of an assembly was tardy, the final assembly

schedule was delayed. The company's objective was to satisfy all due-dates, i.e. never starve the final assembly.

1.2 Thesis Organization

This thesis is organized in the following manner. In Chapter 2, a literature review is presented. Both general research involving alternative routes and more specific research using revised slack and the Virtual Factory Application are discussed. In Chapter 3, the critical path approach used in this research is explained. The new tabu search procedure involving the critical path is discussed in Chapter 4 and the algorithm is presented in detail. Computational results are presented in Chapter 5. Chapter 6 gives final conclusions and future research ideas.

Chapter 2

Literature Review

2.1 Previous Research in Job Shop Scheduling using Alternative Routings

There has been much research involving job shop scheduling with alternative routes. In the last few years, several papers deal with the use of genetic algorithms (GAs). Kim *et al*. (2002) deal with alternative routings in the process planning stage using a co-evolutionary algorithm in order to minimize makespan and minimize mean flow time over all the jobs. They integrate process planning and scheduling such that a change in the process planning stage also changes the scheduling stage. They evaluate alternative routes using all three possibilities (operations, sequencing, and processing) as well as a network consisting of all three together. They also assume that there are a large number of process plans for each job. Their algorithm is tested on problems with 6-18 jobs and 8-22 operations per job.

Candido *et al*. (1998) consider alternative routes by dividing the routes into sub-processes, where every sub-process is a technologically similar sequence of operations. An alternative route must be able to perform all sub-processes. They also consider route changes due to alternate machines that the job can go through. They create an initial

schedule using dispatch rules such that no operation can start earlier without preempting another operation. This technique relies on the assumption that some parts are sub-assemblies for others and must be completed first. A simple hill climbing algorithm is then implemented to find the local minimum makespan. Lastly, a GA is used to further minimize the makespan. Alternative routing is only one of seven "constraints" that are used.

Chryssolouris and Subramaniam (2000) present a scheduling method based on extreme value theory (SEVAT) to solve the dynamic job shop problem. SEVAT, which is traditionally used to predict rare events, in this case creates a "statistical profile of schedules through random selection, and predicts the quality…of a feasible schedule." The objective is to minimize mean job tardiness and mean job cost.

Chryssolouris and Subramaniam (2001) present a GA for scheduling a dynamic job shop using mean job tardiness and mean job cost as performance measures. The problem they consider has both random machine breakdowns and alternative job routings. They consider up to 6 machines and up to 6 different resources to process an operation.

Lee *et al*. (2002) look at an APS (advanced planning and scheduling) model to minimize makespan using alternative machines, alternative operations sequencing and outsourcing. They consider a supply chain to be job-shop-like and use a GA to optimize the process. Their goal is to minimize the makespan of each order, where an order is a set of jobs with the same due date. Their algorithm is tested on 8-64 jobs, 6 machines and 160 operations total among all the jobs.

Algorithms other than GAs have also been researched. Dauzère-Pérès and Paulli (1997) deal with the multi-processor job shop where every operation has more than one machine on which it can be performed. They use the objective of minimizing makespan and consider an integrated approach for assigning machines and scheduling the parts on those machines using a tabu search algorithm with a new neighborhood structure.

Kim and Egbelu (1999) develop two local search heuristics to find alternative routings in order to minimize makespan. The heuristics are greedy in nature and terminate at local optima. The largest problems solved with these procedures are 10 jobs and 10 machines, with 2-5 routings per job. It is noted that with an increase in the number of jobs or the number of routings, the heuristic performs worse.

Thomalla (2001) solves the job shop scheduling problem with alternative routes using Lagrangian relaxation. The largest problem solved is 6 jobs, 6 machines, and 10 operations per job with the objective being to minimize the sum of the weighted quadratic tardiness of the jobs.

Saygin *et al.* (2001) present an algorithm, the Dissimilarity Maximization Method (DMM), to solve a real-time rescheduling problem that arises due to machine breakdowns and other unexpected occurrences on the shop floor. The goal of the research is to maximize the throughput rate. They assume that processing times on alternate machines are the same as on the originally scheduled machine. Alternate routes consist of changing one or two machines on a route to a parallel alternative. DMM picks which routings to use in order to maximize the dissimilarity between machines used in the set of routings. The algorithm is solved on small problems; 6 parts, 7 machines and up to 8 alternate routings per part.

Choi and Choi (2002) consider alternative routes and sequence dependent setups together. They develop a mathematical programming model and a local search algorithm using dispatching rules with an objective of minimizing makespan.

Weintraub *et al.* (1999) contains an extensive review of the literature up to 1997. Most of these papers, however, consider the problem from a process planning viewpoint rather than that of detailed scheduling. The interested reader is referred to that article for review.

2.2 Previous Research in Job Shop Scheduling using Revised Slack and Alternative Routings

Weintraub *et al.* (1999) consider the problem of determining the appropriate set of routings to use for a given set of jobs and to sequence the operations on all machines in order to minimize the latest job (L_{max}). Each job has a primary route as well as an alternative route that uses an alternative machine for the bottleneck machine. The primary route for each job has the lowest production cost.

Determining the optimal solution for this problem is computationally intractable, even for small problems. Therefore, Weintraub *et al.* (1999) develop a tabu search-based heuristic with the objective to minimize cost. In the tabu search a solution is the set of routes to use for every job. The neighborhood of a given solution is the set of all solutions that differ only in the route selection for one job, i.e., if there are N jobs, then there are N neighbors of a given solution. Solutions are evaluated based upon a lower bound calculation using a relaxation proposed by Carlier and Pinson (1989). The lower

bound can be calculated very quickly, however it does not solve the sequencing problem. A detailed, dynamic job shop scheduling tool (Virtual Factory Algorithm, VFA) is used for this purpose (Hodgson *et al*. 1998, 2000).

VFA is a deterministic, iterative/adaptive simulation model including both single (Hodgson *et al*. 1998, 2000, Weintraub *et al*. 1999) and batch processors (Thoney 2000 and Thoney *et al*. 2002). It is capable of providing optimal or near-optimal solutions for industrial-sized problems with a relative minimum of computation. The scheduling procedure in the VFA is based on an approach that was first proposed by Lawrence and Morton (1986) and Vepsalainen and Morton (1988). The approach consists of repeatedly simulating the system to be scheduled while simultaneously updating job sequences based on the results of the previous simulation. During the first iteration, jobs are sequenced on machines in order of increasing slack. Let d_i be the due date of job i and p_{ij} be the processing time of job i on machine j. The slack of job i on machine m is computed as

$$slack_{im} = d_i - \sum_{j \in m+} p_{ij},$$

where $m+$ is the set of all operations on job i's route subsequent to the one performed on machine m. Slack is a measure of the amount of time a job can queue and still meet its deadline. In general, however, dispatching in order of increasing slack may not provide good results (Caroll 1965). This can be attributed to the fact that slack does not take queuing time into account. In subsequent iterations, the queuing time of the previous iteration is used to modify slack, and jobs are resequenced in order of revised slack. Let q_{ij} be the queuing time of job i at machine j. Revised slack is computed as

$$slack'_{im} = d_i - \sum_{j \in m+} p_{ij} - \sum_{j \in m+*} q_{ij},$$

where *m*+* is the set of all operations on job *i*'s route subsequent to the one performed on machine *m* except the one immediately following *m*. The slack calculation introduces an intermediate due date for the job. This is the time at which the job needs to begin processing at the next machine to meet its overall due date.

The procedure is run for a fixed number of iterations and the best solution saved. The procedure tends to converge monotonically over the first 10 iterations or so (regardless of the size of the problem). In other words the queuing time estimates become more accurate during the first 10 iterations, producing better and better solutions. After that, the solutions tend to fluctuate with no particular pattern involved. However, additional iterations typically yield better solutions. It should be noted that job expediting is handled implicitly.

Chapter 3

A Critical Path Approach for Scheduling using Alternative Routings

3.1 Previous Research using a Critical Path Neighborhood

In this paper, we attempt to improve the Weintraub *et al.* (1999) procedure by reducing the neighborhood to include only those jobs on the "critical path" of the $L_{\max}$ job. Other researchers have also explored this idea, including Nowiski and Smutnicki (1996) and Kolisch and Hess (2000).

The research by Nowiski and Smutnicki (1996) studies a tabu search on job shop scheduling with minimizing makespan as their objective. Their tabu search is unique because they limit their neighborhood by only looking on a predefined critical path. Moves are based on adjacent or non-adjacent pairs of operations on a machine. There are no predetermined alternative routes mentioned. The neighborhood is reduced by eliminating those moves for which it is known that they do not improve the makespan. This method is based on research by Van Laarhooven *et al* and is cited in Nowiski and Smutnicki (1996). They test their algorithm on medium to large size problems and report good results; however, they point out that their neighborhood is very specialized to their problem and probably will not work in other situations.

Kolisch and Hess (2000) consider the scheduling of large-scale make-to-order assemblies using several methods including tabu search. Their objective is to minimize the sum of the weighted tardiness. Using a list scheduling approach to create feasible schedules, they consider two tabu searches, one with a simple neighborhood and one with a "critical neighborhood" which uses "problem insight." For the "critical neighborhood" tabu search, they determine for each operation the "set of blocking operations B_j which hampers operation *j* from being started one period earlier." A neighbor of the list π, with respect to *j*, is reached by "shifting *j* (and it predecessors) in front of the operation $h \in B_j$ which has the smallest list position." A large-step optimization tabu search that attempts to improve only local optima is then used while employing this neighborhood.

3.2 Definitions for the New Critical Path Neighborhood

As seen in the literature, the idea of a tabu search using a critical path for job shop scheduling is not a new concept. However, this neighborhood reduction idea has not been applied to scheduling with alternative routings. In light of that, this research attempts to solve the same problem as approached by Weintraub *et al.* (1999) using a more intelligent tabu search by choosing alternative routings based on a critical path.

For this tabu search algorithm, the $L_{\max}$ job is identified as the job with the largest lateness, where lateness is the completion time minus the due date. After the $L_{\max}$ job is identified, a localized neighborhood is developed to form a candidate group of jobs and a critical path. For each operation of the $L_{\max}$ job, a path is determined by tracing the $L_{\max}$

job, on that particular machine, backward to a break in the plan. A break is the last idle time on the machine just prior to the arrival of the $L_{\max}$ job. The candidate group of jobs to be switched is the set of jobs on that machine from the $L_{\max}$ job back to the break.

Since a candidate group of jobs is found for more than one machine, a larger candidate set of jobs, the global candidate set, is formed. A local candidate set is added to the global set as follows. The first candidate set to be added is that of the $L_{\max}$ job on its final machine. The $L_{\max}$ job is then traced backward through its route. If there is no idle time between processing the $L_{\max}$ job on its current machine and processing the $L_{\max}$ job on its previous machine, then the candidate group of the previous operation is also added to the global candidate group. This continues until we reach idle time in the path of the $L_{\max}$ job. The critical path contains every job in the global candidate group of jobs. The alternative routings of this set of jobs are then searched in order to improve $L_{\max}$.

A lower bound for $L_{\max}$ is determined using a method developed by Carlier and Pinson (1989). For each job i on machine m, the earliest possible start time (ES_{im}) and latest possible finish time (LF_{im}) are computed as follows,

$$ES_{im} = \sum_{j \in m_i^-} p_{ij}$$

$$LF_{im} = d_i - \sum_{j \in m_i^+} p_{ij}$$

where m_i^- is the set of all operations for job i that occur before machine m, m_i^+ is the set of all operations for job i that occur after machine m, p_{ij} is the processing time for job i on machine j, and d_i is the due date for job i. Since LF_{im} is seen as the effective due date for job i and ES_{im} is seen as the effective release time for job i (r_i), solving the $N/1/L_{\max}|r_i$

problem creates a lower bound for the N/M/$L_{\max}$ problem. Therefore, the lower bound for $L_{\max}$ can be computed as follows,

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

where $LB_m(L_{\max})$ is the solution to the N/1/$L_{\max}|r_i$ problem for machine m. To enable $LB(L_{\max})$ to be computed quickly, each $LB_m(L_{\max})$ is computed using preemption.

The job/routing combination that yields the best (i.e. minimum) lower bound from the tabu search is then input into the VFA in order to determine the best possible schedule for those jobs on the machines available.

The problems to be considered are large, industrial-size problems, (50-500 jobs and 10-100 machines). As inspired by the furniture industry, the preferred routing of a job has the shortest processing time; while the alternative routing uses older or less computerized equipment that bypasses the bottleneck machine. Therefore, all alternative routes will have a longer processing time than the original routes.

Chapter 4

A Tabu Search Procedure using a Critical Path Neighborhood to Schedule with Alternative Routings

4.1 Definitions for the Tabu Search

The following definitions are used in the proposed heuristic.

Routing selection representation – the routing selection for each job. Figure 1 shows a routing selection representation for a 5-job problem. Jobs 1, 3 and 5 use their route 1 while jobs 2 and 4 use their route 2.

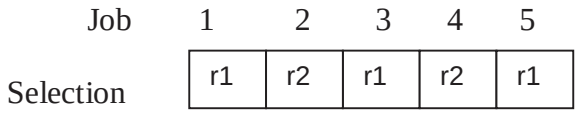

Figure 1: Routing selection representation

Critical Path – the critical path is defined as above. The backward trace needed is made using the schedule supplied by the *LPsolver* heuristic (Schultz *et al.*, working paper) that is the optimal schedule given a fixed sequence of jobs on each machine.

CP Neighborhood – the critical path neighborhood is a set consisting of solutions (job route selections) that differ from the current solution by a single alternative routing switch for a job on the critical path.

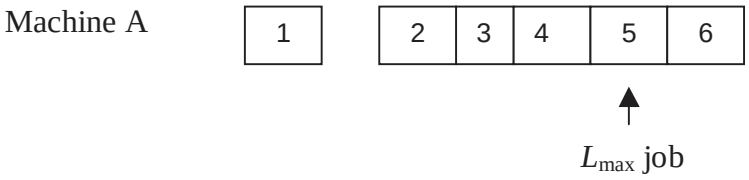

Figure 2: L_{max} Candidate group

The L_{max} candidate group in Figure 2 includes the jobs 2, 3, and 4. Job 1 is not in the candidate group because idle time exists prior to job 2. Job 6 is not in the candidate group because it follows the L_{max} job in the sequence.

4.2 The Critical Path Tabu Search Algorithm

1. Set the global minimum, $G_{min} = \infty$.
2. Run VFA for X iterations and save the Y best solutions, s_i, $i = 1,2,\ldots,Y$
3. Set $i = 1$
4. Set solution lower bound, *SLB*, equal to the lower bound value corresponding to s_i. If the L_{max} value corresponding to s_i is less than G_{min}, set G_{min} equal to the L_{max} value corresponding to s_i.
5. Set the current solution, S, equal to s_i.
6. Set the counter, C, equal to 0.

7. If $C < m$ (the number of neighbor moves), find the critical path for *S*. Otherwise, go to step 18.
8. Find the neighbors, n_i, $i = 1,2,\ldots t$ on the critical path that have alternative routes. If there are no neighbors that qualify, go to step 18.
9. Compute the lower bound for all n_i, $LB(n_i), i = 1,2,\ldots t$.
10. Set the neighbor lower bound, *NLB*, to infinity.
11. If $LB(n_i) < SLB$, set $SLB = LB(n_i)$ and set $NLB = LB(n_i)$. (aspiration criteria)

    Otherwise, if $LB(n_i) < NLB$ and the move is not tabu, set $NLB = LB(n_i)$. Repeat for all n_i, $i = 1,2,\ldots t$.
12. If *NLB* is not infinity (i.e. not every possible move is tabu), then $S = s_i$ where $NLB = LB(n_i)$.
13. Make the job switch from the former *S* to the current *S* tabu.
14. Run the VFA for *x* iterations on *S*.
15. Record the best solution, $L_{\max}$.
16. If $L_{\max} < G_{\min}$, set $G_{\min} = L_{\max}$.
17. $C = C+1$. Go to step 7.
18. $i = i+1$. If $i < Y$, go to step 4.
19. Output $G_{\min}$.

Chapter 5

Computational Results

5.1 Initial Trials

5.1.1 Data Generation

Problems are generated that have a single bottleneck machine, which, without loss of generality, is arbitrarily set to machine 1. For each job, a primary route is generated. A selected portion of the jobs that visit the bottleneck machine will have an alternative route. This alternative route is identical to the primary route with the exception that one or more alternative machines replace the bottleneck machine. This is the first form of alternative routes described earlier. Jobs that do not visit the bottleneck machine have only a primary route.

The following parameters are used to generate problems. B is the percentage of jobs whose primary route includes the bottleneck machine. Of those jobs visiting the bottleneck machine, A is the percentage with an alternative route. M is a multiplier by which the process time increases on the alternative operation, and, hence, $M \geq 1.0$. R is the maximum number of operations (or machines) that can replace the bottleneck operation on the alternative route.

Jobs are first identified as being on or off the bottleneck based on the value of *B*. If a job does not use the bottleneck machine, routes are generated as in a standard job shop with the only exception that machine numbers are generated between 2 and *p*, where *p* is the maximum number of machines in the shop (i.e. Machine 1 can not be generated since machine 1 is the bottleneck machine.) If a job is routed through the bottleneck, it has a certain probability (based on *A*) of possessing a second route. Jobs on the bottleneck with only one route use a similar route generator as in a standard job shop with the exception that machine 1 must be on the route.

For those jobs with an alternative route, the second route is identical to the first with the following exceptions: (1) the operation containing machine 1 is randomly replaced with other machine(s) in the shop. The number of replacement operations is generated from a uniform distribution U[1, *R*] and (2) the process time for the original operation is multiplied by the *M* parameter. If multiple operations replace the bottleneck operation, then the process time for each operation is the original process time multiplied by *M*, then divided by the number of replacement operations.

5.1.2 Setting Parameters

The critical path algorithm (*CPA)* explained above has several parameters that must be determined. The number of tabu search moves, *m*, and the number of VFA iterations, *x*, performed for each move in the tabu, both directly impact the quality of the solution, as well as the runtime of the algorithm. The values of these parameters must be determined experimentally.

The quality of solution is defined as $L_{\max} diff = L_{\max}(W) - L_{\max}(CP)$, where $L_{\max}(W)$ is the best $L_{\max}$ value found using the Weintraub *et al.* (1999) algorithm (*WA*) and $L_{\max}(CP)$ is the best $L_{\max}$ value found using the critical path algorithm. The time is measured using $tdiff = t(W) - t(CP)$ where $t(W)$ is the runtime for the Weintraub *et al.* (1999) algorithm and $t(CP)$ is the runtime for the critical path algorithm. These measures are evaluated over several due date ranges where the given value is the average of 10 replications.

Smaller problems have been evaluated in order to understand the effect that the number of tabu search moves, *m*, and the number of VFA iterations, *x*, have on $L_{\max} diff$ and *tdiff*. Random problems were created with 40 jobs and 20 machines. All of these jobs are routed through the bottleneck machine and they all have alternative routes. All processing time for each operation is U[1, 8] and all jobs have a U[5, 7] distribution for the number of operations.

Figure 3 shows $L_{\max} diff$ and *tdiff* for 20 tabu moves and 20 VFA iterations. Likewise, figure 4 shows $L_{\max} diff$ and *tdiff* for 20 tabu moves and 100 VFA iterations. Figures 5 and 6 show $L_{\max} diff$ and *tdiff* for 50 tabu moves, 20 VFA iterations and 50 tabu moves, 100 VFA iterations, respectively. In each graph, $L_{\max} diff$ is shown on the left vertical axis and *tdiff* is shown on the right. Due date ranges from 0-100 are shown on the x-axis. Analysis of these graphs shows that an increase in *m* from 20 to 50 moves greatly affects $L_{\max} diff$, especially at lower due date ranges. For 50 moves, all the $L_{\max}(CP)$ averages are less than the $L_{\max}(W)$ averages. However, this is also associated with a decrease in *tdiff*, seen more clearly with the higher number of iterations, *x*. As *x* is increased from 20 iterations to 100 iterations, *tdiff* decreases significantly but $L_{\max} diff$ is

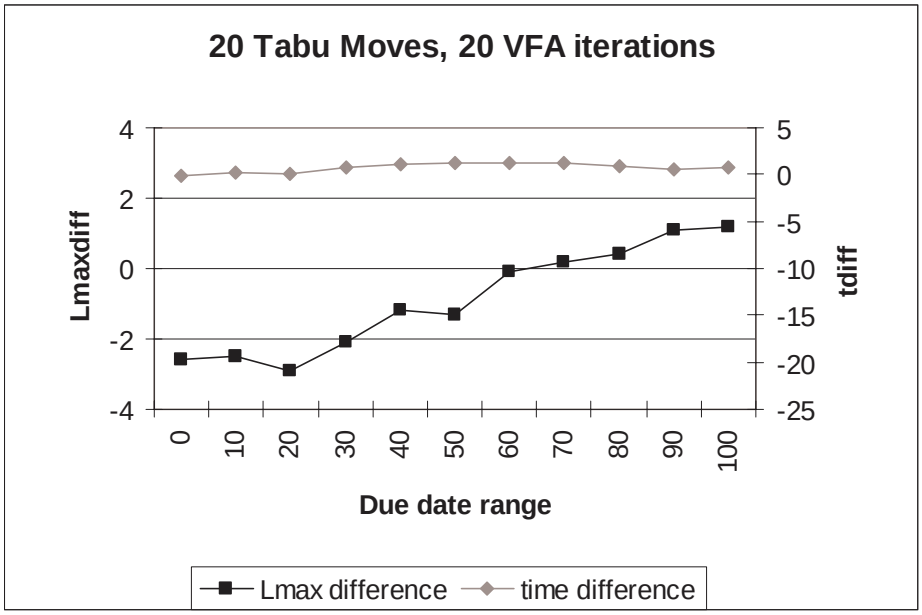

Figure 3: L_{max} *diff* and *tdiff* vs. due date range for m = 20 and x = 20

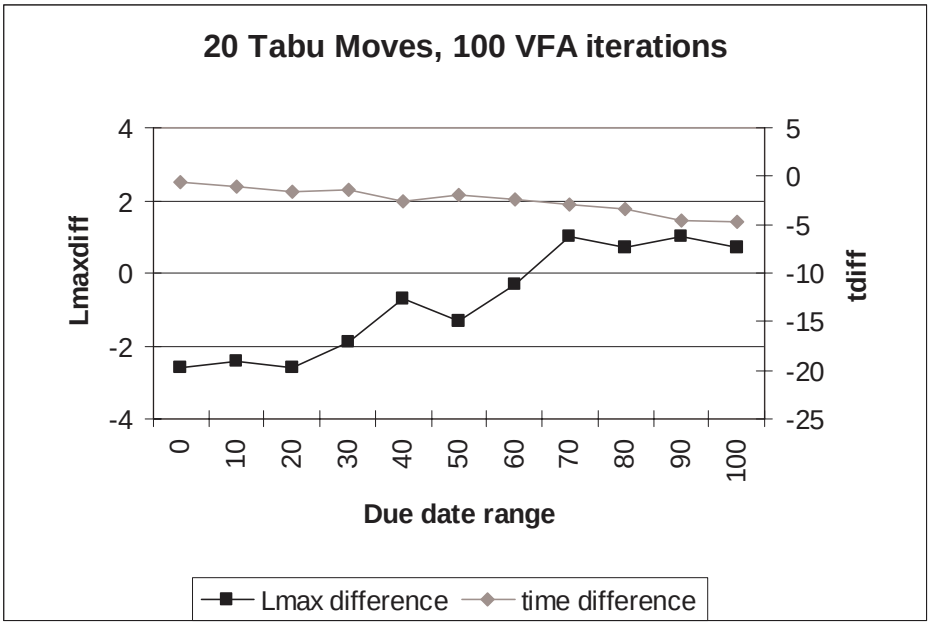

Figure 4: L_{max} *diff* and *tdiff* vs. due date range for m = 20 and x = 100

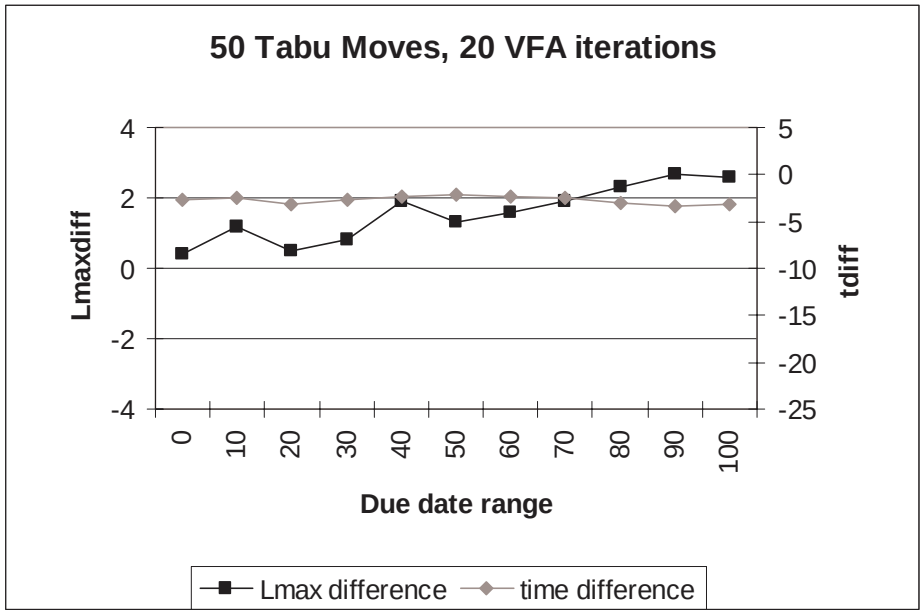

Figure 5: L_{max} *diff* and *tdiff* vs. due date range for m = 50 and x = 20

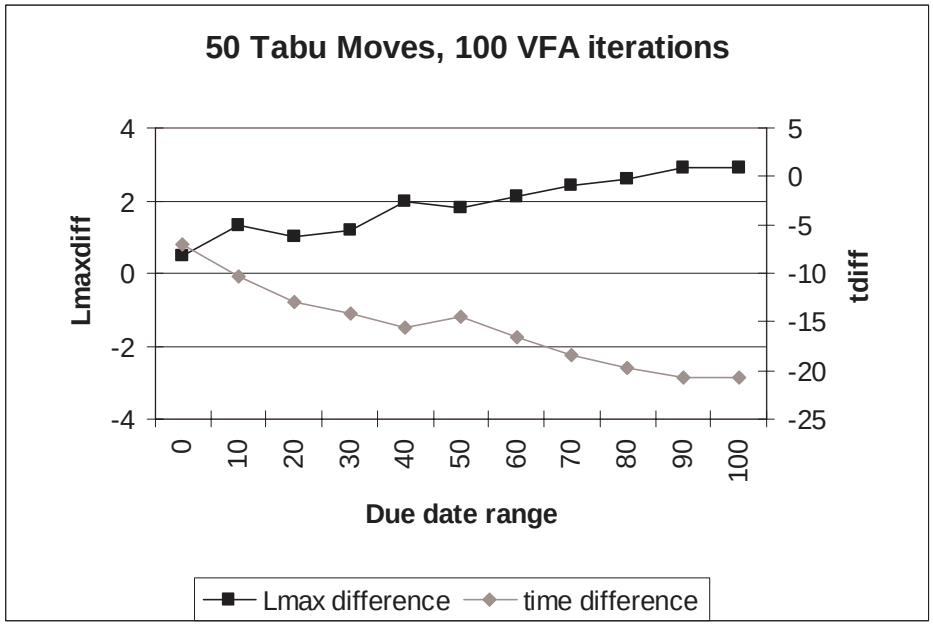

Figure 6: L_{max} *diff* and *tdiff* vs. due date range for m = 50 and x = 100

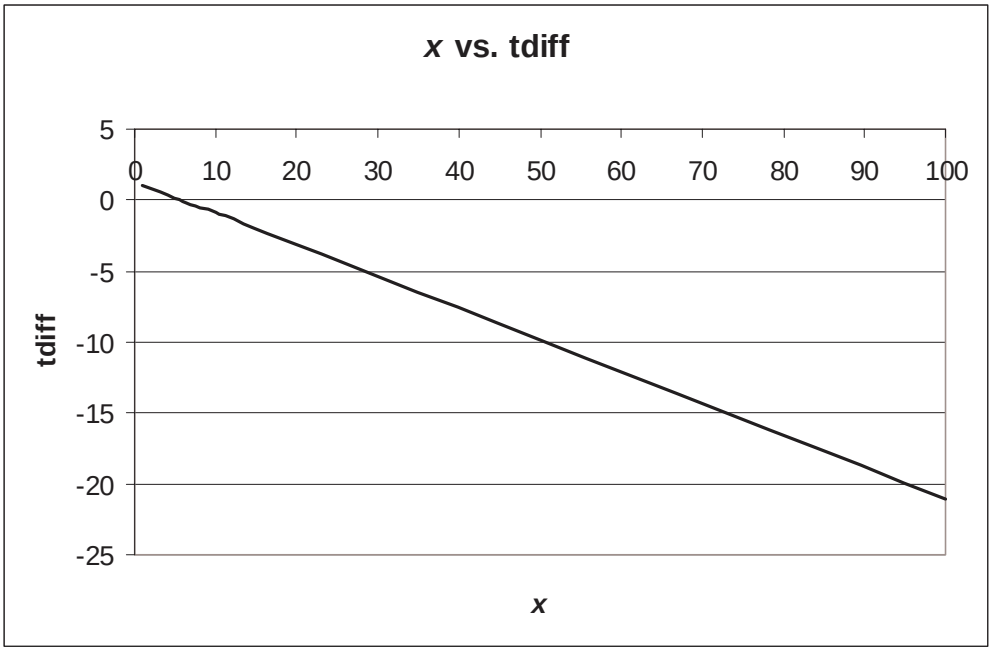

Figure 7: Number of iterations, *x*, vs. *tdiff* for *m* = 50

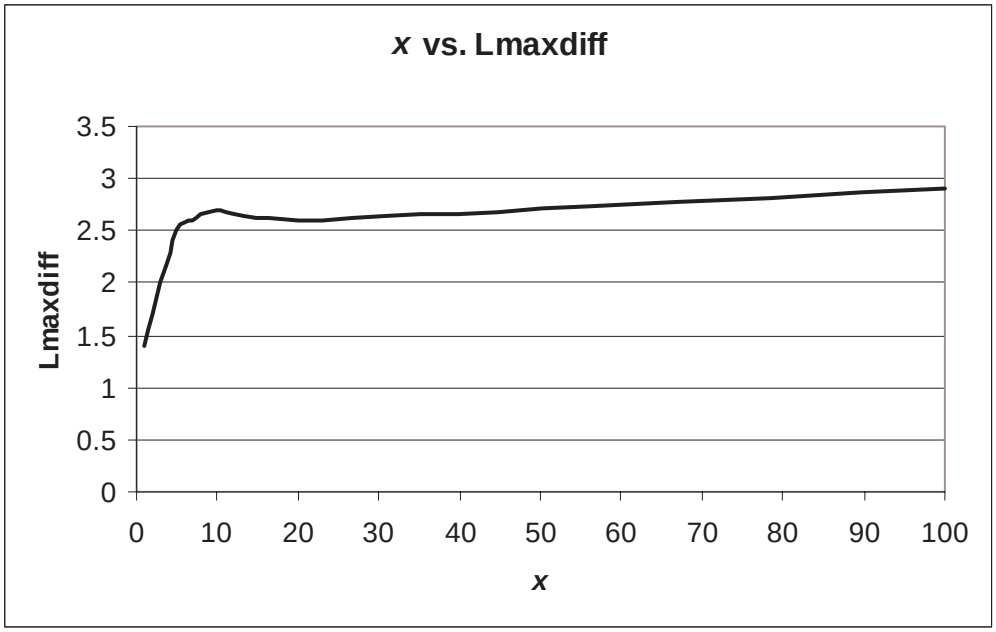

Figure 8: Number of iterations, *x*, vs. L_{max} *diff* for *m* = 50

affected very little.

For these trials, the higher number of moves and lower number of iterations is better when the goal is to maximize $L_{\max}$ *diff* and *tdiff*. In other words, raising the number of moves seems to improve the solution quality without too much increase in computation time while raising the number of iterations only serves to increase the amount of time without improving the quality of solution.

After observing this pattern, further trials were run with value of *m* ranging from 50 to 100, while values of *x* ranged from 1 iteration to 20 iterations. An increase in *m* from 50 to 100 resulted in almost identical $L_{\max}$ *diff* values, with some points showing a slight improvement. It may be noted here that in order to preserve solution quality when a higher percentage of jobs go through the bottleneck, it may be necessary to run the simulation with more moves than at lower percentage.

At *m* = 50, *tdiff* is about -2 seconds. However, *tdiff* decreases by 2 seconds, on average, when *m* is increased to 100. For that decrease in *tdiff* , the increase in $L_{\max}$ *diff* is not enough. For this reason, *m* is kept constant at 50. As seen above, a lower number of VFA iterations produces a good $L_{\max}$ *diff* while minimizing the computation time required. Therefore, *x* is varied further to determine the fewest number of iterations that will still yield an acceptable $L_{\max}$ *diff*. In Figure 7, as *x* increases, *tdiff* decreases linearly. In Figure 8, it is seen that as *x* begins to increase, $L_{\max}$ *diff* increases. However, at about 10 iterations $L_{\max}$ *diff* levels off. This may be expected since similar results were found in the original VFA analysis (Weintraub *et al*, 1999). After 10 iterations, the VFA procedure converges to a good solution.

5.1.3 Computation Time

The next major issue is that *CPA* consistently has a longer runtime than *WA*. The first step in determining the solution to this problem is to find out if $t(CP)$ is erratic or in a tight distribution. If $t(CP)$ has a small variance centered on its mean, then the best course of action is to change the mean runtime. This will allow the times of all problem instances to be relatively similar. If $t(CP)$ has a large variance then more work will be required to reduce the runtime for all problem instances, since both the mean and the variance will have to be reduced. Both $t(CP)$ and $t(W)$ are evaluated for 10 randomly generated problems at each due date range. Problems were generated with 40 jobs and 20 machines. All of the jobs are routed through the bottleneck machine and they all have alternative routes. All processing time for each operation is U[1, 8] and all jobs have a U[5, 7] distribution for the number of operations. The results are seen in Figures 9 and 10.

CPA produces very consistent results, time-wise. It shows a tight distribution centered on the average for each due date range. *WA*, however, has a much larger variation in runtimes for each replication. Since $t(CP)$ is a very tight process, meaning the variation among instances is small, finding a general way to reduce time in the algorithm should help lower the runtime for every instance.

To better understand this, each time the minimum $L_{\max}(CP)$ is updated, the time, $t(CP)$, is recorded and both values are output. This information is used to determine when the best $L_{\max}(CP)$ is found and compared to the time when the algorithm ends according to its predetermined stopping criteria. Figure 11 shows these results.

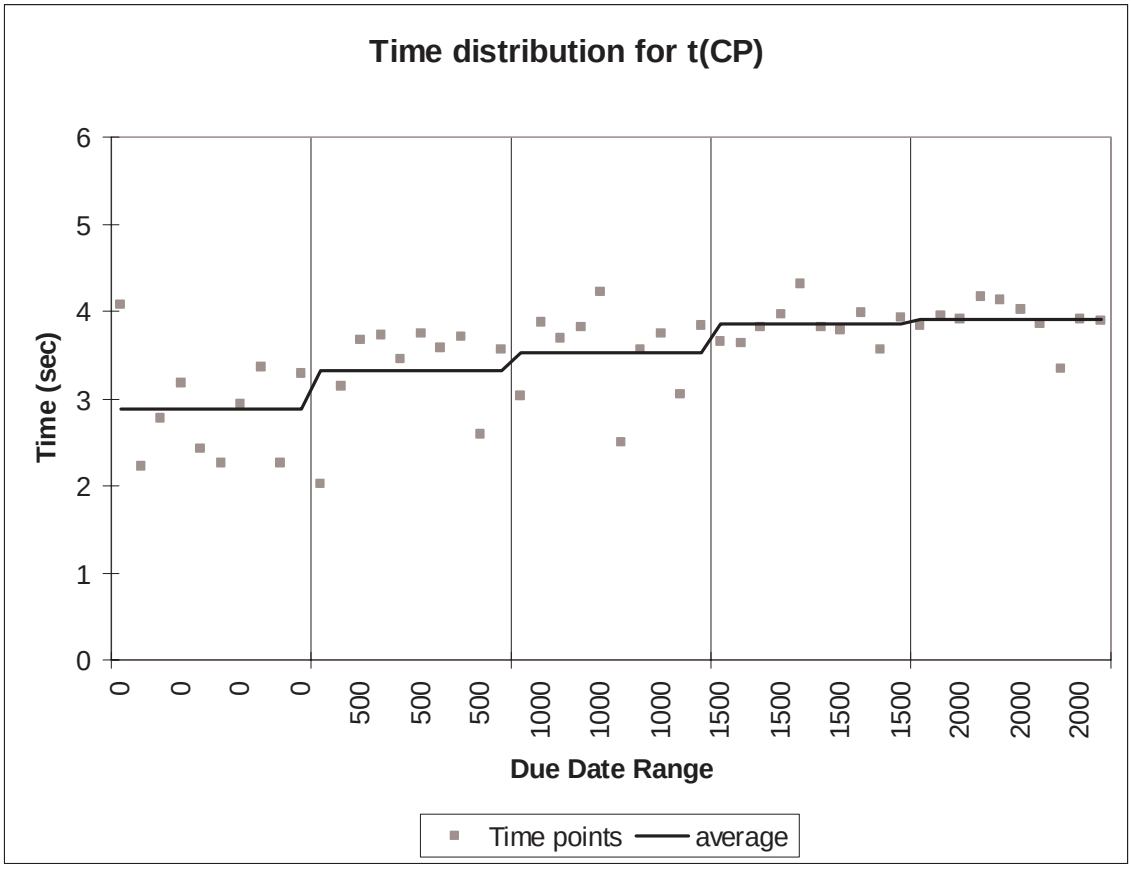

Figure 9: Time distribution for *t*(*CP*) for 40 jobs, 20 machines

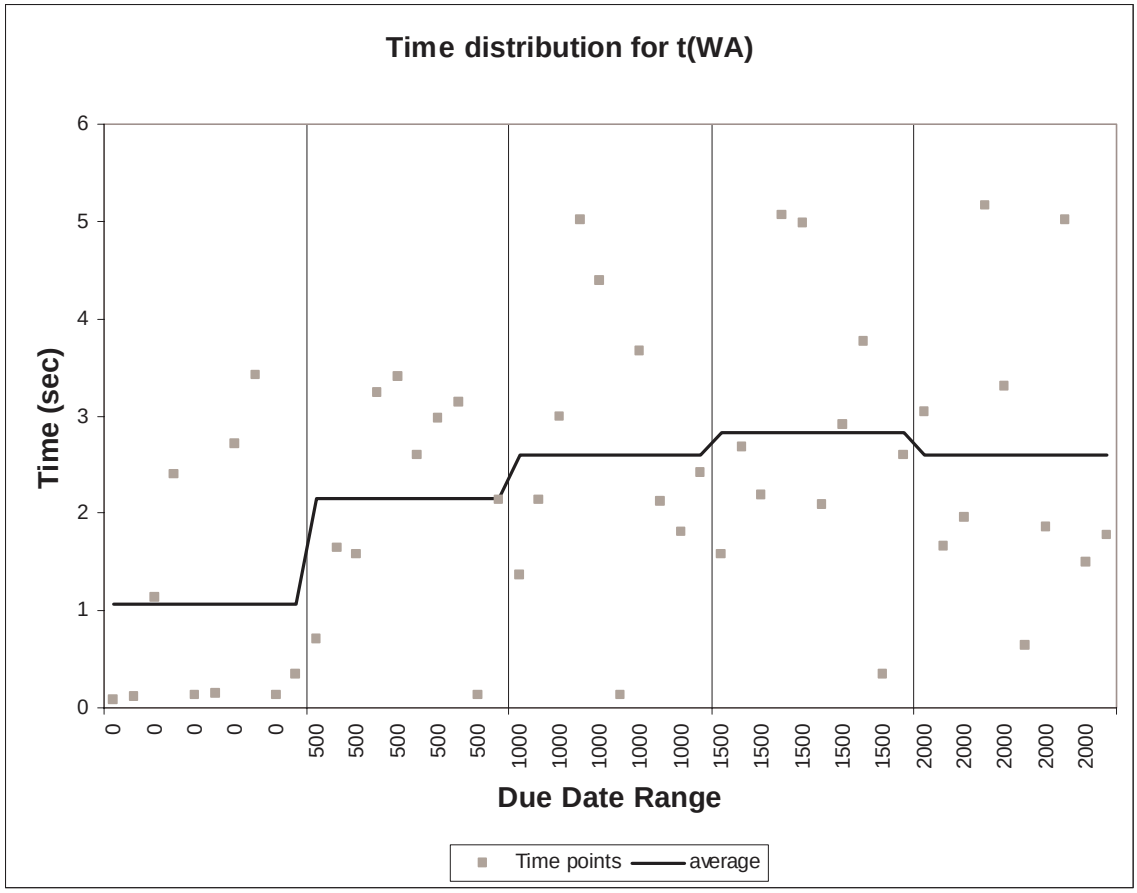

Figure 10: Time distribution for *t*(*W*) for 40 jobs, 20 machines

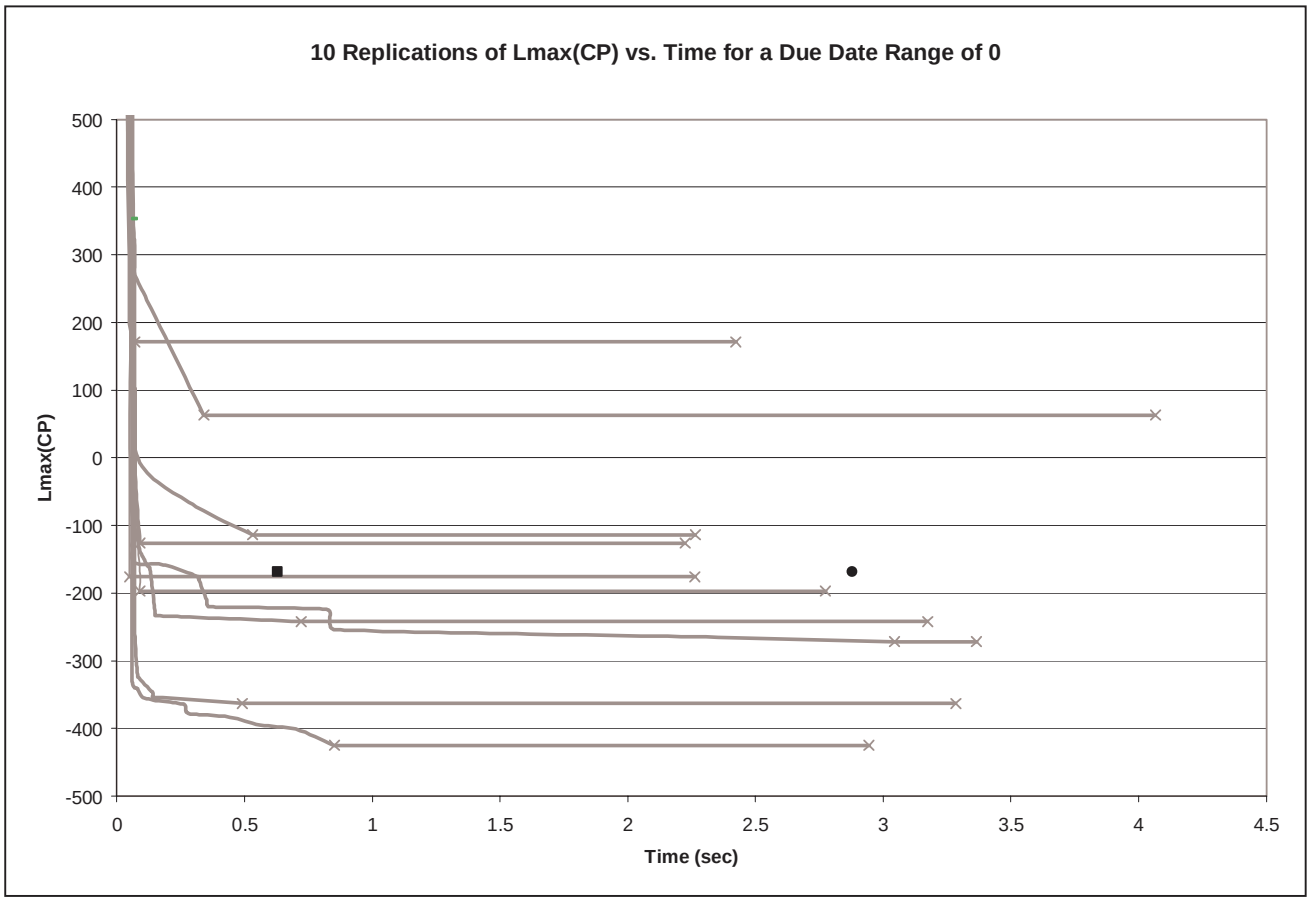

Figure 11: Ten Replications of $L_{max}(CP)$ vs. $t(CP)$ for all jobs due at time 2000

The best solutions occur several seconds before the algorithm ends. The dark single point on the right is the average time for *CPA* to run to completion. The dark single point on the left is the average time until the algorithm finds $L_{\max}(CP)$. In the worst case, the algorithm runs 45 times longer than needed to find $L_{\max}(CP)$. In the best case, it only runs 0.3 seconds longer, reaching the best $L_{\max}(CP)$ about the same time it ends. On average, *CPA* runs 16 times longer than needed.

The stopping criteria for *CPA* is based on several parameters: (1) *X*, the number of initial VFA iterations, (2) *Y*, the number of starting solutions chosen from *X* that are explored using the tabu search, (3) *m*, the number of tabu search moves made for each starting solution, (4) *x*, the number of VFA iterations made for each move, and (5) TLL, the length of the tabu list. The parameters, *m* and *x*, were examined and set above. Their impact on time was noted and as the problem size (i.e. the number of jobs and machines) increases, the number of tabu moves and VFA iterations per move are adjusted accordingly.

That leaves *X*, *Y*, and TLL that can be changed. For all initial trials, *CPA* ran 100 VFA iterations and chose the best 20 solutions to explore; and the tabu list length was chosen to be very small. These parameters are now varied and tested on problems with 40 jobs, 20 machines, 100% of the jobs passing through the bottleneck machine and all of those having an alternate route available. The processing time for each operation was U[1, 200] with each job having U[5, 7] operations. The number of initial VFA iterations, *X*, is tested at values, 10, 50 and 100; and the number of starting solutions explored, *Y*, is tested at 10 and 20. The tabu list length, TLL, is held constant as the number of jobs.

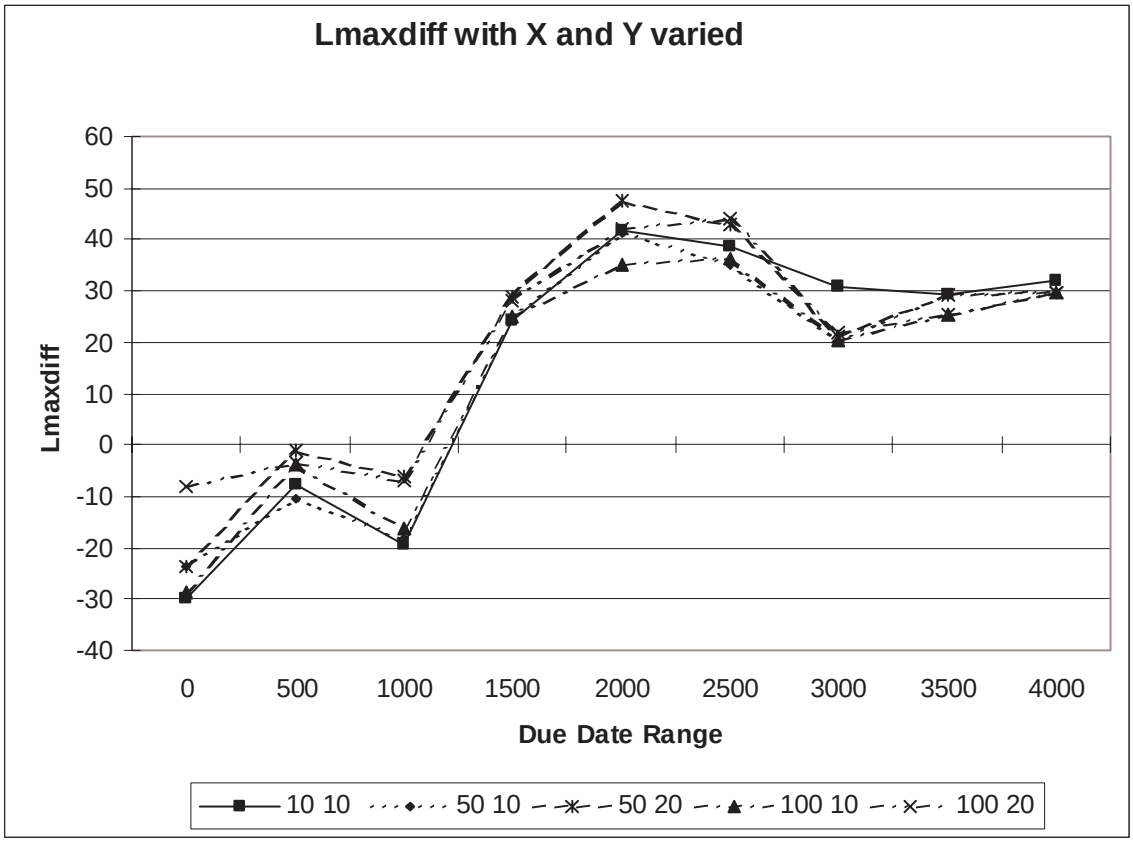

Figure 12: L_{max} *diff* with *X* and *Y* varied

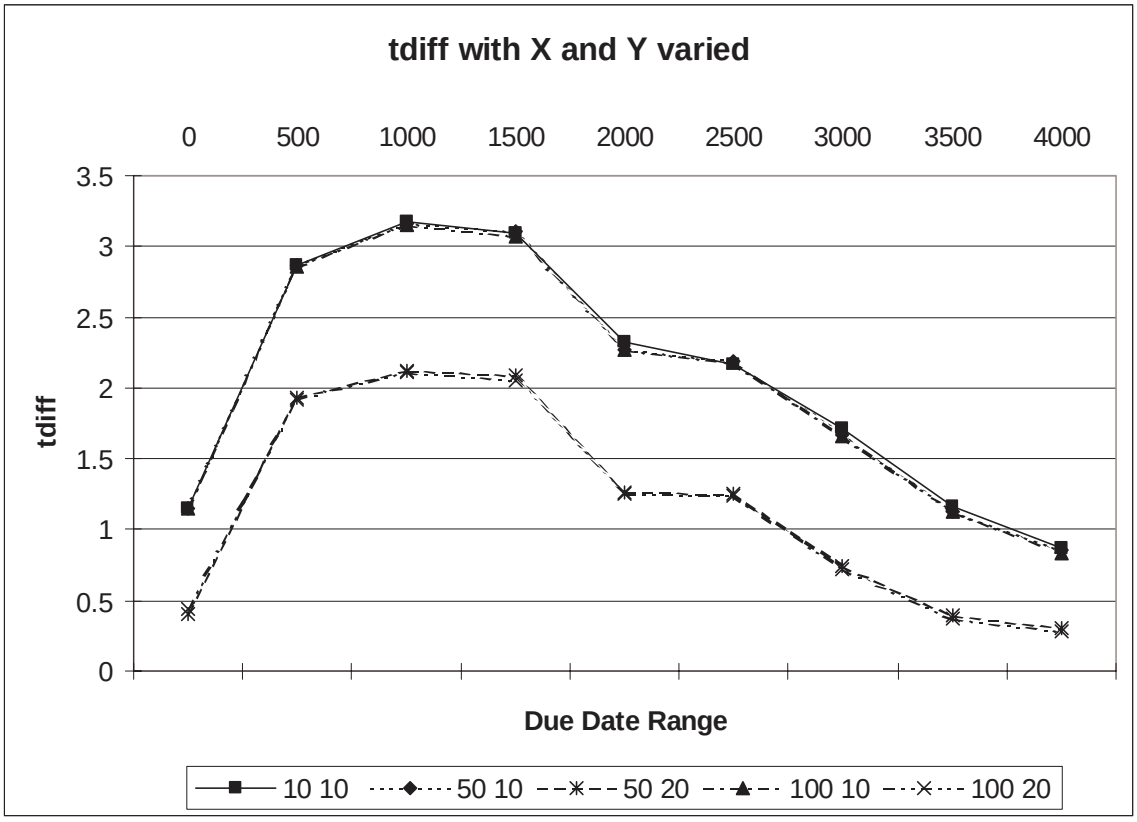

Figure 13: *tdiff* with *X* and *Y* varied

In the legend of Figure 12, the first number represents the value of *X* and the second represents the value of *Y*. As shown in both Figures 12 and 13, all of these trials yield faster results than *WA* (i.e. *tdiff* > 0), while they also result in very similar values for $L_{\max}$ *diff*. The best explanation for this is that when the tabu list length is increased, it causes the search to be more particular in its move choices. Limiting the number of possible moves limits the runtime, since the algorithm stops if all possible moves are tabu. The trials with 10 starting solutions are consistently faster than those with 20 starting solutions, as expected. Since they also produce comparable values for $L_{\max}$ *diff*, *Y* is chosen to be 10. Changing the value for *X* has little impact on $L_{\max}$ *diff* or *tdiff*, so the original value of *X* = 100 is kept.

5.2 Final Trials

The strength of the VFA is that large, industrial problems can be run quickly yielding very good sequences. Thus, the real test is to see whether the *CPA* can produce "good" results for industrial size problems in a reasonable amount of time. Problems are randomly generated for 500 jobs, 50 machines, 5 operations per job, a U[1, 200] processing time per operation, and due date ranges from 0 to 10,000 in increments of 500. The percentage of jobs that pass through the bottleneck machine is tested at 100% and 20%. Of those going through the bottleneck, all have alternative routes available. Each parameter set was run on 10 randomly generated problems.

In order to better compare the *CPA* with the *WA*, the same parameters are run for each algorithm. The given set of parameters is first run using the *WA*. The algorithm is

run for 1000 *VFA* iterations and 100 starting solutions. The runtime, $t(W)$, is recorded, as well as the number of neighbors that the algorithm searches. Then the *CPA* is run for the same amount of time while varying the number of starting solutions and tabu search moves.

5.2.1 Trials using 100% of Jobs Passing Through Bottleneck Machine

Table 1: Summary of Results for 100% of jobs through the bottleneck machine

	Time	No. Nbrs	No. Iterations	No. SS	No. Moves	% w/better Lmax
WA	0.4658	101	20	-	-	**-**
CPA (1SS)	2.40	497	101	1	1	**0.00%**
CPA (2SS)	2.32	500	101	1	1	**0.00%**
CPA (5SS)	2.31	500	101	1	1	**0.00%**

The first set of trials are run using 100% of the jobs passing through the bottleneck machine and 100% of those having alternative routes. The results shown in Table 1 reflect the average of all runs and all due date ranges for the given set of parameters. When *WA* is tested, it runs, on average, 0.4658 seconds and searches 101 neighbors. The total number of *VFA* iterations during the run is 20 on average. This means that the *WA* is reaching its lower bound before completing all 1000 iterations of the *VFA*.

The *CPA* is then run for the same parameters using 1 starting solution and a large number of moves. For the initial run, 500 moves were chosen. The algorithm is allowed to run for one second or until the end of its current tabu move, whichever is longer. As seen in Table 1, on average, these parameters cause the algorithm to run for 2.40 seconds. It searches 497 neighbors and runs for 101 *VFA* iterations. This run is inferior to that of *WA* in each due date range.

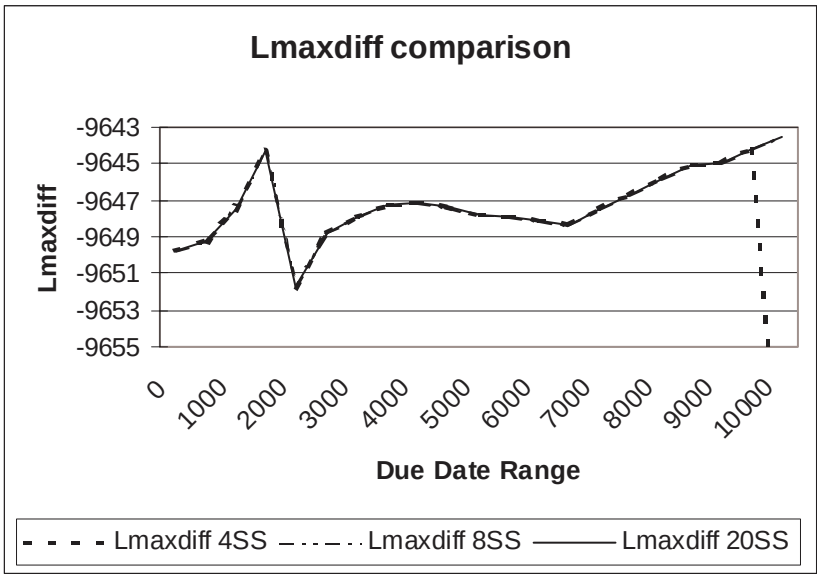

Figure 14: L_{max} *diff* vs. Due Date Range for 1, 2 and 5 starting solutions and 100% of jobs through the bottleneck

The L_{max} *diff* values for each of the three scenerios, 1, 2 and 5 starting solutions can be seen in figure 14. All three data sets follow the same pattern. At first these results do not seem to be consistent with those found in earlier trials. However, expanding the search area by increasing the problem size causes the *CPA* to take longer to find a good solution. In the above trials, the search time is limited, which limits the number of neighbors that the *CPA* can search. Table 1 shows that the algorithm ended on the first move every time. This means that the algorithm looked at every neighbor of one solution (maximum of 500 neighbors), moved to the best neighbor and was forced to stop due to the time constraint. The same results are found when the number of starting solutions is increased to 2 starting solutions and 5 starting solutions.

Further analysis reveals that since all jobs pass through the bottleneck, most, if not all, of the jobs will also be included in the critical path. The search space, then, is proportional to the number of jobs in the problem.

In comparison, the *WA* has a neighborhood that searches all of the available neighbors for each starting solution searched, 500 neighbors for each of 100 starting solutions in this case. The new *CPA* searches its critical path neighborhood every time a move is made. The total number of moves can be calculated as (the number of starting solutions) * (the number of tabu moves per solution). The total number of neighbors searched is therefore, (the total number of moves) * (the number of jobs). This can get very large when the number of jobs is large. This fundamental difference in the two algorithms causes the *CPA* to run longer than *WA* when larger problems are studied.

5.2.2 Trials using 25% of Jobs Passing Through Bottleneck Machine

After observing the algorithm's behavior for 100% of jobs passing through the bottleneck, the next step is to lower that percentage and analyze the results. The same sets of parameters are again run, however this time only 25% of the jobs go through the bottleneck. All 100% of these have alternative routes.

Table 2: Summary of Results for 25% of jobs through the bottleneck machine

	Time	No. Nbrs	No. Iterations	No. SS	No. Moves	% w/better Lmax
WA	34.18	4877	5420	-	-	**-**
CPA (4SS)	30.96	12893	710	4	30	**4.76%**
CPA (8SS)	34.71	2791	141	2	8	**0.00%**
CPA (20SS)	34.68	3057	136	6	3	**0.00%**

The *WA* is run and takes, on average, 34 seconds per replication. Using this as the time limit, *CPA* is run for three different scenarios. To determine the initial number of starting solutions to run, the number of starting solutions was increased from 1 until *CPA* ran for 34 seconds. This happened after 4 starting solutions; therefore, the first run of the *CPA* was with 4 starting solutions. For the initial run a large number of moves, 500, was chosen. After analyzing the results of this run, the algorithm was run again, but with twice as many starting solutions. The number of tabu moves was chosen to be half of the average number of moves found when running 4 starting solutions. This was repeated for 5 times the number of starting solutions and one-fifth of the initial trial's average number of tabu moves. Figure 15 and Figure 16 show these results.

Figure 15 shows L_{max} *diff* for the three different scenarios, while Figure 16 shows *tdiff*. Only in one instance, the 9500 due date range, does *CPA* produce a better L_{max}

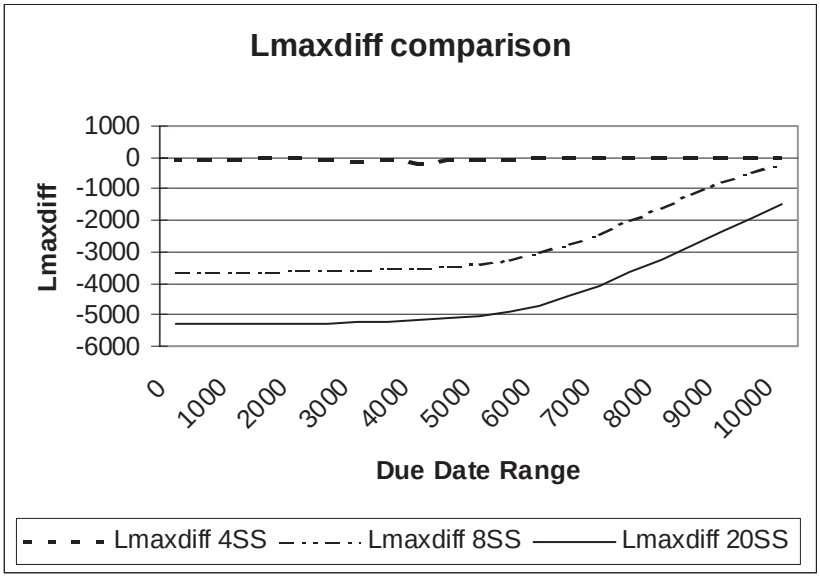

Figure 15: L_{max} *diff* vs. Due Date Range for 4, 8 and 20 starting solutions and 25% of jobs through the bottleneck

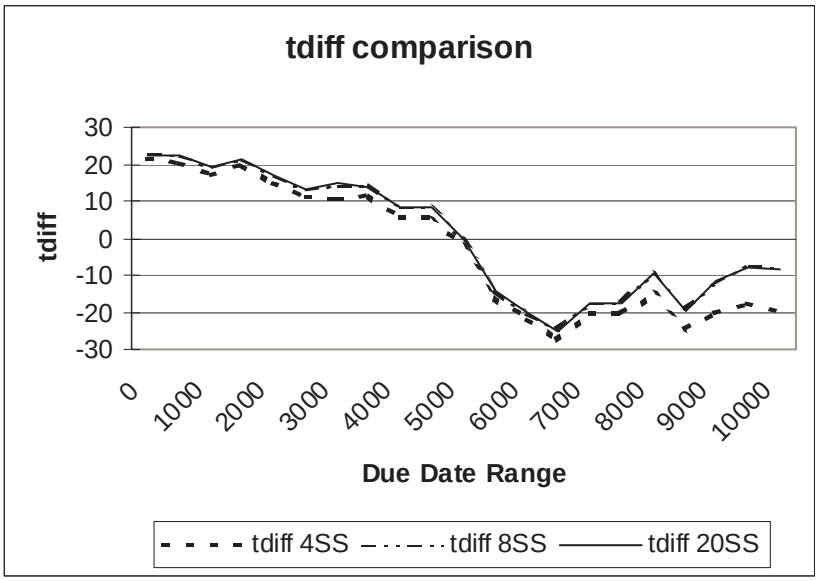

Figure 16: *tdiff* vs. Due Date Range for 4, 8 and 20 starting solutions and 25% of jobs through the bottleneck

than *WA* when 4 starting solutions (4SS) are used. When the number of starting solutions is raised and the number of moves is lowered, the results are much worse. *WA* outperforms *CPA* every time.

Table 2 shows the results for this experiment in a similar fashion to Table 1. While the average time between algorithms is very similar, *WA* searches more neighbors than *CPA* for every instance except when 4 starting solutions are used. This can be accounted for by noting that *CPA* must limit the number of neighbors it searches due to the time constraint. When 4 starting solutions are used, *CPA* requires more tabu moves per starting solution. By doing this, the solution space is searched more in depth for each starting solution; however, the search may not be expanding into the necessary areas of the space.

Chapter 6

Conclusions and Future Research

6.1 Thesis Summary and Conclusions

The experimentation in this thesis is related to research done by Weintraub *et al* (1999). In that paper, a tabu search algorithm was created in order to select job routings for jobs to determine the lowest possible L_{max}. To improve on that algorithm, this research must solve the same problem yielding better L_{max} results, or yielding the same L_{max} results in less time. The original conjecture was that reducing the search neighborhood by using the critical path approach will produce these results.

The initial trials were done using small problems, 40 jobs, 20 machines, 5-7 operations per job, and U[1, 8] hours processing time per operation. All jobs were routed through the bottleneck machine and had alternative routes available. For these problems, the *CPA* produces mixed results. For almost all cases, the new algorithm finds better L_{max} values than the original *WA*. However, it takes longer to find these values than the earlier algorithm. After varying the number of starting solutions, even better numbers are reported. The *CPA* almost always has a shorter runtime than *WA*. When the due date range is medium to large, *CPA* outperforms *WA* in solution quality.

The results change, however, when the problem size is increased. The large problems were run with 500 jobs, 50 machines, 5 operations per job and U[1, 200] processing time per operation. When all jobs pass through the bottleneck machine and have alternative routes available, *WA* performs better than *CPA*.

The analysis shown in this thesis leads to an alternative conjecture. *CPA* performs better than *WA* for very small problems since *CPA* investigates a lot more lower bounds of neighboring solutions. The solution time for *CPA* is smaller since *WA* spends a long time running its *VFA* iterations, as compared to the time it takes to evaluate its neighbors. The same conjecture can be made for larger problems except that the *CPA* solution time is much larger when the number of jobs increases. If allowed to run to longer, *CPA* may produce better L_{max} results at the expense of more computation time.

6.2 Future Research

6.2.1 Conditions for a Lower Bound

During the course of the research for this thesis, several observations were made about conditions in which it can be known that a minimum lower bound has been reached.

1. If the L_{max} job starts at its earliest start time on the L_{max} machine (the machine where the L_{max} job queues the longest), is not pre-empted, and uses its shortest processing times for all upstream operations, then the minimum lower bound has been found.
2. If all jobs in the L_{max} candidate group have their alternative operation on some other machine and if the alternative operation is earlier in the sequence than on

the L_{max} machine and the shorter sequence is in use, then the minimum lower bound is found.

Note that by condition 2, for the case of alternative operations on the critical machine only, the minimum lower bound is found when all jobs use their primary route since their alternative routes are longer than their primary route.

These conditions were observed during the current research, but they were not proven or implemented. After proving the validity of these statements, incorporating these conditions into the *CPA* could help shorten the runtime by allowing the lower bound search to stop when it is known that a minimum lower bound has been reached. Since a large part of the calculations for the algorithm are those involved in finding the lower bounds of neighbors, the addition of these conditions could have a great impact on computation time.

6.2.2 Generate Problems using Other Assumptions

The problems generated for this thesis all involve jobs in a job shop traveling through a bottleneck machine. A change from the bottleneck machine to an alternate machine or machines is the only type of alternative route studied. As seen in the section 6.1, the new *CPA* created does not always perform well using these alternative routing assumptions.

Several varying assumptions have been studied using the *WA*. The first assumption is that the ordering of consecutive operations within a route may be switched. The second assumption is that an alternative route can be entirely different from the original route. In this case, the alternative route is randomly generated; and it may or may not have operations in common with the original route.

Running new trials using *CPA* on these alternate assumptions may yield better results as compared with *WA*. The main problem of the current research is that in order to find good solutions, too much computation time is required. This happens because the number of neighbors to be searched is proportional to the problem size. The hypothesis is that when the underlying bottleneck machine assumption is changed, this will also change the neighborhood size. The number of neighbors will no longer be directly proportional to the problem size, and a smaller neighborhood requires less time to search. Then an additional question can be answered. Does the *CPA* truly search a better neighborhood than the *WA*?

6.2.3 Increase Runtime of Search with 25% through the Bottleneck

For this research, although almost all of the trials were run with 100% of jobs passing through the bottleneck, the final trials were done with only 25% of jobs using the bottleneck machine. However, these experiments did not allow the *CPA* to run to completion, *i.e.* to the end of the specified number of moves for each starting solution. Instead, the $L_{\max}$*diff* was evaluated after the program had run for a given number of seconds. The purpose of these experiments was to compare the quality of solution between *WA* and *CPA* when the computation time was held constant.

Even though computation time is an important performance measure, it is not always a limiting factor. In other words, while some situations require a schedule in a very short amount of time, others do not. In light of this, further trials should be run with 25% of the jobs passing through the bottleneck machine. These trials should allow the algorithm to run to completion and compare the results found with the results of the *WA*.

Studies using other, varied bottleneck percentages should also be performed and compared to those reported in this paper. That analysis will show how sensitive *CPA* is to changing the percentage of jobs going through the one bottleneck machine in a job shop.

Also, more in-depth studies of the relationship between the number of moves and the percentage of jobs on the bottleneck should be studied. Since *m*, the number of tabu moves per starting solution, is a large factor contributing to the computation time of the algorithm, varying this while varying the percentage through the bottleneck will give great insight into the strength and speed of the algorithm.

6.2.4 Revised Tabu Search within Lower Bound

The tabu search algorithm developed in this paper switches between performing computations in the lower bound when evaluating neighbors and computing the actual L_{max} through the *VFA* when determining the critical path for a solution. Revising the algorithm to involve only lower bound calculations may help limit the computation time required to run *CPA*. A new critical path definition would have to be created using only the job schedule and sequence found while evaluating the lower bound.

The new lower bound algorithm would be as follows. For each starting solution, the algorithm would run for *m* tabu moves computing the best neighbor and the critical path using only the lower bound calculation. After the last move for a given starting solution, the best *v* lower bound solutions found would be input into the *VFA*. When the previous procedure had been run for all starting solutions, the best overall *VFA* solution would be output.

Bibliography

1. Candido, M.A.B., S.K. Khator, and R.M. Barcia (1998) A genetic algorithm based procedure for more realistic job shop scheduling problems, *International Journal of Production Research*, **36** (12), 3437-3457.
2. Carlier, J. and E. Pinson (1989) An algorithm for solving the job shop problem. *Management Science*, **35**(2), 164-176.
3. Caroll, D.C. (1965) Heuristic sequencing of single and multiple component jobs, Unpublished Ph.D. Dissertation, Massachusetts Institute of Technology, USA.
4. Choi, I.-C. and D.-S. Choi (2002) A local search algorithm for job shop scheduling problems with alternative operations and sequence-dependent setups, *Computers & Industrial Engineering*, **42**(1), 43-58.
5. Chryssolouris, G. and V. Subramaniam (2000) Dynamic scheduling of manufacturing job shops using extreme value theory, *Production Planning and Control*, **11**(2), 122-132.
6. Chryssolouris, G. and V. Subramaniam (2001) Dynamic scheduling of manufacturing job shops using genetic algorithms, *Journal of Intelligent Manufacturing*, **12,** 281-293.
7. Dauzère-Pérès, S. and J. Paulli (1997) An integrated approach for modeling and solving the general multiprocessor job-shop scheduling, *Annals of Operations Research* **70**, 281-306.
8. Hodgson, T.J., D. Cormier, A.J. Weintraub, and A. Zozom (1998) Satisfying due-dates in large job shops. *Management Science*, **44**(10), 1442-1446.
9. Hodgson, T.J., R.E. King, K.A. Thoney, N. Stanislaw, A.J. Weintraub, and A. Zozom (2000) On satisfying due-dates in large job shops: idle time insertion. *IIE Transactions*, **32**(2), 177-180.
10. Kim, K.-H. and P.J. Egbelu (1999) Scheduling in a production environment with multiple process plans per job, *International Journal of Production Research*, **37**(12), 2725-2753.
11. Kim, Y.K., K. Park and J. Ko (2002) A symbiotic evolutionary algorithm for the integration of process planning and job shop scheduling, *Computers & Operations Research*, In Press, Uncorrected Proof.
12. Kolisch, R. and K. Hess (2000) Efficient methods for scheduling make-to-order assemblies under resource, assembly area and part availability constraints, *International Journal of Production Research*, **38**(1), 207-228.
13. Lawrence, S.R. and T.E. Morton (1986) Patriarch: hierarchical production scheduling. *National Bureau of Standards Special Publication*, 724, pp. 87-97.

14. Lee, Y.H., C.S. Jeong and C. Moon (2002) Advanced planning and scheduling with outsourcing in manufacturing supply chain, *Computers & Industrial Engineering*, **43**(1-2), 351-374.

15. Nowicki, E. and C. Smutnicki (1996) Fast taboo search algorithm for the job shop problem, *Management Science*, **42**, 797-813.

16. Panwalkar, S.S., R.A. Dudek, and M.L. Smith (1973) Sequencing research and the industrial problem. In *Symposium on the Theory of Scheduling* (Berlin, Springer-Verlag).

17. Raman, N. (1995) Input control in job shops. *IIE Transactions*, **27**(2), 201-209.

18. Saygin, C., F.F. Chen, and J. Singh (2001) Real-time manipulation of alternative routings in flexible manufacturing systems: A simulation study, *International Journal of Advanced Manufacturing Technology*, **18**(10), 755-763.

19. Schultz, S.R., T.J. Hodgson, R.E. King, and K.A. Thoney (2002) Technical note: On minimizing $L_{\max}$ for large-scale job shop scheduling problems, Department of Industrial Engineering, North Carolina State University, Working paper.

20. Smith, W.E., R. Ramesh, R.A. Dudek, and E.L. Blair (1986) Characteristics of U.S. flexible manufacturing systems - a survey. In *Proceedings of the Second ORSA/TIMS Conference on Flexible Manufacturing Systems*, Ann Arbor, Michigan, USA.

21. Thomalla, C.S. (2001) Job shop scheduling with alternative process plans, *International Journal of Production Economics*, **74** (1-3), 125-134.

22. Thoney, K.A. (2000) Simultaneous plant and supply chain scheduling. unpublished Ph.D. dissertation, Department of Industrial Engineering, North Carolina State University, USA.

23. Thoney, K.A., T.J. Hodgson, R.E. King, M.R. Taner, and A.D. Wilson (2002) Satisfying due-dates in large multi-factory supply chains. *IIE Transactions*, **34**(9), 803 – 811.

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

25. Weintraub, A.J., D. Cormier, T.J. Hodgson, R.E. King, J.R. Wilson, and A. Zozom (1999) Scheduling with alternatives: A link between process planning and scheduling, *IIE Transactions*, **31**(11), 1093-1102.