MODELING SCHEDULING ALGORITHMS WITH ALTERNATIVE PROCESS PLANS IN AN OPTIMIZATION PROGRAMMING LANGUAGE

A thesis presented to
the faculty of the
Fritz J. and Dolores H. Russ College of Engineering and Technology of Ohio University

In partial fulfillment
of the requirements for the degree
Master of Science

Ramachandra Sharma Harihara

November 2004
This thesis entitled

MODELING OF SCHEDULING ALGORITHMS WITH ALTERNATIVE PROCESS PLANS IN AN OPTIMIZATION PROGRAMMING LANGUAGE

BY

Ramachandra Sharma Harihara

has been approved for

Department of Industrial and Manufacturing Systems Engineering

and the Russ College of Engineering and Technology by

Dušan N. Šormaz
Associate Professor of Industrial and Manufacturing Systems Engineering

Dennis Irwin
Dean, Fritz J. and Dolores H. Russ College of Engineering and Technology
HARIHARA, RAMACHANDRA SHARMA. M.S. November 2004. Industrial and Manufacturing Systems Engineering

Modeling of Scheduling Algorithms with Alternative Process Plans in an Optimization Programming Language (89pp.)

Director of Thesis: Dušan N. Šormaz

This thesis optimizes scheduling functions with alternative processes for manufacturing features. The problem considered is an N job – M methods problem, where each feature can be processed in up to M methods. Linear and polynomial models have been built in Optimization Language (OPL) to optimize the make span without setup times. The user is provided an interface in Java to enter the system parameters, generate the data file, choose the OPL model and execute the model. This thesis concludes that in the absence of heuristics, linear model performs better than a non-linear model for the same problem and combining system parameters intelligently gives a better chance of solving the problem faster.

Approved:

Dušan Šormaz

Associate Professor of Industrial and Manufacturing Systems Engineering
ACKNOWLEDGEMENTS

No achievement of mine is complete without the love and continuous support of my father, Mr. S.H Sharma, my mother, Mrs. R. Samyuktha, and my brother Mr. Suraj Sharma. I take this opportunity to thank them for everything that they have provided me all these years, which has made me the person I am today. Academically and personally, I was very fortunate to have Dr. Dušan Šormaz as my Graduate Advisor. Undoubtedly, it was his guidance and technical insights that enabled me complete this thesis successfully. Special thanks to my committee members, Dr. Dale Masel and Dr. Bhavin Mehta, for their important and valuable suggestions towards my thesis. I would also like to thank the Industrial and Manufacturing Systems Engineering Department at Ohio University for providing me with the much needed resources to successfully complete this thesis. Finally, I would like to thank my fellow researchers Mr. Jaikumar Arumugam, Mr. Srinivas N. Rajaraman, Mr. Vikesh B. Jain, Mr. Sam S. Kolli, Mr. Ajith Wadatkar, Mr. Deepak Pissippatti and my friends Mr. Ganesh Nagarajan, Mr. V. Ramaswamy, Mr. R. Nagapan, Mr. Thiagarajan Ramamoorthy, Mr. Capton Siluvairajan and Mr. Balajee Thangavelu for their invaluable support and motivation throughout.
# TABLE OF CONTENTS

1 INTRODUCTION ............................................................................................................. 9
1.1 MOTIVATION ............................................................................................................. 13
1.2 OBJECTIVE ............................................................................................................. 15
1.3 OVERVIEW OF APPROACH ................................................................................. 17

2 LITERATURE REVIEW ............................................................................................. 19
2.1 MATHEMATICAL MODELING OF SCHEDULING FUNCTIONS ......................... 19
2.2 INTEGRATED PROCESS PLANNING AND SCHEDULING .................................. 22
2.3 SCHEDULING WITH ALTERNATIVES ................................................................. 29
2.4 OPL AND ILOG OPL STUDIO ............................................................................ 37
2.5 IMPLANNER ............................................................................................................ 39

3 SYSTEM ARCHITECTURE ......................................................................................... 41
3.1 OVERVIEW OF THE SYSTEM .............................................................................. 41
3.2 SYSTEM COMPONENTS ......................................................................................... 42
  3.2.1 Scheduling system model .................................................................................. 42
  3.2.2 System information in XML file ......................................................................... 43
  3.2.3 Solver interface .................................................................................................. 44
  3.2.4 OPL solver ........................................................................................................ 44
  3.2.5 Building mathematical models .......................................................................... 45

4 IMPLEMENTATION ................................................................................................... 46
4.1 MATHEMATICAL MODELING – A GENERIC SCHEDULING MODEL ............... 46
4.2 POLYNOMIAL MIXED INTEGER PROGRAMMING MODEL ............................. 47
  4.2.1 Notation ............................................................................................................. 48
  4.2.2 Fixing variables and data instances .................................................................. 49
  4.2.3 Process planning and scheduling objective function ........................................ 51
  4.2.4 Sequence constraints and non-overlapping constraints for parts without sequence constraints ................................................................. 51
  4.2.5 Non-overlapping constraint for features to be machined on the same machine ............................................................................................................. 55
  4.2.6 Process plan selection constraint ...................................................................... 57
4.3 LINEAR MIXED INTEGER PROGRAMMING MODEL (LMIPM) ......................... 57
  4.3.1 Linearization of polynomial model by avoiding cross product terms .............. 57
  4.3.2 Linearizing logical decisions in constraints ...................................................... 58
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.4</td>
<td>USER INTERFACE</td>
<td>61</td>
</tr>
<tr>
<td>4.4.1</td>
<td>Interface components</td>
<td>61</td>
</tr>
<tr>
<td>4.5</td>
<td>STEPS TO RUN THE SYSTEM AND GET THE FINAL SOLUTION</td>
<td>64</td>
</tr>
<tr>
<td>4.5.1</td>
<td>Reading the XML file containing system information</td>
<td>65</td>
</tr>
<tr>
<td>4.5.2</td>
<td>Selecting and displaying model file</td>
<td>66</td>
</tr>
<tr>
<td>4.5.3</td>
<td>Generating data file</td>
<td>69</td>
</tr>
<tr>
<td>4.5.4</td>
<td>Executing the model and displaying results</td>
<td>70</td>
</tr>
<tr>
<td>5</td>
<td>TESTING</td>
<td>71</td>
</tr>
<tr>
<td>5.1</td>
<td>AN EXAMPLE</td>
<td>71</td>
</tr>
<tr>
<td>5.2</td>
<td>TESTING LINEAR MODEL</td>
<td>73</td>
</tr>
<tr>
<td>5.3</td>
<td>TESTING RESULTS FOR LINEAR MODEL</td>
<td>74</td>
</tr>
<tr>
<td>5.4</td>
<td>GRAPHS FOR LINEAR MODEL</td>
<td>78</td>
</tr>
<tr>
<td>5.5</td>
<td>TESTING RESULTS FOR NON-LINEAR MODEL</td>
<td>80</td>
</tr>
<tr>
<td>6</td>
<td>CONCLUSION AND FUTURE WORK</td>
<td>83</td>
</tr>
<tr>
<td>6.1</td>
<td>EFFECT OF ALGORITHM ON SOLUTION TIME</td>
<td>83</td>
</tr>
<tr>
<td>6.2</td>
<td>EFFECT OF SYSTEM PARAMETERS ON SOLUTION TIME</td>
<td>84</td>
</tr>
<tr>
<td>6.3</td>
<td>FUTURE WORK DIRECTION</td>
<td>85</td>
</tr>
<tr>
<td>6.3.1</td>
<td>Improved non-linear model</td>
<td>85</td>
</tr>
<tr>
<td>6.3.2</td>
<td>Other types of problems and system parameters</td>
<td>85</td>
</tr>
<tr>
<td>6.3.3</td>
<td>Distributed solving of large problems</td>
<td>86</td>
</tr>
<tr>
<td>7</td>
<td>REFERENCES</td>
<td>87</td>
</tr>
</tbody>
</table>
LIST OF TABLES

Table 4-1 Solver interface components – I........................................63
Table 4-2 Solver interface components – II......................................64
Table 5-1 Solution times of linear model........................................74
Table 5-2 Objective function for linear model.................................76
Table 5-3 Solution times of non-linear model.................................81
LIST OF FIGURES

Figure 3-1 System architecture ................................................................. 41
Figure 4-1 Fixing decision variables and data instances in OPL .................. 49
Figure 4-2 Process planning and scheduling objective function ................. 51
Figure 4-3 Sequence constraints and non-overlapping constraints ............... 52
Figure 4-4 Sequence constraint ................................................................. 53
Figure 4-5 Non-overlapping constraints for parts without sequence constraint . 54
Figure 4-6 Non-overlapping constraints of type II – OPL formulation .......... 55
Figure 4-7 Non-overlapping constraints of type II ..................................... 56
Figure 4-8 Linearized formulation of objective function ............................ 58
Figure 4-9 Linearized non-overlapping constraints of type I ....................... 59
Figure 4-10 Linearized formulation of non-overlapping constraints of type II .... 60
Figure 4-11 Solver interface ..................................................................... 62
Figure 4-12 System information read from an XML file ............................ 66
Figure 4-13 Selecting a model file ............................................................ 67
Figure 4-14 Displaying model file ............................................................. 68
Figure 4-15 Generating and displaying data file ......................................... 69
Figure 4-16 Displaying results .................................................................. 70
Figure 5-1 Processing times of features ................................................... 72
Figure 5-2 Starting times of features ......................................................... 72
Figure 5-3 Gantt chart for 3x3x3 example problem .................................... 73
Figure 5-4 Solution time Vs Problem size (with precedence) ...................... 79
Figure 5-5 Solution time Vs Problem size (without precedence) .................. 79
1 Introduction

With the advent of information technology, the influence of computers on our everyday life has ever been on the increase. This is true with manufacturing industries too, which have seen a tremendous increase in the different avenues in which computers could be used to make things in a more efficient manner. A very simple example of this is Computer Integrated Manufacturing (CIM). CIM could be defined as integration of total manufacturing enterprise through the use of integrated systems and data communications coupled with new managerial philosophies that improve organizational and personal efficiency. Thus, the emphasis clearly is on integration of information systems with data management that results in better productivity and higher profit for the management.

Process planning and scheduling are two very important steps in the manufacturing process of any product. Process planning has been defined by Chang and Wysk [14] as the act of preparing detailed operation instructions to transform an engineering design into a final part. Scheduling can be defined as the process of converting a general or outline plan for a project into a time-based representation, generating information on available resources and time constraints. These two important phases of a manufacturing process could be and are
increasingly computerized. This streamlines the production process enabling the industries to meet increasing market demand effectively and efficiently.

Scheduling in general is done during the final stages of the manufacturing planning. It is especially the case when industries have separate departments that perform process planning and scheduling for their manufacturing process. In this type of manufacturing the process plans are made well before the machines are scheduled for machining. This might and in many cases does lead to overloaded machines (bottlenecks) when the parts actually reach the manufacturing stage. Integrated process planning and scheduling avoids this problem to an extent, by pushing part of the process planning steps to the scheduling stage. By doing this, process planners know what machines are actually available when the part actually reaches the machining stage.

The main area of concern in the above aspect is that, in the traditional manufacturing organization, both the process planning function and the CAPP systems have very limited interactive communication and collaboration with the scheduling function. This results in process plans that are generated based on the condition of producing a given part in the best possible way, with a set of manufacturing resources assuming that all resources are available at all times. As studies have shown, this is not necessarily true always. There might be bottleneck machines, broken tools, unavailable tools etc, which has to be taken into account.
Thus, integration of process planning and scheduling becomes all the more important in practice.

Scheduling problems are typically constraint-based problems, modeled with a minimization or maximization of an objective function, satisfying a set of constraints. The objective function is decided by the management. Examples of minimization objective functions are minimizing make-span, average flow-time of parts, throughput time, etc. An example of a maximization objective function is maximizing the utilization of machines. Sometimes, these two could be combined in a model such as minimizing lead time while maximizing machine utilization. The constraints are those conditions that are used to avoid time conflicts on machines and parts. In plain words, a simple constraint avoids two parts to be machined on the same machine at the same time.

Alternative process plans or simply alternatives are the different methods of manufacturing the same part. The different methods could be at the machine level or the equipment level. At the machine level, there could be more than one machine available to machine a given feature and at the equipment level, to manufacture a feature on a given machine, there could be more than one cutting tool available on that machine that could be used to machine that feature. Process planning with alternatives greatly enhances the flexibility in manufacturing process of a part. The need for alternatives arises from the fact that a machine planned for machining a
given part might break down when the part reaches it for machining. At the equipment level too, a certain cutting tool may become unavailable when it is needed to machine a part. In these scenarios, alternative process plans give the process planners and schedulers greater flexibility and control over the manufacturing process.

Mathematical modeling is the technique of converting real-world problems into mathematical formulations of variables and equations whose analysis gives us solutions, insights and guidance for the original problem. Mathematical models consist of decision variables, objective functions and constraints. The main advantages of this technique are that, through mathematical modeling, the system being analyzed could be better controlled or the existing design could be vastly improved. Mathematical modeling also allows for the efficient use of computers and modern technology to solve complex problems. This technique could be used effectively in solving complex scheduling problems.

Depending upon the problem being solved, either linear or non-linear models could be built. Linear models have in their objective functions and constraints, terms of degree strictly less than or equal to one. Non-linear models can and in most cases have terms that are of degree typically higher than one, or models correspond to logical variables. The choice of the type of model depends very much
on the nature of the problem. For any given problem, the solution time may not be the same for the two types of models.

1.1 Motivation

One of the most important issues in scheduling is the time taken by the scheduling algorithm to reach the desired optimal schedule, if one exists. The time taken by the algorithm to find the optimal schedule is called the solution time. As extensive research has proven, many scheduling problems become NP complete as their size increases and do not have an efficient polynomial time algorithm to find the optimal schedule in practical time. When designing scheduling algorithms, equally important as getting to the optimal solution is the time taken by the algorithm to get there. As an example, for a $n$ job and $m$ machine job-shop scheduling problem, there are $(n_1)! (n_2)! \ldots (n_k)!$ theoretically possible sequences, where $n_k$ denotes the number of operations to be performed on machine $k$; however, not all of them may be feasible [9]. Clearly, this number becomes practically impossible to completely explore for a large (real world) problem and this directly influences the solution time.

Depending upon the method of exploring the possible sequences, different scheduling algorithms can have different solution times when applied to the same problem. In some cases, intelligent heuristics that prune the search space could be
applied in conjunction with the main scheduling algorithm to reduce the solution time significantly.

Also clear from the above example is that the solution time for any particular algorithm depends on the number of system parameters, which in this case are the number of parts, \( n \) and the number of machines, \( m \). Other examples of possible system parameters are the number of features on each part, the number of alternatives available for manufacturing each feature, and precedence information between features on a part. Some of these system parameters, if increased in number, increase the solution time while some improve the solution time. Hence, it becomes essential to study the effect of the system parameters on any given scheduling algorithm and find the relationship between the two. This helps in deciding which of the possible system parameters could be modified to get a better solution time.

The motivation of this thesis is to study the solution times of different job-shop scheduling algorithms for a given job-shop scheduling problem and also to study the effect of the system parameters on a given scheduling algorithm. This study provides the user with a benchmark to compare with and select the best among the available algorithms for his scheduling problem before he actually starts solving it. Also, the study of the effect of the system parameters on an algorithm’s solution
time helps the user in setting his system parameters, within reasonable limits, to lower the solution time for the algorithm chosen to solve his problem.

1.2 Objective

It is clear from the above discussion that the solution time of any algorithm for a scheduling problem depends to a considerable extent on the time spent (wasted) in exploring infeasible sequences, and for any given scheduling problem, this time varies with the algorithm chosen. The best scheduling algorithm optimizes (minimizes) the time spent in exploring the infeasible sequences in its search for the optimal sequence.

However, this does not necessarily mean that just minimizing the time spent on infeasible sequences will ensure optimal sequence in practically measurable time. In many large scheduling problems, the number of feasible sequences is itself so large that the time required to find the optimal sequence is prohibitive, especially on a single computer. Also, not all algorithms take the same time to solve a given problem. Some might be best suited for a class of problems while others may not be, depending upon the type of problem and the algorithm itself. Clearly, there is a need to know the best among different scheduling algorithms available for the problem at hand before applying one of them to find the optimal sequence. This
avoids the time wasted in testing the problem on all the different algorithms available and then selecting the best one for that problem class.

Equally important is to study the effect of the system parameters on the solution time for a given scheduling algorithm. As shown by the example above, the solution time in that example is very much dependant on the number of parts $n$ and the number of machines $m$. Though in this particular example, increasing $n$ and/or $m$ increases the solution time, it cannot be generalized that with more system parameters or the larger their size, longer the solution time. As an example, if the given problem has feature precedence information specified, then that reduces the search space, thus improving the solution time. Thus, different system parameters have different effect on the solution time for a given problem and scheduling algorithm, and studying that effect is very helpful in setting the system parameters for the problem at hand so as to improve the solution time.

As the above discussions show, the objectives of this thesis are first to compare the solution times of different job-shop scheduling algorithms for a given problem and second to study the effect of the system parameters on the solution time of a given scheduling algorithm. While the former helps us to understand which algorithm is best suited for a given class of problem instance, the latter helps us to optimally set the system parameters for the algorithm chosen to solve a given scheduling problem.
1.3 Overview of approach

This section briefly explains the overview of the approach undertaken for this thesis. As explained earlier, the objectives of this thesis are comparing solution times of different scheduling algorithms for job-shop scheduling problems and studying the effect of system parameters on any given algorithm chosen to solve a class of problems.

This thesis uses mathematical modeling techniques to represent real world scheduling problems as mathematical models. Linear and non-linear mathematical models representing the general class of job-shop scheduling problems have been built using a commercial modeling language. These models represent the system parameters as variables and employ constraints to avoid the infeasible sequences. While linear models employ the Simplex algorithm to solve the problem, non-linear models employ constraint programming algorithm to find the optimal sequence, if one exists.

In comparing different algorithms for the same problem, the same class of test problems have been tested on both types of models (algorithms), and the solution times of each recorded and studied. For the second objective, in studying the effect of the system parameters on the solution time for a given algorithm, the same algorithm was used many times to solve a particular class of problems, each time
changing the system parameters and recording the change in solution time, if any. Plots have been drawn for this case too, which show the variation of solution times for the selected algorithm, with the system parameters.

The mathematical models have been built using ILOG Optimization Language (OPL) and were developed using ILOG OPL Studio, the Integrated Development Environment (IDE) for building and editing mathematical models. A comprehensive and easy to use prototype system has been built for representing scheduling problems and testing scheduling algorithms. This prototype was built using the Java 2.0 programming language.
2 Literature Review

This section provides the literature review relevant for this research. The main topics that were explored are mathematical modeling of scheduling functions, integrated process planning and scheduling, and scheduling with alternatives.

2.1 Mathematical modeling of scheduling functions

An algorithm and mathematical model formulation for short term scheduling of multistage, multi-product batch plants are presented in [3]. The authors have proposed a novel formulation of a continuous–time mixed integer linear programming model with binary decision variables. This formulation does not require much solution time, as it is needed for a similar programming model with tetra index binary variables.

The problem under study is a multi-product batch plant involving non-identical production units at each production stage. Each product has to pass through a certain number of stages before being converted into final product. Each stage has a fixed number of units, which work upon the products. There are no two units that co-exist in two different stages. The objective of their research is to generate a short-term production schedule that minimizes the earliness of completion of the orders. The objective function of the model is to maximize the completion time of
orders in each stage and to impose a heavy penalty on tardy orders. This effectively forces the orders to be processed near the due dates.

The authors have also considered an ordering heuristic to force sort orders to be processed in a sequence of increasing completion dates. This heuristic helps reduce the number of decision variables by eliminating some unlikely order sequences. The heuristic is applied to only those orders that have a common unit of production. Orders are presorted based on the earliest intermediate completion time, where intermediate completion time is defined as the completion time at a stage intermediary between the current stage and a later one.

Constraints have been formulated that make sure that if an order, among two orders, has the earlier intermediate earliest completion time, then it gets to be processed first. If two orders have the same intermediate earliest completion times, then they are sorted based on the earliest starting time. If this also matches for the two orders then they are sorted on the basis of the processing time, with the order with a longer processing time getting preference over the other. The experiments performed by the authors have proven that applying heuristics largely reduces the solution time without affecting the objective value. As an extension of the authors’ work, research could be done on the effect of the system parameters like the number of units or the stages, on the solution time. Also, if the orders are pre-
sequenced then the solution time should be expected to improve compared to without order pre-sequencing.

A single operation, non-pre-emptive, deterministic scheduling problem with a set of \( n \) jobs to be processed on \( k \) identical machines is presented in [6]. Jobs assigned to each machine have a common due date. The number of machines is unknown. Activating a machine will require additional costs to be incurred. The objective is to find an optimal sequence, the optimal number of machines \( (k) \) and the respective due dates to minimize the weighted sum of earliness, tardiness and machine activation costs. The authors have proposed a polynomial time algorithm to solve the problem.

The trivial solution for the above mentioned problem would be to have \( n \) machines and process each job on an individual machine. But this is not feasible if the number of parts is huge. Hence the authors have formulated an activation cost for each machine that is selected for processing. The \( n \) jobs are grouped into \( k \) groups with each group being processed on one machine. Thus we will have \( k \) machines, which is also a decision variable to be found out by the formulated model.

The authors have considered two cases, one with the number of machines equal to 1 and the other with the number of machines between 1 and \( n \) and proposed algorithms for solving them. The authors in the preceding research have considered
a single operation scheduling problem, whereas ours is a multi-operation problem in that each part has at least one feature. Also they have not considered alternative process plans for each part, which is taken up in this research. The number of machines is a decision variable in their research, but is known to us in our research.

### 2.2 Integrated process planning and scheduling

A solution approach for an integrated process planning and scheduling function and performance analysis for the approach has been given in [5]. The authors have given importance to the scheduling phase in that whenever the scheduling module assigns an operation to a machine, the process planning module is called to check the validity of the assignment. This is a departure from the traditional practice of doing scheduling after deciding on the process plan. The authors have proposed a solution approach for the integrated model of process planning and scheduling, which gives improved schedules, but at the same time increases the complexity of the problem. The authors define and use a time window, which is a finite time frame from the current scheduling time to a time in the near future for part-machine assignments. When assigning parts to machines, the authors assign them only inside the time window.

Once the time window is defined, the system identifies a subset $M^*$ of the available machines and $S$ of the available parts. The subset is formed such that at
least one feature in each part of the subset S could be assigned to at least one machine in the subset M* within the time window. Once the subset is formed, the algorithm calculates the effectiveness for each part-machine combination based on some established criteria. Machines are assigned to parts based on an effectiveness matrix, with parts as rows and machines as columns and the effectiveness values as its elements.

In the above research, time window size affects the quality of schedules. Whereas a large window size might look like the better choice, it is not so always. This is because, large window size means the time gap between machine availability times is large and since every machine that becomes available in a time window should have at least one job assigned to it, this forces us to assign urgent jobs on machines that become available closer to the time window. In some problems, using a small window size would enhance the schedule by improving the start time of a part. Thus it is clear that, the quality of schedule is affected by the system parameters, in this case the time window, and identifying the correct system parameters is very important towards improving the schedule properties.

An implementation of an automatic process planning and scheduling system based on the concept of non-linear process plans is presented in [4]. The authors have explained the ESPIRIT project COMPLAN and have also presented a new
collaborative approach that uses production constraints as a feedback from scheduling to the process planning.

The authors try to emphasize the fact that in traditional production systems, process planning and scheduling are two separate functions. The Process Planning and Control (PPC) department tries to integrate these two functions. Separating process planning and scheduling functions leads to unwanted pressure on the schedulers, who are given too stringent due dates to be met, by the PPC department.

The authors explain the Non-Linear Process Plans (NLPP) which are in effect alternative process plans. In this case, we have more than one operation sequence for each job and this gives the scheduling function more flexibility in setting the schedules. The NLPPs grow during the lifetime of the product. This means other interesting alternatives could be later added on. Also, information coming from the workshop concerning performed times enables validation and improvements of NLPPs. The NLPPs have been found to improve reactivity on disturbances in case of reactive scheduling. Also, they increase the schedule performance by increasing productivity and decreasing mean lead time and work in progress.

The NLPPs have some shortcomings, the most important one being the absence of feedback from the scheduling department to the process-planning department. Though NLPPs provide the flexibility in selecting a particular process plan from a
set of alternatives, what actually happens is that the process planning department generates an arbitrary set of alternatives, based on the experience of the process planner, which may not be of interest at all to the scheduling department. To solve this problem, the authors propose establishing a feedback control loop between the process planning and scheduling departments, thus ensuring two-way information flow rather than one way namely from the process planning to the scheduling department.

The collaborative process planning and scheduling function described in this paper consists mainly of three traditional departments (activities), namely process planning, scheduling and workshop. Each of these activities has an evaluation module, which evaluates the output based on the feedback received from other activities. The feedback between activities is made at all levels, thus making the evaluation procedure easier.

The working of this system is as follows: the process planning department prepares NLPPs and gives them to the scheduling department, which prepares schedules based on the alternatives given to it. The feedback from the scheduling to the process planning department are the production constraints, based on which the NLPPs are evaluated and improved. After getting the alternative routings from the production planning department, the scheduling department prepares schedules and passes them onto the workshop, whose evaluation module evaluates the
schedule based on some established criteria. The feedback from the workshop to the process planning and scheduling department is in the form of performed operation times and starts and finish times of operations respectively.

The system proposed by the authors achieves a truly closed loop feedback between the various departments of the manufacturing organization. This eliminates the time involved in working on the alternatives that are of no use. More importantly, this feedback system avoids the risk of overlooking important aspects of production planning and control.

A review of the research in the area of integrated process planning and scheduling is presented in [10]. The authors discuss the extent of applicability of various approaches and suggest directions for future research. The authors give an overview of the kind of integration being done between CAPP and CAD system in most of today’s manufacturing systems. They opine that in most cases, CAPP systems try to integrate with the CAD systems, but not with the scheduling functions. In other words, the CAPP systems that try to do upstream integration almost overlook the potential benefits of downstream integration. The authors try to emphasize the fact that downstream integration between the CAPP systems and the scheduling functions has more advantages in terms of satisfying the process planning and scheduling criteria.
The other aspect of this problem when viewed from the scheduling point of view is the inability of utilizing alternative process plans due to the complexity of re-planning and re-scheduling. Fixed process plans often lead to severely unbalanced resource loading create superfluous bottleneck machines and lead to overall over-utilization of resources and poor on-time delivery performance.

The authors have also discussed about the various Computer Aided Process Planning (CAPP) systems that are in practice. All CAPP systems can be categorized under one of the following types: variant, generative and automatic. The variant approach uses group technology to retrieve plans for similar components using a table look-up procedure and modifying the selected plan to suit the component at hand. The generative approach tries to select a plan for each component without referring to existing plans. Plans selected on this approach are most consistent and can be easily incorporated into the plans of new processes, equipment, methods and tooling. Automated systems on the other hand completely eliminate human activity from the planning process.

The authors also look upon the different scheduling systems in practice. This research particularly confines to the integration of process planning with job-shop scheduling problems. It is the authors' view that job-shop scheduling problem are not only NP hard but also very complex even to solve for a simple test case. These
problems don’t seem to have a fixed solution procedure that seems to dominate over other methods for all possible test cases.

The authors’ research and comparison of traditional offline process planning and scheduling systems with integrated process planning and scheduling systems throws light on the benefits of the latter. Though integrated process planning and scheduling might look like the panacea of all the scheduling problems that industries face today, the user should also be aware of some difficulties in integrated process planning and scheduling. Some of the issues that needs attention depends on the algorithm used to implement the integration framework.

As an example, if a time window approach is used to integrate process planning and scheduling as done in [5], the time window size has to be correctly determined, because this affects the quality of the schedule. Also, this method tends to be more localized, in that an assignment done within a time window does not evaluate its effect on the overall solution over the entire scheduling horizon. Similarly, Simulated Annealing gives good quality solutions, but almost always, the search space is hopelessly large for most practical problems. Thus, the decision of integrating process planning and scheduling is not one that should be made overnight, but one that has to be made after due considerations have been given to such issues as the algorithm to be used to implement the integration framework, effect of the system parameters on the algorithm, etc.
2.3 Scheduling with alternatives

An effective computationally efficient procedure for scheduling jobs with alternative process plans, in a large scale manufacturing system has been developed in [13]. To construct a production schedule that minimizes maximum lateness, the authors have used a simulation-based iterative algorithm. Job queuing times observed at each machine in the previous iteration are used to compute a refined estimate of the effective due date for each job at each machine and in the current iteration, jobs are dispatched in the order of increasing slack. To further reduce maximum lateness, the authors have used a second scheduling algorithm that uses a tabu search procedure that identifies process plans with alternative operations and routings for the jobs. This enhancement has yielded improved schedules that minimize manufacturing cost and satisfy due dates at the same time.

To evaluate the quality of a schedule that minimizes the maximum lateness, the authors have used a lower bound on maximum lateness calculated using a relaxed scheduling algorithm. The latest possible finish time for job $i$ on a particular machine $m$, is calculated by taking into consideration the due date of the job, processing times of the job on all subsequent machines to $m$ and the queuing time for job $i$ at all subsequent machines to $m$ except the immediate subsequent machine. Each iteration of the simulation uses the revised late finish time from the previous iteration to dispatch jobs at each machine. Iterations of the scheduling algorithm
terminate when the lower bound on maximum lateness is achieved or the iteration limit is reached.

The second scheduling algorithm designed by the authors employs a tabu search procedure to find the best process plan from the search space that minimizes cost and the maximum lateness relative to the current solution. The performance measure used for this is a ratio of the difference between the maximum lateness of the current and the candidate solution (process plan) in the search space to the cost of the candidate solution (process plan). This metric is meant to determine which member of the neighborhood set of the current solution has the potential to result in the largest reduction in the lower bound for maximum lateness at the least cost, relative to the current solution.

The authors have evaluated the performance of their algorithms and also the attractiveness of scheduling with alternatives, over scheduling without alternatives. The authors have also evaluated the solution quality versus the computational time. In this scenario, it would be interesting to compare the effectiveness of the iterative and the tabu search methods. Also helpful would be to study the effect of the system parameters like the feature precedence information, number of operations for each job, etc. on the solution time.

A hierarchical approach to solve a multi objective production-scheduling problem with alternative process plans has been presented in [1]. The authors' approach is
based on the decomposition of the original problem into two sub-problems, namely, machine loading and scheduling, with a solution being presented only for the machine-loading sub-problem. A machine-loading problem tries to find efficient solutions with respect to the load balancing and cost objectives. In other words, solving the machine-loading sub-problem essentially amounts to selecting a process plan for each job and to route jobs to machines.

The generalized approach of the authors considers process plan as a set of macro-operations, with each macro-operation consisting of a set of micro options. Selecting a process plan is thus selecting a micro option for each macro-operation. Knowing the processing time and cost involved for each macro-operation/micro-option pair on each machine, the authors have formulated the problem as a bi-criterion integer programming model with the objective function of minimizing the load on the machines. Two heuristics have been proposed to solve the model, one based on surrogate duality theory and the other based on a genetic descent algorithm.

The mathematical model built by the authors is similar to the one used for this research, the only major difference being that the authors have relaxed the sequence constraints dealing with features on the same part and also the non-overlapping constraints for parts to be produced on the same machine. This is understandable, as their objective was only to solve the machine-loading sub-
problem, and these two constraints deal with the scheduling aspects of the production-planning problem. The authors have used three types of constraints. The first constraint is for the total cost of production, the second sets an upper bound for the load on the machines and the third is the process plan selection constraint, which together with the binary constraint on the decision variable avoids batch splitting. Avoiding batch splitting ensures that each part is processed on one and only one of the available machines.

The authors have used a set of benchmark problem instances to test both the heuristics. The tests have shown that, while the genetic descent method works better for smaller problem instances, the dual heuristic approach works better for large problem instances. As an alternative, applying the dual heuristic algorithm followed by the genetic descent algorithm ensures suitably good results.

An investigation into an optimization methodology for scheduling jobs in a just in time environment is presented in [12]. The authors consider a non-preemptive case, where each job consists of a distinct number of operations to be processed in a specified order. Each operation has to be processed on one of a set of resources with possibly different efficiency and hence processing time.

The objective of their research is to minimize the sum of weighted quadratic tardiness of the jobs. The problem set up is a set of jobs, each job has a given number of operations to be processed in a specific order. The authors have
assumed a linear ordering of operations for simplicity. Each job has an earliest start date and a due date. If the due date is not met, then a penalty is applied. This penalty is made time dependant, so that a part that is late by two time units has more penalty than two parts that are one time unit late.

The objective function is the sum of the weighted quadratic tardiness of each job. The model built is a constrained discrete-time integer optimization model. The authors have considered three types of constraints. Capacity constraints make sure that the number of operations active in a particular machine group should be less than or equal to the total number of active machines in that group at that time. Precedence constraints make sure that no operation on any part is done earlier than the earliest start date for that job. Integrity constraint makes sure that the start times, processing times and the finish times are integers. The authors have followed a Lagrangian relaxation technique to relax the capacity constraints and reach the solution.

The authors in their research have considered the start time of a particular operation on a machine to be independent of that machine. This is because they have assumed a linear ordering of the jobs on the machines. In our research, at least for the non-linear models, start times are dependent on the machine selected for an operation. This even though increases the size of the model, it helps understand the final solution in a better way. Also, depending on whether the start
time is independent of the machine or not, changes the way the constraints are formulated.

Development of a prototype feature-based multiple-alternative process planning system in which the process plan would be generated directly from design and available factory facility information is described in [15]. An overall removable volume is generated by graphically comparing the 3D part and 3D work piece blank. The manufacturing features are decomposed into a series of general manufacturing features by using a mixed graph-based and rule-based algorithm. The multiple-alternative process plan generation is based on recognized manufacturing features and various production rules. After generating multiple process plans, each process plan is allocated the possible manufacturing scheduling time and the candidate process plans are retrieved based on the required due date.

The proposed prototype system has four main components: a relational manufacturing database, form feature recognition, process alternative generation and scheduling state evaluation. The inputs to the prototype are the 3D removable volumes model and the blank raw material model which are entered in the IGES data format. Once the form feature recognition is done, the process alternative generation is performed using the CLIPS expert system shell into which the recognized manufacturing units are fed. Finally, the generated manufacturing process information for the manufacturing units is then transferred into the
scheduling state evaluation subsystem for manufacturing sequence generation and schedule validation.

The relational manufacturing database is developed based on an entity-relationship (ER) model, which has as entities the manufacturing activities, facilities to conduct those activities, material and work orders that consume these resources. In addition to these entities many other attributes are also added to achieve the functional requirements of scheduling verification and process planning. In short, this database is designed to provide all the required information needed for the prototype system to generate process plans, verify scheduling states and calculate the estimated cost for future.

The form feature recognition module retrieves the geometric design data of the product design and recognizes the manufacturing features. The design is first converted by using the removable volumes, which is the volume left out after the part is manufactured from the blank. The removable volumes contain the feature information need to be processed by machine tools. Such geometric data are given in the form of a CAD model represented by vertices, edges, surfaces, cylindrical centers and so on.

The process alternative generation module generates the various possible alternatives by using a rule-based knowledge representation and the current facility configuration based on available machine tools. This is achieved by using two
information resources namely the current facility configuration that is provided by the manufacturing database and the information on the manufacturing units that represent the removable volumes. The form feature recognition module provides this information.

The process alternative generation module performs two main tasks in order to have validated scheduling for those process assignments that were generated in the previous module. The manufacturing sequence establishment constructs all possible alternative process plans with their operation sequences and the bi-direction schedule verifies the availability of machine tools for each alternative process plan and generates the corresponding starting and finishing times.

Earliest start time determination, minimizing average value of total penalty and storage expenses for a job-shop manufacturing cell of $n$ jobs (orders) and $m$ machines is described in [2]. Each operation has random time duration and each job has a due date. The authors have taken into account a one-time penalty cost to be paid to the customer if the job is delayed beyond the due date. Also, they have an additional penalty cost for each time unit of delay and a storage cost, if the job is completed before the due date. The research tries to determine the earliest start time for the problem described above in order to minimize the average value of total penalty and storage expenses.
The authors have developed a simulation model of manufacturing the job-shop and comprising of the decision-making for each competitive situation, where a competitive situation is defined as a situation where, several jobs are ready to be served on one and the same machine. An optimization is carried out by applying the coordinate descent search method to the simulation model. The variable to be optimized is the earliest start time.

The situation under consideration is to select a job to be machined, from a set of available jobs when all of them are ready to be processed on a machine that is free at that point of time. Under this circumstance, the authors have applied a pair-wise comparison method to compare between all the available jobs, taking two at a time and eliminating one from the pair. In deciding the job to be selected from a pair in question, the authors have applied a cost-objective method that selects a job based on minimal total penalty cost and storage cost. The cost-objective method uses a coordinate descent search method to optimize the earliest start time for a job based on the total cost involved for that job.

2.4 **OPL and ILOG OPL Studio**

OPL stands for Optimization Programming Language. This is a modeling language, which can be used for solving linear, integer and constraint programming problems [7]. It provides a programmatic representation of the algebraic notations
like the objective function, constraints etc. OPL addresses the issue of constraint programming by providing a two-level architecture integrating a constraint and a programming component. While the constraint component deals with the basic operations of the architecture, such as satisfiability and entailment of constraints, the programming component specifies how to combine the basic operations, often in non-deterministic ways.

In addition to the above mentioned-features OPL also provides novel features as activities and resources, which are very helpful in modeling scheduling functions. Also OPL facilitates logical comparison of constraints, which prove very useful in framing constraints based on some run-time conditions.

ILOG OPL Studio is an Integrated Development Environment provided for creating, editing and executing models in an interactive environment. Solving a problem in OPL involves three steps. The first step is to create the model file, which explains the structure of the problem like the variables, objective function and the constraints. Next is to create a data file that has the data instances needed to execute the model. This need not be a separate file, but OPL provides for separation of data from the model. The third step is to create a project file and attach the model and data files to the project file, which is ready to be executed. ILOG OPL Studio also provides functionalities for debugging and visualizing OPL statements.
This research chose to use ILOG OPL as the modeling language and ILOG OPL Studio as the IDE for two main reasons. First one is the ease of representing real world problems as mathematical models in OPL. This is especially true for scheduling problems as is the case for this research. Second and the more important reason is the Java Application Programming Interface (APIs) provided for OPL. It is this API that enables the communication between the Scheduling Interface and the Solver used to solve the problem. Nevertheless, due to licensing restrictions, this research uses this API only for solving linear models using the ILOG CPLEX 7.5.0 optimizer. When we are trying to solve non-linear models, we have to do it from the ILOG OPL Studio IDE.

2.5 IMPlanner

IMPlanner stands for Intelligent Manufacturing Planner. It is a software application under development at the Laboratory for Intelligent Manufacturing and Planning (LabIMP) of the Industrial and Manufacturing Systems Engineering (IMSE) department at Ohio University. It can be defined as distributed system that integrates process representation, feature modeling, process visualization and process network generation with specialized modules such as CAD system and process planner on the internet via advanced programming technologies [8].
IMPlanner consists of several Java classes that contain information about the number of parts in a manufacturing system, their geometry and design, features on each part, manufacturing processes, machine tools, process plan visualization, etc. In addition to these, the IMPlanner also has Java classes that enable the user to view his problem instance on custom built Graphical User Interfaces (GUI). Custom built means, IMPLanner provides the general framework for these GUIs and the user can extend/modify them to suit his needs.
3 System architecture

3.1 Overview of the system

Figure 3-1 shows the system architecture. The user provides information on the system parameters and this is stored as Java objects in the Scheduling System Model (SSM) component of the IMPlanner.
This information is also stored as a XML file for later retrieval. The solver interface forms a bridge between the Scheduling System Model and the solver. The system information has to be represented in a solver compatible data file before it is sent to the solver. This data file, coupled with the selected model file is sent to the solver from the solver interface. In addition to invoking the solver from the interface, the user can also invoke the solver in a stand-alone mode, from the OPL IDE.

3.2 System components

As shown in Figure 3-1, the system consists of five main components namely the scheduling system model, the XML file for storing system information, solver interface, OPL IDE with the solver and the mathematical models with data file. The following paragraphs explain all these system components in detail.

3.2.1 Scheduling system model

A manufacturing system model consists of manufacturing activities. A manufacturing activity is any activity that contributes to the manufacturing of a part. A manufacturing activity comprises of part activities, machine activities and tool direction activities. This research deals only with part and machine activity and not tool direction activity. This is because, tool direction does not have any effect on the scheduling properties of a manufacturing system model. Part activities are process plans for parts, which give the machines through which the part has to go through
before being completely manufactured. Machine activities represent the machining processes of a part on each machine in its process plan. Hence, it could be said that each part activity is a collection of one or more machine activities. A part can have more than one process plan (alternatives), in which case it has many part activities, each of which can possibly have one or more machine activities.

A scheduling system, in an industrial set-up, could be interpreted as a subset of a manufacturing system model. A scheduling system consists of parts, manufacturing features and machines (resources) to manufacture those features. It is used to represent a scheduling problem, wherein the aim is to optimize the use of available resources, subject to some manufacturing constraints. The hierarchy used in representing a scheduling problem is the scheduling system has one or more parts, with each part having one or more manufacturing features. Each feature could be machined on one or more of the available machines in the system.

### 3.2.2 System information in XML file

This research uses an XML file to store the scheduling system information. The XML data file stores the system information as attributes for element tags. XML file format was chosen for the data file representation because of its portability over the web and the availability of XML parsers that can parse and glean information from an XML file.
Information about a part in the scheduling system includes its name, material, batch size and the list of manufacturing features it contains. Some examples of manufacturing features are hole, pocket, and slot. Each feature contains the following information: its name, precedence with other features in the same part (if any) and a list of processes called the process list. The process list contains information on all the alternative machines that could be used to manufacture the feature, together with the process name and the processing time. The machines also have their names stored in the scheduling system. All the above information is entered by the user when creating the problem instance.

3.2.3 Solver interface

The solver interface acts as the bridge between the scheduling system and the solver. It gets the information stored in the scheduling system, passes it to the solver, and communicates the final result from the solver back to the scheduling system, where it is stored as activities. Using this interface the user can instantiate a new problem or open an existing problem, select the model file he wants to use for his problem, generate the data file and invoke the solver to test the problem.

3.2.4 OPL solver

The optimization components are the modeling language used to represent the problem as mathematical models, the Integrated Development Environment (IDE)
used for building and editing the models and the actual solver that solves the problem. The solver uses the Mixed Integer Programming (MIP) technique together with the branch and bound algorithm for solving linear models and a constraint-programming algorithm for solving non-linear models. Using the MIP technique for the linear model permits the variables in the model to take real or integer values, but the constraint-programming algorithm used for non-linear models, restricts the variables to take only integer values. In general, for the same problem, the solver takes longer to solve the non-linear models than their linear counterparts.

### 3.2.5 Building mathematical models

Linear and non-linear models are used for this research. While the linear model can only have terms of degree one, the non-linear model can, and typically does have terms of degree more than one. The model uses variables to represent system parameters like the parts, features, machines, etc and mathematical relationships to represent resource constraints.
4 Implementation

This section explains the implementation details of this research. It explains how the system has been implemented, the user interface developed, and what the user has to do in each step to get the results.

4.1 Mathematical modeling – a generic scheduling model

Mathematical modeling is the representation of the model in terms of variable(s), an objective function and constraint(s). It is the technique of translating real world problems into mathematical formulations of variables and equations, whose analysis gives us answers, insights and guidance for the original problem.

Before formulating the model itself, the algorithm that will be used to solve the model has to be decided. The choice of the algorithm depends on the problem and its solvability. The algorithm to be used is important in that it controls the type of variables that can be used in the model. In achieving the objective of this research, linear and non-linear models have been built, which are solved by linear-programming and constraint-programming algorithms respectively. The next two sections describe in detail the two types of models being used for this research.

A generic scheduling model comprises data instances, decision variables, objective functions, constraints and the search strategy being used, if any. The data
instances represent the system parameters and the decision variables are those to which the solver assigns values at run time, during the optimization process. The objective function describes the parameter to be optimized in terms of the decision variables, and the set of constraints represents the resource constraints.

A generic scheduling model when formulated as a mathematical model could be linear or non-linear. While the former allows only linear terms and no logical constraints, the latter allows both non-linear terms and logical constraints. Logical constraints mean using logical decision making constructs like “and”, “or” etc., in the model. Apart from this, the main difference between these two types of models is the type of algorithm used by the solver in solving the problem represented by them.

4.2 Polynomial mixed integer programming model

This type of mathematical model, as explained earlier, has a polynomial objective function and/or polynomial constraints. This means the objective function and the constraints can, and in most cases will, have terms involving variables of second degree or more. As this is a polynomial model, the solver uses constraint programming for solving. Next, we explain the OPL formulation of the polynomial model being used in this research.
4.2.1 Notation

This section explains the notations being used in the models [11]. The problem in question consists of \( n \) jobs, each job has a number of features and each feature can be processed in up to \( m \) methods (alternative process plans), where a method is a process plan that uses a set of particular setup, tool and machine to produce the feature.

\( i \) = index on part

\( j \) = index on feature

\( k \) = index on process plan by which feature could be produced

\( k \) indicates that the process plan is to use machine \( k \)

\( \chi_{ijk} = 1 \), if feature \( j \) of part \( i \) is to be produced using process plan \( k \)

0, otherwise

\( t_{ijk} \) = start time of feature \( j \) on part \( i \) using machine \( k \)

\( p_{ijk} \) = processing time of feature \( j \) of part \( i \) on machine \( k \)
$u_{i,j_1} = 1$, if feature $j_1$ of part $i$ precedes feature $j_2$

0, otherwise

### 4.2.2 Fixing variables and data instances

The OPL formulation for declaring the data instances and the variables being used for the polynomial model is shown in Figure 4-1.

```plaintext
enum Parts = ...;
enum Features = ...;
enum Machines = ...;
int bigM = 10000;

//This array gives the precedence matrix of features on the same part
int samePartPrec[Parts,Features,Features] =...;
// This array gives the processing times of features on machines
float+ dur[Parts,Features,Machines] = ...;

// The variables used in the model
var int sTime[Parts,Features,Machines] in 0..131;
var int x[Parts,Features,Machines] in 0..1 ;

//The following structure is used in specifying the precedence matrix for features
//to be produced on the same machine.
struct partData{
  Parts partName;
  Features featureName;
};

{partData} partDataSet = ...;
{ Features} partFeatures [Parts] = ...;
```

Figure 4-1 Fixing decision variables and data instances in OPL
Parts, Feature and Machines are the enumerated data types for the number of parts, features and the machines in the system. The constant bigM is used in the constraints. The array samePartPrec gives the feature precedence information for features on the same part. As an example, if samePartPrec[p1, f1, f4] = 1, then it means, in part p1, feature f1 precedes feature f4. The array dur gives the duration of machining features on machines. The enumerated types described above, bigM and these two tri-index arrays, constitute the data instances for this model.

The variables used in the model are as follows. The array sTime gives the starting time of features on machines. The array x, of binary values, is the decision variable used in this model. If x[p1,f1,m5] = 1, then feature f1 of part p1 is machined on machine m5. The structure partData encapsulates part and feature information and is used in specifying the non-overlapping constraints described later in this section. The set partDataSet is an OPL set of the above-mentioned structure. The array partFeatures, defined for each part in the system, is an array of sets, where each set is a set of features in the part for which the array element is defined. For example if partFeatures[p2] = {f2,f3,f6}, then part p2 has the features f2, f3 and f6.
4.2.3 Process planning and scheduling objective function

Figure 4-2 shows the OPL formulation for the objective function in this model. The objective function, as explained earlier, is to minimize the total time spent by all the parts in the shop floor. In effect, this minimizes the average time spent by each part. It is obvious from the above formulation that this objective function has the product of two decision variables, namely $x$ and $sTime$, which makes this a non-linear model.

\[
\text{minimize} \\
\sum_{p \in \text{Parts}} \sum_{f \in \text{partFeatures}[p]} \sum_{m \in \text{Machines}} (x[p,f,m] \cdot sTime[p,f,m] + x[p,f,m] \cdot \text{dur}[p,f,m])
\]

Figure 4-2 Process planning and scheduling objective function

4.2.4 Sequence constraints and non-overlapping constraints for parts without sequence constraints

Having formulated the objective function and the decision variables with the data instances, the next step is to formulate the constraints needed for solving the model. Figure 4-3 shows the OPL formulation of the sequence constraints and the non-overlapping constraints of type I.
subject to
{
//1. Sequence Constraints
forall(p in Parts)
forall(f in partFeatures[p])
forall(df in partFeatures[p])
  if (f <> df) then
    if samePartPrec[p,f,df] = 1 then
      sequenceConst[p,f]: sum(m in Machines) (x[p,f,m]*sTime[p,f,m] + x[p,f,m]*dur[p,f,m])
      <=
      sum(m in Machines) (x[p,df,m]*sTime[p,df,m])
    else
      if samePartPrec[p,df,f] = 1 then
        sequenceConst[p,f]: sum(m in Machines) (x[p,df,m]*sTime[p,df,m] + x[p,df,m]*dur[p,df,m])
        <=
        sum(m in Machines) (x[p,f,m]*sTime[p,f,m])
      else
        //The non-overlapping constraint for parts without sequence constraints
        noSequenceConst[p,f]:
        (sum(m in Machines) (x[p,f,m]*sTime[p,f,m] + x[p,f,m]*dur[p,f,m])
        <=
        sum(m in Machines) (x[p,df,m]*sTime[p,df,m]))
        \n        (sum(m in Machines) (x[p,df,m]*sTime[p,df,m] + x[p,df,m]*dur[p,df,m])
        <=
        sum(m in Machines) (x[p,f,m]*sTime[p,f,m]))
    endif
  endif
else
endif;

Figure 4-3 Sequence constraints and non-overlapping constraints
These constraints are used for scheduling parts without sequence constraints defined between them. The basic formulation of the sequence constraint is, if feature $j_1$ precedes feature $j_2$ on part $i$, then the start time plus processing time of $j_1$ should be less than or equal to the starting time of feature $j_2$. Figure 4-4 shows this constraint and equation 4-1 shows the mathematical formulation of the same.

$$\sum_{k\in K} \left(x_{ijk} t_{ijk} + x_{ijk} \cdot p_{ijk}\right) \leq \sum_{k\in K} x_{ijk} t_{ijk}$$ (4-1)

The non-overlapping constraint for parts without sequence constraints makes sure that two features on a part with no sequence constraint defined between them, are not assigned to the same machine overlapping in time. In other words, let $J_i$ be the set of features on a particular part $i$ and let $j_1$ and $j_2$ be any two features in the set $J_i$, i.e., $j_1, j_2 \in J_i$ such that there is no sequence constraint defined between $j_1$ and $j_2$. In this case it is essential not to schedule $j_1$ and $j_2$ overlapping in time. More information on the mathematical formulation of this constraint can be obtained from [11]. Figure 4-5 and equations 4-2 and 4-3 show this constraint.
Either:

Part i

\[ \sum_{k} (x_{ijk}t_{ijk} + x_{ijk}p_{ijk}) \leq \sum_{k} x_{ij2k}t_{ij2k} \] (4-2)

or

\[ \sum_{k} (x_{ij2k}t_{ij2k} + x_{ij2k}p_{ij2k}) \leq \sum_{k} x_{ijkl}t_{ijkl} \] (4-3)
4.2.5 Non-overlapping constraint for features to be machined on the same machine

These constraints, referred to as non-overlapping constraints of type II, make sure that two features being machined on the same machine are not scheduled overlapping in time. Figure 4-6 shows the OPL formulation of this constraint.

```plaintext
// 2 Non overlapping constraints for features to be machined on the same machine
for all (m in Machines)
  for all (s in partDataSet)
    for all (ds in partDataSet)
      if (s <> ds) then
        nonOverLappingConst[s,partName,s.featureName,m]:
          // ds precedes s
          (x[s.partName,s.featureName,m]*sTime[s.partName,s.featureName,m] -
           x[ds.partName,ds.featureName,m]*sTime[ds.partName,ds.featureName,m]
           >=
           x[ds.partName,ds.featureName,m]*dur[ds.partName,ds.featureName,m])
      /
        // s precedes ds

      (x[ds.partName,ds.featureName,m]*sTime[ds.partName,ds.featureName,m] -
       x[s.partName,s.featureName,m]*sTime[s.partName,s.featureName,m]
       >=
       x[s.partName,s.featureName,m]*dur[s.partName,s.featureName,m])
      else
        endif;
```

Figure 4-6 Non-overlapping constraints of type II – OPL formulation
Let \( j_k \) be the set of features that can be produced on machine \( k \). Let \( j_1 \) of part \( i_1 \) and \( j_2 \) of part \( i_2 \) be two such features i.e. \( j_1, j_2 \in j_k \). Then either \( j_1 \) may precede \( j_2 \) or vice-versa. This is an example of either-or constraint, modeled using a binary variable. Figure 4-7 and equations 4-4 and 4-5 show these constraints.

**Equation 4-4**

\[
(x_{i1j1k} t_{i1j1k} - x_{i2j2k} t_{i2j2k}) \leq x_{i2j2k} P_{i2j2k}
\]  

**Figure 4-7 Non-overlapping constraints of type II**
4.2.6 Process plan selection constraint

This constraint makes sure that, of all the alternative process plans available for a feature, one and only one is always selected for scheduling. This constraint is used only for process planning with alternative process plans available for each feature. Equation 4-6 shows this constraint.

\[
\sum_k x_{ijk} = 1
\]  

(4-6)

4.3 Linear mixed integer programming model (lmipm)

As explained in section 4.2, the non-linearity in the model arises from the cross product terms in the objective function and constraints and the logical OR operator used in implementing the either/or constraints. The techniques used in linearizing the two types of polynomial terms are explained in sections 4.3.1 and 4.3.2, respectively.

4.3.1 Linearization of polynomial model by avoiding cross product terms

The cross product term in the polynomial formulation is the product of \( x_{ijk} \) and \( t_{ijk} \). To linearize this term, the start time variable \( t_{ijk} \) has been made independent of
the process plan decision variable, $x_{ijk}$. This was achieved by modifying $t_{ijk}$ from being a function of parts, features and machines, to be $t_{ij}$, a function of only parts and features [11]. Figure 4-8 shows the linearized formulation of the objective function in OPL, after the above-mentioned modification to $t_{ijk}$.

\[
\begin{align*}
\text{minimize} & \quad \sum_{p \in \text{Parts}} \sum_{f \in \text{partFeatures}[p]} (sTime[p,f] + (\sum_{m \in \text{Machines}} x[p,f,m] \cdot \text{dur}[p,f,m]) ) \\
\end{align*}
\]

Figure 4-8 Linearized formulation of objective function

Comparing Figure 4-2 and Figure 4-8, it is clear that the term $x[p,f,m] \cdot sTime[p,f,m]$ in the polynomial model has been replaced by its linear counterpart $sTime[p,f]$.

**4.3.2 Linearizing logical decisions in constraints**

The use of OPL’s logical OR (\lor) operator in formulating the non-overlapping constraints (types I & II) as shown in Figure 4-3 and Figure 4-6, introduces non-linearity into the model. As explained in 4.2.4, the non-overlapping constraints of type I are used to schedule parts without sequence constraints defined between their features. To linearize this constraint, this research has used the binary variable
$u_{i_j,i_2}$ defined in section 4.2.1. The linear formulation of this constraint in OPL is shown in Figure 4-9.

![Figure 4-9 Linearized non-overlapping constraints of type I](image)

As shown in Figure 4-9, depending upon the value of the variable $u_{i_j,i_2}$, formulated in OPL as $u[p,f,df]$, one of the two constraints is made redundant by the large number bigM, thus achieving the either/or condition.

Similar to the type I constraint, the logical decision making involved in the non-overlapping constraint type II, has been linearized by using another binary variable $y_{i_p,m}^{j_1,j_2}$, defined as follows [11].

$$y_{i_p,m}^{j_1,j_2} = 1, \text{ if feature } j_1 \text{ of part } i_1 \text{ precedes feature } j_2 \text{ of part } i_2 \text{ on machine } m$$

0, otherwise
Figure 4-10 shows the linearized formulation of this constraint.

```
// 2. Non-overlapping constraints for features of different parts to be machined on the
// same machine.
forall(m in Machines)
  forall(s in partDataSet)
    forall(ds in partDataSet)
      if s <> ds then // checks if parts are different
        // part ds precedes s
        bigM * (2 - x[s.partName, s.featureName, m] - x[ds.partName, ds.featureName, m]) +
        bigM * (1 - y[s.partName, s.featureName, ds.partName, ds.featureName, m]) +
        sTime[ds.partName, ds.featureName] -
        sTime[s.partName, s.featureName] >= dur[ds.partName, ds.featureName, m]
      &
      // part s precedes ds
      bigM * (2 - x[ds.partName, ds.featureName, m] - x[s.partName, s.featureName, m]) +
      bigM * y[s.partName, s.featureName, ds.partName, ds.featureName, m] +
      sTime[ds.partName, ds.featureName] -
      sTime[s.partName, s.featureName] >= dur[s.partName, s.featureName, m]
      else
      endif;
```

Figure 4-10 Linearized formulation of non-overlapping constraints of type II

As shown in Figure 4-10, depending on the value of the binary variable $y_{ij_k}^{i_1, i_2}$, formulated in OPL as $y[part, feature, part, feature, machine]$, one of the two constraints is nullified by the large number $bigM$, thus implementing the either/or
condition. The process plan selection constraint, as defined in equation 4-6, does not involve any polynomial terms and hence was not linearized.

### 4.4 User interface

This section details the solver interface built for this research. The interface has been built using the Java programming language. Through this interface, the user can enter the system parameters, store the problem as a scheduling system model, and can solve it by invoking the solver.

#### 4.4.1 Interface components

The solver interface is shown in Figure 4-11. It can be broadly divided into two halves, the upper half, used for representing the problem as a scheduling model and the lower half, used for the optimization process by invoking the solver. The list of part displays the parts in the system. This list is updated dynamically as the user adds/deletes parts.
On clicking a part from the list of parts, the part model panel displays information about that part, which includes the processing times of features on machines and feature precedence information, if any. The scheduling system tool bar and the optimization tool bar contain the various components which are explained in Table 4-1 and Table 4-2. The model selection drop box helps select a model and the solution panel displays the final solution as returned by the solver.
Table 4-1 Solver interface components - I

<table>
<thead>
<tr>
<th>SNo</th>
<th>Name</th>
<th>Type</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Button</td>
<td>Button</td>
<td>Opens system file</td>
</tr>
<tr>
<td>2</td>
<td>Button</td>
<td>Button</td>
<td>Saves system information in XML file</td>
</tr>
<tr>
<td>3</td>
<td>Add Part</td>
<td>Button</td>
<td>Adds part to the system</td>
</tr>
<tr>
<td>4</td>
<td>Delete Part</td>
<td>Button</td>
<td>Deletes selected part</td>
</tr>
<tr>
<td>5</td>
<td>Add Machine</td>
<td>Button</td>
<td>Adds machine to the system</td>
</tr>
<tr>
<td>6</td>
<td>Delete Machine</td>
<td>Button</td>
<td>Deletes selected machine</td>
</tr>
<tr>
<td>7</td>
<td>Add Feature</td>
<td>Button</td>
<td>Adds feature to the part</td>
</tr>
<tr>
<td>8</td>
<td>Delete Feature</td>
<td>Button</td>
<td>Deletes selected feature</td>
</tr>
<tr>
<td>9</td>
<td>Save Part</td>
<td>Button</td>
<td>Saves part information</td>
</tr>
<tr>
<td>10</td>
<td>Feature-Machine Table</td>
<td>Tab</td>
<td>Displays feature processing times</td>
</tr>
<tr>
<td>11</td>
<td>Feature-Precedence Table</td>
<td>Tab</td>
<td>Displays feature precedence</td>
</tr>
<tr>
<td>SNo</td>
<td>Name</td>
<td>Type</td>
<td>Purpose</td>
</tr>
<tr>
<td>-----</td>
<td>-----------------------</td>
<td>--------</td>
<td>----------------------------------------------</td>
</tr>
<tr>
<td>1</td>
<td>Generate Data</td>
<td>Button</td>
<td>Generates OPL compatible data file</td>
</tr>
<tr>
<td>2</td>
<td>Open Data File</td>
<td>Button</td>
<td>Opens an existing data file</td>
</tr>
<tr>
<td>3</td>
<td>Execute Model</td>
<td>Button</td>
<td>Invokes the solver to solve the problem</td>
</tr>
<tr>
<td>4</td>
<td>Save Model</td>
<td>Button</td>
<td>Saves the model file</td>
</tr>
<tr>
<td>5</td>
<td>Save Data</td>
<td>Button</td>
<td>Saves the data file</td>
</tr>
<tr>
<td>6</td>
<td>Save Result</td>
<td>Button</td>
<td>Saves the result as a text file</td>
</tr>
<tr>
<td>7</td>
<td>Update Solution</td>
<td>Button</td>
<td>Updates scheduling system with solution</td>
</tr>
<tr>
<td>8</td>
<td>Model File</td>
<td>TextField</td>
<td>Displays physical path to model file</td>
</tr>
<tr>
<td>9</td>
<td>Browse</td>
<td>Button</td>
<td>Browse for the model file</td>
</tr>
<tr>
<td>10</td>
<td>Model</td>
<td>Tab</td>
<td>Displays the model file</td>
</tr>
<tr>
<td>11</td>
<td>Data</td>
<td>Tab</td>
<td>Displays generated data file</td>
</tr>
<tr>
<td>12</td>
<td>Result</td>
<td>Tab</td>
<td>Displays the solution</td>
</tr>
</tbody>
</table>

### 4.5 Steps to run the system and get the final solution

This section explains the steps to be followed by the user in initializing the system, invoking the solver, and viewing the final solution. For simplicity, this
section will consider the example of reading an XML file containing the system information instead of the user adding all the parameters manually using the GUI components just described.

4.5.1 Reading the XML file containing system information

The first step after initializing the Java application (this gives the solver interface) is to read the XML file containing the system information. The file being considered for this example is “2p-2f-2m-nopre-nosetup-2alternatives.xml”. This file is for a problem with the system containing 2 parts, 2 features, 2 machines with no precedence between the features and no set-up times included in the model. Also, every feature on all the parts has both the two machines as possible alternatives. The objective is to assign one and only one machine to each feature, minimizing the total completion time of all the features, avoiding conflicts. Figure 4-12 shows the solver interface, after the xml file was opened.
Figure 4-12 System information read from an XML file

The figure shows the processing times of the features on part \( p_1 \) on the machines \( m_1 \) and \( m_2 \). The part whose information is being displayed is highlighted in the list of parts. Even after reading in the XML file, the user can make changes to the processing times in the table.

4.5.2 Selecting and displaying model file

In this step, the user selects the model file of his choice to be used in solving the problem at hand. This could be a linear or a non-linear model. This is shown in
Figure 4-13, where the user has selected a linear model file (lmipmodel.mod), using the drop down menu provided.

Once the model file has been selected, its contents are displayed in the Model tab in the lower half of the interface. This is shown in Figure 4-14. This window is editable, thus providing a miniature OPL model editor for the user.
Figure 4-14 Displaying model file
4.5.3 Generating data file

This step generates the data file to be used by the solver during the execution phase. This data file has to conform to the syntax and data structure followed by the OPL language. The user can generate this data file by clicking the “Generate Data” button. The generated data file is displayed as shown in Figure 4-15.

Figure 4-15 Generating and displaying data file
4.5.4 Executing the model and displaying results

This is the final step, in which the user invokes the solver from the solver interface and solves the problem. The solver is invoked by clicking the **Execute Model** button and once the problem is solved, the final results, along with the solution time are displayed in the **Result** tab as shown in Figure 4-16.

![Figure 4-16 Displaying results](image-url)
5 Testing

This section details the testing carried out on the proposed system to study the effect of model linearity on the solution time. The effect of system parameters, namely number of alternatives and feature precedence, on the solution time of the linear model has also been studied and reported.

5.1 An example

This section explains an example problem solved using the system described in section 4. The example considered is a 3x3x3 problem with 3 alternatives for each feature and no precedence among the features.

Figure 5-1 shows the processing times of the three features on the three parts, for this example. The granularity of the time unit for the processing times doesn’t matter, because this research is concerned only about the solution time. Figure 5-2 shows the starting times of the three features on the machines that were selected by the solver in the optimal solution. Figure 5-3 shows the Gantt chart for this example and it is clear from the chart that the objective function value, which is the sum of the starting and processing times of all the features, for this example is 270. The total time taken for solving this problem was 1.06 seconds, as shown in Table 5-1.
<table>
<thead>
<tr>
<th>Sno</th>
<th>Parts</th>
<th>F1</th>
<th>F2</th>
<th>F3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>M1</td>
<td>M2</td>
<td>M3</td>
</tr>
<tr>
<td>1</td>
<td>P1</td>
<td>10</td>
<td>20</td>
<td>17</td>
</tr>
<tr>
<td>2</td>
<td>P2</td>
<td>9</td>
<td>18</td>
<td>11</td>
</tr>
<tr>
<td>3</td>
<td>P3</td>
<td>30</td>
<td>32</td>
<td>25</td>
</tr>
</tbody>
</table>

Figure 5-1 Processing times of features

<table>
<thead>
<tr>
<th>Sno</th>
<th>Parts</th>
<th>F1</th>
<th>F2</th>
<th>F3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>M1</td>
<td>M2</td>
<td>M3</td>
</tr>
<tr>
<td>1</td>
<td>P1</td>
<td>11</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>P2</td>
<td>-</td>
<td>-</td>
<td>11</td>
</tr>
<tr>
<td>3</td>
<td>P3</td>
<td>-</td>
<td>34</td>
<td>-</td>
</tr>
</tbody>
</table>

Figure 5-2 Starting times of features
5.2 Testing linear model

Tests were carried out to study the effect of two system parameters, namely number of alternatives and feature precedence, on the solution time of a linear model. Problem sizes were varied from 2 parts-2 features-2 machines (2x2x2) to 5 parts-5 features-5 machines (5x5x5). In each case, the number of alternatives was increased from the minimum (one) to the maximum (number of machines available). The testing was done for problems with and without precedence. Two
graphs have been plotted, both between the solution times and problem size, for all possible alternatives, one with precedence and the other without precedence information.

### 5.3 Testing results for linear model

Table 5-1 and Table 5-2 show the solution time and objective function value results, respectively, for the linear model. NOA stands for number of alternatives available for each feature.

Table 5-1 Solution times of linear model

<table>
<thead>
<tr>
<th>SNo</th>
<th>Type</th>
<th>NOA</th>
<th>Number of Variables</th>
<th>Constraints</th>
<th>Solution time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>With precedence</td>
</tr>
<tr>
<td>1</td>
<td>2x2x2</td>
<td>1</td>
<td>52</td>
<td>32</td>
<td>0.19</td>
</tr>
<tr>
<td>2</td>
<td>2x2x2</td>
<td>2</td>
<td>52</td>
<td>32</td>
<td>0.19</td>
</tr>
<tr>
<td>3</td>
<td>3x3x3</td>
<td>1</td>
<td>306</td>
<td>243</td>
<td>0.49</td>
</tr>
<tr>
<td>4</td>
<td>3x3x3</td>
<td>2</td>
<td>306</td>
<td>243</td>
<td>0.50</td>
</tr>
<tr>
<td>5</td>
<td>3x3x3</td>
<td>3</td>
<td>306</td>
<td>243</td>
<td>0.61</td>
</tr>
<tr>
<td>6</td>
<td>4x4x4</td>
<td>1</td>
<td>1168</td>
<td>1024</td>
<td>1.58</td>
</tr>
<tr>
<td></td>
<td>Shape</td>
<td>#</td>
<td>Size</td>
<td>Res</td>
<td>Num</td>
</tr>
<tr>
<td>---</td>
<td>-------</td>
<td>---</td>
<td>------</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>7</td>
<td>4x4x4</td>
<td>2</td>
<td>1168</td>
<td>1024</td>
<td>2.19</td>
</tr>
<tr>
<td>8</td>
<td>4x4x4</td>
<td>3</td>
<td>1168</td>
<td>1024</td>
<td>2.53</td>
</tr>
<tr>
<td>9</td>
<td>4x4x4</td>
<td>4</td>
<td>1168</td>
<td>1024</td>
<td>6</td>
</tr>
<tr>
<td>10</td>
<td>5x5x5</td>
<td>1</td>
<td>3400</td>
<td>3125</td>
<td>3.64</td>
</tr>
<tr>
<td>11</td>
<td>5x5x5</td>
<td>2</td>
<td>3400</td>
<td>3125</td>
<td>6.25</td>
</tr>
<tr>
<td>12</td>
<td>5x5x5</td>
<td>3</td>
<td>3400</td>
<td>3125</td>
<td>5540.64</td>
</tr>
<tr>
<td>13</td>
<td>5x5x5</td>
<td>4</td>
<td>3400</td>
<td>3125</td>
<td>Out of memory</td>
</tr>
<tr>
<td>14</td>
<td>5x5x5</td>
<td>5</td>
<td>3400</td>
<td>3125</td>
<td>Out of memory</td>
</tr>
</tbody>
</table>
Table 5-2 Objective function for linear model

<table>
<thead>
<tr>
<th>SNo</th>
<th>Type</th>
<th>NOA</th>
<th>Number of Variables</th>
<th>Constraints</th>
<th>Objective value (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>With precedence</td>
<td>No precedence</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>2x2x2</td>
<td>1</td>
<td>52</td>
<td>32</td>
<td>67</td>
</tr>
<tr>
<td>2</td>
<td>2x2x2</td>
<td>2</td>
<td>52</td>
<td>32</td>
<td>57</td>
</tr>
<tr>
<td>3</td>
<td>3x3x3</td>
<td>1</td>
<td>306</td>
<td>243</td>
<td>562</td>
</tr>
<tr>
<td>4</td>
<td>3x3x3</td>
<td>2</td>
<td>306</td>
<td>243</td>
<td>403</td>
</tr>
<tr>
<td>5</td>
<td>3x3x3</td>
<td>3</td>
<td>306</td>
<td>243</td>
<td>332</td>
</tr>
<tr>
<td>6</td>
<td>4x4x4</td>
<td>1</td>
<td>1168</td>
<td>1024</td>
<td>1441</td>
</tr>
<tr>
<td>7</td>
<td>4x4x4</td>
<td>2</td>
<td>1168</td>
<td>1024</td>
<td>1311</td>
</tr>
<tr>
<td>8</td>
<td>4x4x4</td>
<td>3</td>
<td>1168</td>
<td>1024</td>
<td>1095</td>
</tr>
<tr>
<td>9</td>
<td>4x4x4</td>
<td>4</td>
<td>1168</td>
<td>1024</td>
<td>1024</td>
</tr>
<tr>
<td>10</td>
<td>5x5x5</td>
<td>1</td>
<td>3400</td>
<td>3125</td>
<td>2944</td>
</tr>
<tr>
<td>11</td>
<td>5x5x5</td>
<td>2</td>
<td>3400</td>
<td>3125</td>
<td>2171</td>
</tr>
<tr>
<td>12</td>
<td>5x5x5</td>
<td>3</td>
<td>3400</td>
<td>3125</td>
<td>1863.8</td>
</tr>
<tr>
<td>13</td>
<td>5x5x5</td>
<td>4</td>
<td>3400</td>
<td>3125</td>
<td>Out of memory</td>
</tr>
<tr>
<td>14</td>
<td>5x5x5</td>
<td>5</td>
<td>3400</td>
<td>3125</td>
<td>Out of memory</td>
</tr>
</tbody>
</table>
As shown in Table 5-1 above, and as expected, the solution time increases as the problem size and number of alternatives are increased. Of special interest is the case of 5x5x5 problem, with 5 parts, 5 features on each part and 5 machines in the system. The computer ran out of memory solving this problem without precedence. That shows that the number of solution states to be explored was huge. Even with precedence, the computer ran out of memory for the case of four and five alternatives. It is interesting to note the large difference between the solution time for the two and three alternatives cases (6.25 and 5540.64 seconds respectively). This shows that, for problems of this size and greater, with many alternatives, it would be better for the user to choose an algorithm that employs some search strategy to prune the solution space more efficiently.

Also notable is, for a given number of alternatives, the difference between the solution time, with and without precedence. This is more pronounced in the case of the 4x4x4 and 5x5x5 problems. As explained above, the computer runs out of memory solving the 5x5x5 problem, without precedence. These results show that, for large problems (4x4x4 and above), it is better for the user to explore the feasibility of specifying precedence information before solving the problem.

Table 5-2 shows the objective function values for the testing done with the linear model. The objective function considered by this research is the sum of the starting and processing times of all the features on all the parts in the system. This table
shows that introducing precedence information worsens the optimal solution value. This is because, more the precedence, smaller the solution space for the problem, hence the quality of the solution goes down. This effect of the precedence on the optimal solution value is more emphasized in the case of problems of size 4 and more. Comparing Table 5-1 and Table 5-2, it is clear that, while introducing precedence constraints improves the time taken to solve the problem using linear model, it does bring down the quality of the optimal solution.

### 5.4 Graphs for linear model

Figure 5-4 and Figure 5-5 show the graphs plotted for the linear model, from the test results of Table 5-1. Problem size 5x5x5 was not plotted for the obvious reason of non-availability of plottable results, as is evident from Table 5-1. It is clear from the graphs that the number of alternatives has a dramatic effect on the solution time of a problem. This is all the more pronounced in the case of a problem without precedence. Even in this case, for a given problem size, the effect of the number of alternatives is greater as the number of alternatives increases. As shown in Figure 5-5, for the problem size 4 without precedence, the solution time increases exponentially with the number of alternatives, from being 6.92 seconds for 1 alternative to 244.69 seconds for 4 alternatives.
Figure 5-4 Solution time Vs Problem size (with precedence)

Figure 5-5 Solution time Vs Problem size (without precedence)
Precedence comes into play on the maximum solution time needed for a given problem size. The maximum solution time, for the 4x4x4 problem with precedence is 6 seconds, while the same problem takes 244.69 seconds to get to the optimal solution, without precedence information. This shows the effect of specifying precedence information on the solution time.

5.5 Testing results for non-linear model

The non-linear model was also tested with the same set of data files as the linear-model. In the case of non-linear model, the solver used constraint programming algorithm to find the optimal solution. This, unlike the Simplex algorithm used for solving the linear model, does an exhaustive search on the solution space to find the optimal solution. Hence, understandably, the solution time grows exponentially as the problem size increases. As an example, for a problem with just 20 binary variables, the solver has to explore $2^{20}$ possible solution points before getting to the optimal solution, if one exists. This is the main reason why many of the test cases did not yield optimal solution, within the testing time-frame, with the non-linear model. The test results for the non-linear model are shown in Table 5-3.
Table 5-3 Solution times of non-linear model

<table>
<thead>
<tr>
<th>SNo</th>
<th>Type</th>
<th>NOA</th>
<th>Number of Variables</th>
<th>Constraints</th>
<th>Solution time (seconds)</th>
<th>With precedence</th>
<th>No precedence</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2x2x2</td>
<td>1</td>
<td>16</td>
<td>32</td>
<td>0.09</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2x2x2</td>
<td>2</td>
<td>16</td>
<td>32</td>
<td>0.01</td>
<td>0.09</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3x3x3</td>
<td>1</td>
<td>54</td>
<td>243</td>
<td>0.09</td>
<td>0.17</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>3x3x3</td>
<td>2</td>
<td>54</td>
<td>243</td>
<td>0.03</td>
<td>3735.09</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>3x3x3</td>
<td>3</td>
<td>54</td>
<td>243</td>
<td>0.53</td>
<td>No solution</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>4x4x4</td>
<td>1</td>
<td>128</td>
<td>1024</td>
<td>685.66</td>
<td>No Solution</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>4x4x4</td>
<td>2</td>
<td>128</td>
<td>1024</td>
<td>15368.89</td>
<td>No solution</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>4x4x4</td>
<td>3</td>
<td>128</td>
<td>1024</td>
<td>17045.88</td>
<td>No solution</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>4x4x4</td>
<td>4</td>
<td>128</td>
<td>1024</td>
<td>No solution</td>
<td>No solution</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>5x5x5</td>
<td>1</td>
<td>250</td>
<td>3125</td>
<td>11031.84</td>
<td>No Solution</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>5x5x5</td>
<td>2</td>
<td>250</td>
<td>3125</td>
<td>11126.70</td>
<td>No Solution</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>5x5x5</td>
<td>3</td>
<td>250</td>
<td>3125</td>
<td>15536.73</td>
<td>No Solution</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>5x5x5</td>
<td>4</td>
<td>250</td>
<td>3125</td>
<td>No solution</td>
<td>No Solution</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>5x5x5</td>
<td>5</td>
<td>250</td>
<td>3125</td>
<td>No solution</td>
<td>No solution</td>
<td></td>
</tr>
</tbody>
</table>
All the test runs were conducted for a maximum of 5 hours. In the table above, entries underlined, indicate the time taken to reach the best feasible solution before the test run was aborted. As is evident from the table above, precedence information plays a vital role in determining the rate at which the solver finds the solution, for a non-linear model. For problem sizes ranging from 3x3x3 with 3 alternatives to 5x5x5 with 5 alternatives, without precedence, the solver did not get even a single solution in the 5 hour testing period. For the same problems, with precedence information, the solver was able to get to a feasible solution, in the 5 hour testing period and may have reached the optimal solution if the test run was continued. But, it is clear that, more than the precedence information, a better search strategy for the non-linear model could enable the solver reach the optimal solution, if one exists, within a measurable amount of time.
6 Conclusion and future work

This research has done a comparative study of two mathematical models and their associated algorithms used for solving integrated job-shop scheduling problems without set-up times. The models compared were linear and non-linear, using Simplex and constraint programming algorithms, respectively. This section gives the concluding remarks for this research and suggests some future work directions.

6.1 Effect of algorithm on solution time

As seen from the preceding sections, the choice of algorithm influences the solution time to a great extent. Table 5-1 and Table 5-3 show that, with no pruning strategy applied with the constraint programming algorithm, the linear model works better than its non-linear counterpart. This is especially true for large problems where the solution space to be explored is large. Thus, in the absence of any heuristic to perform efficient search on the solution space, it can be concluded that, for a given problem, the linear model works faster than the non-linear model.
6.2 Effect of system parameters on solution time

This research has also studied the effect of the system parameters namely, number of alternatives and feature precedence information, on the solution time, for both the algorithms. As shown in Table 5-1 and Table 5-3 for linear and non-linear models respectively, it is clear that, the number of alternatives profoundly influences the solution time. This influence is all the more pronounced as the problem size increases, as shown by the 4x4x4 and 5x5x5 examples of both the above tables. But this could be offset by providing feature precedence information for one or more parts in the system. The user, satisfying feasibility conditions, can try a mixture of both the above system parameters to improve the solution time.

As an example, as shown by Table 5-1, the solution time for a 4x4x4 problem, being solved with a linear model, for one alternative and no precedence is 6.92 seconds. The same problem, using the same linear model, with four alternatives and fixed precedence information provided, takes only 6 seconds to solve. Thus, it is clear that, the fastest way to get an optimal solving time, with either model type, is to intelligently combine the system parameters, of course, if that does not violate any other manufacturing constraints.
6.3 Future work direction

The following sections give some suggestions for future work directions, following this research.

6.3.1 Improved non-linear model

As is clear from Table 5-3, the non-linear model used for this research did not get to the optimal solution for most of the test cases. This can be attributed to the exponentially increasing solution space to be explored with increasing problem size and the constraint programming algorithm doing a blind search on all such possible solution points. An obvious continuation this research would be to improve the solution time of non-linear models by employing a search strategy that prunes the solution space to a manageable size, such that the solver need not search all the possible states for the optimal solution.

6.3.2 Other types of problems and system parameters

It would be interesting to use the system built by this research, with the needed modifications, to study the effect of solution algorithms on the solution time for other scheduling problem types, like open-shop scheduling. Also interesting would be to include more system parameters like set-up times, machine loading/unloading
times, batch size, etc., into consideration and study their interaction with the type of model being used for solving, in influencing the solution time.

### 6.3.3 Distributed solving of large problems

As is evident from Table 5-1 and Table 5-3, single computers are exhausted trying to solve problems of even moderate size, like 5x5x5. The problem worsens if there is no precedence information specified. Under these conditions, the most obvious choice would be to use more than one computer in solving large problems. This could be done using a distributed solving paradigm, where multiple computers, called the “clients” participating in a solution process, solve a sub-set of the original problem and report their local optimal solution to a central computer, called the “server”. This reporting happens in real time. The best solution among the local optimal solutions becomes the current global optimal solution. The server too, in real time, maintains and updates the global optimal solution, as and when clients report their optimal solution.
7 References


