

<http://researchcommons.waikato.ac.nz/>

Research Commons at the University of Waikato

Copyright Statement:

The digital copy of this thesis is protected by the Copyright Act 1994 (New Zealand).

The thesis may be consulted by you, provided you comply with the provisions of the Act and the following conditions of use:

- Any use you make of these documents or images must be for research or private study purposes only, and you may not make them available to any other person.
- Authors control the copyright of their thesis. You will recognise the author's right to be identified as the author of the thesis, and due acknowledgement will be made to the author where appropriate.
- You will obtain the author's permission before publishing any material from the thesis.

**THE APPLICATION OF
SEARCH HEURISTICS
TO THE
SINGLE MACHINE
EARLY/TARDY
SCHEDULING PROBLEM**

A thesis
submitted in fulfilment of
the requirements for the Degree
of
Doctor of Philosophy
at the
University of Waikato
by

ROSS JONATHAN WILLIAM JAMES

University of Waikato
New Zealand
1995

Abstract

Just-In-Time manufacturing aims to reduce waste within an organisation, by focusing on completing orders at the time requested by customers. Failure to complete orders at the time specified causes the organisation to incur additional costs. The Just-In-Time environment has introduced complexities for scheduling which have been modelled by the Single Machine, Early/Tardy Machine Scheduling problem. Research to date has concentrated on models with assumptions such as the use of a single due date for all jobs or restrictions on early/tardy penalties. This research uses a generalised model which makes no such assumptions. The use of three popular 'intelligent' search techniques: Simulated Annealing, Genetic Algorithms and Tabu Search are investigated as possible solution techniques.

The properties of the common due date, early/tardy scheduling problem enable schedules to be defined from early/tardy job specifications - which state whether individual jobs are scheduled earlier or later than requested. Search techniques can then be applied using an early/tardy job specification solution space rather than the sequence of jobs solution space. Experiments found that the former was more efficient.

The properties of the common due date problem are generalised for the distinct due date problem so that a distinct due date schedule can also be defined from early/tardy job specifications, however, this is only possible with the use of a heuristic. By incorporating an early/tardy heuristic into a search, the early/tardy job specification solution space can be used in a search technique. Two early/tardy heuristics are developed, one being a construction based heuristics, the other based on a principle of conflict resolution. Both of these heuristics, unlike any other heuristic for this problem, determine the sequence and timing of each job simultaneously.

Experiments demonstrated that the early/tardy heuristics could find good quality solutions, however, search techniques based on the standard sequence of jobs solution space found better solutions. On larger sized problems, searches based on the early/tardy heuristic made large initial gains but little subsequent progress. Hybrid techniques which used both the heuristic solution space and the sequence of jobs solution space were found to produce the best results. Of the search techniques, Tabu Search was found to consistently produce better results than Simulated Annealing and Genetic Algorithms. Experiments also found that a Tabu Search with an early/tardy heuristic-based diversification strategy was the least sensitive to the starting point used by the search.

A final experiment found that if an error in the earliness and tardiness penalties is within the range of -25% and 20%, there is unlikely to be a significant difference in the solution quality.

Acknowledgments

There are many people who, either voluntarily or not, became involved in the research presented in this thesis. My thanks goes to all who have taken an interest in my work, those who have encouraged me and supported me through the highs and the lows in the last few years.

Special thanks go to Dr John Buchanan who went well beyond his call of duty in being my supervisor. He acted as a sounding board for many of my ideas and helped me see the results objectively - often keeping me on an even keel when things appeared to be going wrong.

Thanks too must go to Professor Les Foulds for his supervision, help and valued input, especially in the write-up stages of the thesis.

Thanks too must go to the University of Waikato and the William Georgetti Trust who generously provided scholarships for this research which made the whole project possible.

A special thank you also goes to my parents who have supported me throughout, not only this project, but in all that I have done. Their love and prayers have seen me through it all. Thanks also goes to them for providing me with the computer equipment which I have used throughout this thesis - without it I would have been lost!

My love and thanks also go to my wife Dale, who has also supported me through the good and the bad. For putting up with talk about E/T scheduling day after day and maintaining an interest throughout.

Last, but not least, my thanks and praise goes to my heavenly Father whose grace is all that I've needed, for his power is strongest when I am weak.

Table of Contents

Abstract.....	ii
Acknowledgments	iii
Table of Contents	iv
List of Tables.....	x
List of Figures	xii
1. Introduction.....	1
1.1. Scheduling Just-In-Time	1
1.2. Scheduling Problems	2
1.3. Scope of Research.....	4
1.4. Research Questions	5
1.5. Methodology.....	5
1.6. Notation.....	6
1.7. Thesis Outline.....	6
2. E/T Scheduling Literature Review.....	7
2.1. Introduction	7
2.2. Common Characteristics of E/T Scheduling.....	7
2.3. Classification of E/T Scheduling Problems	8
2.3.1. Due Date Considerations	9
2.3.2. Idle Time Considerations	9
2.3.3. Penalty Considerations.....	10
2.3.4. Processing Time Considerations.....	10
2.3.5. Other Extensions to the Model.....	10
2.3.6. Other Modifications to the General E/T Scheduling Problem.....	11
2.4. Common Due Date Literature.....	12
2.5. Distinct Due Date Literature.....	14
2.6. Discussion	17
2.6.1. The Relevance of Scheduling Theory to the Real World.....	17
2.6.2. Evaluation of E/T Scheduling Literature	18
2.7. Conclusions.....	19

Table of Contents continued

3. Search Techniques Literature Review.....	21
3.1. Introduction	21
3.2. Neighbourhood Schemes	23
3.2.1. Sequencing Neighbourhood Schemes.....	23
3.2.2. Problem Space/Heuristic Space Neighbourhood Schemes	24
3.3. Simulated Annealing	26
3.3.1. Introduction.....	26
3.3.2. Overview of the Method	26
3.3.3. Extensions to the Standard Method.....	29
3.3.4. Simulated Annealing Machine Scheduling Applications	30
3.3.5. Conclusions	35
3.4. Genetic Algorithms.....	36
3.4.1. Introduction.....	36
3.4.2. Overview of the Method	37
3.4.3. Genetic Operators.....	39
3.4.4. Genetic Algorithm Machine Scheduling Applications	43
3.4.5. Conclusions	47
3.5. Tabu Search	48
3.5.1. Introduction.....	48
3.5.2. Overview of the Method	48
3.5.3. Tabu Search Extensions	51
3.5.4. Tabu Search Machine Scheduling Applications	53
3.5.5. Conclusions	58
3.6. Summary and Conclusions.....	60
4. Idle Time Insertion for E/T Scheduling Problems	61
4.1. Introduction	61
4.2. Literature Review	62
4.3. Theoretical Development of an Idle Time Insertion Algorithm	64
4.4. Conclusions.....	67

Table of Contents continued

5. Early/Tardy Solution Space.....	68
5.1. Introduction	68
5.2. Theoretical Development of the E/T Solution Space	68
5.2.1. The Common Due Date Scheduling Problem	68
5.2.2. Distinct Due Date Problem	69
Property 1. No idle time is inserted.....	69
Property 2. The schedule is “V-shaped”	69
Property 3. One job is completed on the due date.	70
Property 4. Determining the Best Job to be Scheduled.....	71
5.3. The Distinct Due Date E/T Solution Space	72
5.4. Five Point Insertion Heuristic.....	74
1. Separate the early jobs from the late jobs.	75
2. Insert all early jobs into the schedule.....	75
3. Insert all the late jobs.....	75
5.5. Dynamic Loop Heuristic	78
5.5.1. Detailed Description of the Dynamic Loop Heuristic.....	78
5.5.2. Implementation of the Dynamic Loop	80
5.5.3. Critical Point Calculations.....	81
1. Swap Two Adjacent Jobs	81
2. Swap Two Jobs.....	83
3. Move a Job	84
4. Swap Two Adjacent Jobs to Absorb Idle Time	85
5.6. Evaluation of the Heuristics	86
5.7. Conclusions	89

Table of Contents continued

6. Search Technique Implementation..... 90

 6.1. Overview..... 90

 6.2. Simulated Annealing..... 90

 6.2.1. Standard Simulated Annealing 90

 6.2.2. E/T Simulated Annealing 91

 6.2.3. Early/Tardy Smart Simulated Annealing..... 91

 6.3. Genetic Algorithms..... 92

 6.3.1. Standard Genetic Algorithm..... 92

 6.3.2. E/T Genetic Algorithm 93

 6.4. Tabu Search 95

 6.4.1. Standard Tabu Search..... 95

 6.4.2. Standard Tabu Search With “Windows” 95

 6.4.3. Revised Tabu Search 96

 6.4.4. Revised Window Tabu Search 96

 6.4.5. Standard ET Tabu Search..... 97

 6.4.6. Revised ET Tabu Search..... 98

 6.4.7. Analyse Search 98

 6.4.8. Revised Analyse Search 99

 6.4.9. Revised “Smart” Early/Tardy Tabu Search..... 99

 6.4.10. Revised Window Tabu Search with Early/Tardy Extension 100

 6.4.11. Revised Early/Tardy and Standard Hybrid Tabu Search 100

 6.5. Common Due Date Tabu Search Methods..... 101

 6.5.1. CDD E/T Tabu Search..... 101

 6.5.2. CDD Revised E/T Tabu Search 101

 6.5.3. CDD Revised E/T Tabu Swap Search..... 101

 6.5.4. CDD Revised E/T Tabu Date Swap Search..... 102

 6.6. Summary..... 102

Table of Contents continued

7. Computational Experiments	106
7.1. Introduction	106
7.2. Data Generation and Analysis	106
7.2.1 Data Generation	106
7.2.2 Statistical Analysis	108
7.3. Idle Time Insertion	109
7.3.1. Introduction.....	109
7.3.2. Results.....	109
7.4. Common Due Date Problems.....	110
7.4.1. Introduction.....	110
7.4.2. Computational Experiment	110
7.4.3. Conclusions	114
7.5. Solution Space Computational Experiment	115
7.5.1. Introduction.....	115
7.5.2. Experiment Results.....	115
7.5.3. Conclusions	117
7.6. Small Problems.....	118
7.6.1. Introduction.....	118
7.6.2. Existing Heuristic Results	119
7.6.3. Simulated Annealing Results	120
7.6.4. Genetic Algorithms Results.....	122
7.6.5. Tabu Search Results	123
7.6.6. Search Technique Effects.....	126
7.6.7. Discussion and Conclusions	126
7.7. Large Problems	127
7.7.1. Introduction.....	127
7.7.2. Search Technique Experiment Results.....	128
7.7.3. Search Technique Effects.....	135
7.7.4. Effect of Using Different Starting Points.....	137
7.7.5. Comparison of ‘Intelligent’ Search Heuristics with Other Heuristics	140
7.7.6. Date Based E/T Neighbourhoods.....	145
7.7.7. Conclusions	148
7.8. Conclusions.....	148
8. Determining the Early/Tardy Ratio	151
8.1. Introduction	151
8.2. Early/Tardy Cost Components.....	151
8.3. Effect of Error in the α/β Ratio	153
8.4. Conclusions.....	158

Table of Contents continued

9. Summary and Conclusions 159

 9.1. Summary..... 159

 9.2. Conclusions..... 162

 9.3. Contributions..... 163

 9.4. Further Research 165

Appendix I Search Technique Examples..... 167

Appendix II E/T Heuristic Examples 170

Appendix III Common Due Date ANOVA Results 175

Appendix IV Distinct Due Date ANOVA Results for Large Problems..... 177

Appendix V Starting Point Sensitivity ANOVA Results..... 186

Appendix VI 30 Job Problem Analysis..... 187

Bibliography 196

List of Tables

1-1	Classification of Scheduling Problems.....	3
2-1	A Comparison of Distinct Due Date, E/T Scheduling Literature.....	16
5-1	Data for a Four Job Problem.....	73
5-2	Completion Times of Two E/T Specifications which Map onto the same Sequence.....	73
5-3	Advantages and Disadvantages of the two E/T Heuristics.....	86
6-1	Summary of SA Search Techniques.....	102
6-2	Summary of GA Search Techniques.....	102
6-3	Summary of TS Search Techniques.....	103
6-4	Summary of Modified TS Search Techniques.....	104
6-5	Summary of Starting Point TS Search Techniques.....	104
6-6	Summary of Common Due Date TS Search Techniques.....	105
7-1	Parameters for Each Class of Problem.....	107
7-2	Comparison of Idle Time Insertion Algorithm Times.....	109
7-3	Comparison of the Two E/T Heuristics and Mixed Integer Programming.....	116
7-4	Results from Previous Heuristics on Small Problems.....	119
7-5	Simulated Annealing Small Problem Results.....	120
7-6	Simulated Annealing Tukey HSD Analysis.....	121
7-7	Games and Howell Tukey Analysis between Simulated Annealing Techniques.....	122
7-8	Genetic Algorithms Small Problem Results.....	122
7-9	Genetic Algorithm Tukey HSD Analysis.....	122
7-10	Tabu Search Small Problem Results.....	124
7-11	Tabu Search Tukey HSD Analysis.....	124
7-12	Games and Howell Tukey Analysis on Tabu Search Methods.....	126
7-13	Analysis by Search Method.....	126
7-14	Games and Howell Tukey Analysis between Techniques at the End of the Search.....	136
7-15	Games and Howell Tukey ANOVA between Starting Points.....	139
7-16	Games and Howell Tukey Analysis between Heuristics.....	141
7-17	Schedule Costs of 30 Job Problems.....	143
7-18	Insert Moves from Optimal.....	144
7-19	The Total Number of Tardy Jobs in the Final Schedule.....	144
7-20	The Maximum Tardy Distance in the Final Schedule.....	145
8-1	ANOVA at With Different Error Factors.....	157

List of Tables continued

A1-1	Example Problem Data	167
A1-2	Simulated Annealing Search Iterations.....	167
A1-3	Genetic Algorithm Generations.....	168
A1-4	Tabu Search Iterations.....	169
A2-1	Example Problem Data	170
A3-1	Common Due Date ANOVA Significance Levels of Search Method	175
A3-2	Common Due Date ANOVA Significance Levels of Problem Class.....	176
A4-1	ANOVA of Simulated Annealing Techniques.....	177
A4-2	Genetic Algorithms ANOVA Analysis	178
A4-3	Revised Tabu Search ANOVA	179
A4-4	Revised Early Tardy Tabu Search ANOVA	180
A4-5	Revised Window Tabu Search, Early/Tardy Starting Point ANOVA.....	181
A4-6	Revised Window Tabu Search with Early/Tardy Extension ANOVA	182
A4-7	Early/Tardy Hybrid ANOVA.....	183
A4-8	Across Search Techniques ANOVA	184
A4-9	ANOVA Across Tabu Search Based Methods	185
A5-1	Tabu Search Starting Point Sensitivity	186

List of Figures

2-1	Venn Diagram of E/T Scheduling Problems	8
2-2	“V-Shaped” Property	12
3-1	Local Minima in a Solution Space.....	22
3-2	An Example of a Adjacent Pairwise Interchange Neighbour	23
3-3	An Example of a Swap Neighbour.....	23
3-4	An Example of an Insert Neighbour.....	24
3-5	Mapping Proposed by Storer <i>et al.</i> (1992).....	24
3-6	Basic Simulated Annealing Algorithm (Eglese, 1990)	27
3-7	Algorithm to Determine the Initial Control Parameter (Kirkpatrick <i>et al.</i> 1983)	28
3-8	A Disjunctive Graph of the Job Shop Problem (van Laarhoven <i>et al.</i> 1992)	32
3-9	The Basic Genetic Algorithm (Lawton, 1992).....	37
3-10	“Roulette Wheel Parent Selection” Procedure outlined in Davis (1991).....	40
3-11	Tabu Search Short Term Memory Algorithm (Glover, 1990c)	50
4-1	The Two Phases in E/T Scheduling.....	61
4-2	Corrected Idle Time Insertion Algorithm of Fry, Armstrong <i>et al.</i> (1987)	63
4-3	An example of the Idle Time Insertion Algorithm.....	66
4-4	Group Scheduling Algorithm.....	67
5-1	Determining the Order of Early Jobs.....	70
5-2	The Mapping between Solution Spaces.....	73
5-3	Algorithm for the Insertion of Early Jobs	75
5-4	Insertion of Late Jobs Into the Schedule	77
5-5	The Top Level of the Dynamic Loop Heuristic	78
5-6	Stage 1 of the Dynamic Loop Heuristic - Early Job Scheduling.....	78
5-7	Stage 2 of the Dynamic Loop Heuristic - Late Job Scheduling.....	78
5-8	Stage 3 of the Dynamic Loop Heuristic - Merging Early and Late Schedules	79
5-9	Heuristics for Calculating Schedule Costs at Step 3.6	80
5-10	Heuristic to Estimate the cost of Moving an Early Job Over Late Jobs.....	80
5-11	Algorithm to Move Early Jobs Back	81
5-12	Swap Job Critical Point	82
5-13	Critical Point for Swap with Out of Sequence Job.....	83
5-14	Critical Point to Swap Over Out of Sequence Jobs	84
5-15	Critical Point to Absorb Idle Time	85
5-16	Comparison of E/T Scheduling Heuristics.....	88
6-1	Sample Sequence.....	95
6-2	Searching With Invalid Solution Space	97
6-3	Analyse Algorithm.....	99

List of Figures continued

7-1	Common Due Date Search Deviation from Known Minimum over Time.....	111
7-2	Problem Class Solution Progress	113
7-3	Number of Times Solution Ranked First or Second for Common Due Date Problems	114
7-4	Cumulative Frequency Graph of the Deviation From Optimal for Each E/T Heuristic.....	117
7-5	Tukey One Way ANOVA by Problem Class	117
7-6	SA Average Time and Deviation for Large Distinct Due Date Problems	129
7-7	GA Average Time and Deviation for Large Distinct Due Date Problems.....	130
7-8	TS Average Time and Deviation for Large Distinct Due Date Problems.....	135
7-9	Search Techniques Average Time and Deviation for Large Distinct Due Date Problems	136
7-10	TS Starting Point Experiment Average Time and Deviation for Large Distinct Due Date Problems.....	138
7-11	Comparison of Heuristics	141
7-12	Date Swap vs Standard E/T Tabu.....	147
7-13	Problem Class Solution Progress	147
8-1	Effect of Error in the Early/Tardy Ratio.....	153
8-2	Aggregated Effect of Error in the Early/Tardy Ratio.....	154
8-3	Estimation of Error on the Early/Tardy Ratio for Low Tardy Jobs	155
8-4	Estimation of Error on the Early/Tardy Ratio for High Tardy Jobs.....	155
A2-1	Problem IA With 5 Point Insertion.....	173
A2-2	Problem IA With Dynamic Loop	174
A6-1	Cost Analysis of a Class G Problem.....	187
A6-2	Cost Analysis of a Class H Problem.....	187
A6-3	Cost Analysis of a Class I Problem	188
A6-4	Cost Analysis of a Class J Problem	188
A6-5	Cost Analysis of a Class K Problem.....	189
A6-6	Cost Analysis of a Class L Problem	189
A6-7	Schedules of a G Class Problem.....	190
A6-8	Schedules of a H Class Problem.....	191
A6-9	Schedules of a I Class Problem.....	192
A6-10	Schedules of a J Class Problem.....	193
A6-11	Schedules of a K Class Problem.....	194
A6-12	Schedules of a L Class Problem.....	195

1. Introduction

1.1. Scheduling Just-In-Time

In the 1960's Toyota Motor Corporation introduced the concept of Just-In-Time (JIT) (Schniederjans, 1993) which aims to eliminate waste from within an organisation. Raw materials, work-in-progress and finished goods inventories have all become the target for rationalisation as they often contribute significantly to waste. Waste is not only expensive, but more importantly it can hide the real problems facing an organisation. JIT strives to deliver orders to customers at the time specified by the customer. Not before, as this can create unnecessary finished goods inventory costs, and not after, as this can lead to a loss of goodwill with customers.

It is for the JIT environment that the area of Early/Tardy (E/T) Scheduling has developed. E/T scheduling discourages the completion of a job before its due date and after its due date, as both of these events incur a cost to the organisation. Research to date has focused on problems that have very restrictive assumptions, such as having a common due date for all jobs or having predefined relationship between the cost of producing a job early and the cost of producing a job late. These restrictions have severely impeded the applicability of these models to practical problems. The more general problem, with no restrictions on due dates or penalties, is therefore more versatile and useful to industry.

The problem we consider in this thesis is *The Generalised Single Machine, Distinct Due Date, Early/Tardy Scheduling Problem* and is subsequently referred to in this thesis as the *E/T Scheduling Problem*. Little research has been conducted on this general problem. In their review of early/tardy scheduling, Baker and Scudder (1990) conclude that "With respect to the model with distinct due dates, there are apt to be few special cases where efficient solution procedures can be found". Baker and Scudder (1990) then comment on the methods used in solving the general problem by stating that "Among heuristic procedures, the existing research seems to have examined only a relatively simple pairwise interchange approach; therefore, it would be interesting to explore more sophisticated heuristic methods." The exploration of sophisticated heuristics which can solve the generalised E/T scheduling problem is the focus of this research.

The aim of this research is to construct and analyse a more general E/T scheduling model, which to date has had little theoretical development, and to provide a solution method for it that can provide high quality solutions to practical-sized problems in a reasonable amount of computational time. The E/T scheduling models considered in the literature, and the model that is developed here, do not reflect all the complexities of the shop floor. It is hoped, however, that this research will provide a building block towards the construction of models which are useful to schedulers in industry.

1.2. Scheduling Problems

“Scheduling is the allocation of resources over time to perform a collection of tasks”

(Baker, 1974, p 2)

Scheduling is concerned primarily with resource allocation, whatever those resources may be (production facilities, machines, rooms, staff, etc) to particular tasks (jobs, orders, presentations, etc). It is therefore concerned with predefined tasks that need to be performed and the predefined resources that can be used to perform these tasks. It is not concerned with the planning function of the organisation which determines the amount of resources that are to be provided. These two functions are, however, interrelated as one will determine the success of the other.

The terminology used throughout this thesis is that used in manufacturing, therefore *resources* are called *machines* and *tasks* are called *jobs*. This does not, however, in any way reduce the applicability of the theory or techniques developed to the scheduling of other types of resources and/or tasks.

Baker (1974) divides the scheduling problem into two types of decisions:

1. Allocation decisions which address which job will be assigned to which machine, and
2. Sequencing decisions which address the order in which jobs are to be performed and, subsequently, when each task will be performed.

The way in which these two decision areas are addressed often varies between different scheduling problems. As the problem considered in this thesis deals with a single machine, only the second type of decision is addressed here.

There is a vast amount of literature on many different scheduling problems. To organise this literature many different classification schemes for scheduling problems have been developed. One of the more popular schemes is the three field problem classification scheme proposed by Graham, Lawler, Lenstra and Rinnooy Kan (1979) which classifies the problem by machine information, job information and the objective criteria to be minimised. This classification scheme has been used in Table 1-1 to illustrate where the E/T scheduling problem fits within the entire body of scheduling literature. The elements which are relevant to E/T scheduling are in bold. The notation used is described in Section 1.6.

Machine Environment	Optimality Criteria (Minimise)	Job Characteristics
Single Machine	Regular Objective Functions	Unit processing time
	Makespan	No restraints on processing times
Multi-Machine	Mean Flow Time	Stochastic Processing Times
Flow shop	Lateness	Deterministic Processing Times
Identical	Tardiness	
Uniform		Due Date Determination
Unrelated	Non-Regular Objective	Due Date(s) Given
Job Shop	Functions	Common Due Date
	$\Sigma(E_i+T_i)$	Generalised Due Dates
	$\Sigma(\alpha E_i+\beta T_i)$	Distinct Due dates
	$\Sigma(\alpha_i E_i+\beta_i T_i)$	
		Early/Tardy penalty proportional
		Early/Tardy penalty common for all jobs
		Early/Tardy penalties distinct for all jobs
		Preemption Allowed
		Precedence Relationships allowed
		Ready times for jobs can be specified
		Sequence dependent setup times

Table 1-1. Classification of Scheduling Problems.

The sequencing decision part of the scheduling problem is related to another classical problem called the *Travelling Salesman Problem (TSP)* (Garey and Johnson, 1979). This problem is concerned with a salesman who has a number of cities he needs to visit once and only once to sell his products. The objective is to determine the order in which cities should be visited in order to minimise the total distance travelled. The main difference between the TSP and sequencing is that the TSP is cyclic. That is, there is no real starting point for the sequence as the last city visited is always connected to the starting city forming a cycle. In the sequencing problem there is always a distinct starting job and a distinct ending job.

Time complexity is commonly used to measure problem difficulty. Time complexity is a measure of the number of steps required by an algorithm to solve a given problem. Complexity is related to the size of the problem, such as the number of cities in the TSP or the number of jobs in a scheduling problem. There is strong evidence that for some problems, such as the TSP, no algorithm exists with a complexity function which is polynomial in terms of the problem size. Problems for which no algorithm has been found with a polynomial complexity function have not been “well solved” and are *intractable* (Garey and Johnson, 1979). These types of problems belong to the class of problems known as *Nondeterministic Polynomial time complete* problems or *NP-complete* problems.

However, a prerequisite of a problem being NP-complete is that it belongs to the class of Nondeterministic Polynomial time (NP) problems, which requires that for each instance of a problem, there is an algorithm which can evaluate or check the instance in polynomial time. When NP-complete problems can be reduced to a problem which does not belong to the class of NP, the problem is said to be *NP-hard* (Papadimitriou and Steiglitz, 1982). These problems are at least as difficult to solve as NP-complete problems. For a comprehensive discussion on NP-completeness, the reader is referred to Garey and Johnson (1979). Garey, Tarjan and Wilfong (1988) proved that the early/tardy scheduling problem is NP-complete and therefore relatively difficult to solve.

As the early/tardy scheduling problem is NP-complete optimisation methods, such as mixed integer programming and dynamic programming, are unable to solve general problems of any practical size within a reasonable amount of computational time. The simplest methods of solving scheduling problems, such as the E/T scheduling problem, use a dispatch heuristic to determine the next job to be processed by certain criteria, for example, by the earliest due date of the remaining jobs to be processed. These heuristics do not attempt, in any way, to optimise the objective function and therefore obtain results which often vary dramatically in quality. By using more complex rules for determining how jobs are to be scheduled, better schedules can be created. The solution methods that are the main focus of this thesis are search techniques which start at an arbitrary solution and search for better schedules, which are evaluated according to the objective function provided.

1.3. Scope of Research

This thesis investigates the generalised distinct due date, E/T scheduling problem. It also discusses the generalised common due date E/T scheduling problem in the development of theory for the distinct due date problem. However the common due date problem is not the primary focus of this research.

The main assumptions of the generalised, single machine, distinct due date, E/T scheduling problem are:

1. The problem deals with only a single machine
2. The variables of each job are known with certainty, that is they are deterministic.
3. All jobs are ready to be processed at time zero.
4. Set up times are independent of the sequence of jobs and are therefore included as part of the processing time of the job.
5. Set up costs are constant between jobs and are independent of the sequence of jobs.

Assumption 1 can be justified in that Goldratt and Cox (1986) propose that bottleneck machines should be the main focus of scheduling, as the processing on these critical machines are the ones that will ultimately determine the delivery time of the final product.

Assumptions 4 and 5 can also be justified in theory, if not in practice, as one of the overall objectives of Just-in-Time is to reduce setup costs and times to zero.

Our analysis of the early/tardy scheduling solution techniques deals with only randomly generated, theoretical problems. The use of these techniques in an industrial setting is not going to be considered in this research.

1.4. Research Questions

Three main research questions are addressed in this thesis. These are:

1. Is there a method of designing a search technique which will exploit the nature of the E/T scheduling problem so that problems can be solved more quickly than with traditional search techniques?
2. Which, among the three following search techniques: simulated annealing, genetic algorithms and tabu search, is the most effective at solving the E/T Scheduling problem?
3. What is the effect of errors in the earliness penalty and in the tardiness penalty on the quality of the final solution?

1.5. Methodology

To answer research questions 1 and 2 in Section 1.4., a suite of randomly generated test problems are developed. One set of problems contains small-sized problems which can be solved optimally using mixed integer programming. A second suite of larger problems is used to test the search heuristics under a load similar to that found in industry. Tests are carried out on both small and large problems to compare the performance of the three techniques. By calculating the deviation of each method from either the optimal solution or the best known solution a comparable performance index for each technique is obtained.

To test the effect of errors in the earliness penalty and tardiness penalty, a set of small problems is developed which differ by a given value which simulates the error in the earliness penalty : tardiness penalty ratio. These problems are then solved to optimality. By reevaluating the solutions obtained using the original earliness and tardiness penalties a picture can be built up of how large the error in the earliness penalty : tardiness penalty ratio can become before solution quality is affected significantly.

1.6. Notation

Throughout this thesis, unless otherwise stated, the following notation is used:

n	=	The number of jobs to be scheduled.
c_j	=	The completion time of job j .
d_j	=	The due date of job j .
p_j	=	The processing time for job j .
α_j	=	The penalty per unit of time of producing job j early.
β_j	=	The penalty per unit of time of producing job j late.
E	=	The set of all jobs that are early.
T	=	The set of all jobs that are tardy.
E_j	=	The amount of time job j is early, that is $ d_j - c_j ^+$.
T_j	=	The amount of time job j is tardy, that is $ c_j - d_j ^+$.
$ x ^+$	=	The maximum of x and 0.

1.7. Thesis Outline

After this introductory chapter, Chapter 2 provides an indepth literature review of E/T scheduling, showing how the generalised, distinct due date E/T scheduling problem fits into the wider area of E/T scheduling. Chapter 3 provides an indepth literature review of the three main search techniques: simulated annealing, genetic algorithms and tabu search; and their application to scheduling problems. Chapter 4 discusses the importance of idle time insertion in E/T scheduling, presenting corrections to an already published algorithm, and developing a new algorithm. Chapter 5 develops theoretical results for the E/T scheduling problem and develops two heuristics which can identify a schedule by defining whether a job is to be scheduled early or late. Chapter 6 discusses the way in which each of the search techniques were implemented for the computational experiments which are described in Chapter 7. Chapter 8 discusses the effect of errors in the earliness and tardiness penalties. Chapter 9 summarises the main points found in this research and discusses the contribution that this research has made, both in the fields of scheduling and search techniques. The chapter concludes by discussing areas of potential further research which follow on from this research.

2. E/T Scheduling Literature Review

2.1. Introduction

In this chapter we review the current literature on all deterministic E/T scheduling where the objective is to minimise a function which incorporates the deviation of the completion time from a given due date.

Section 2.2. outlines the common characteristics of E/T scheduling which differentiate it from other forms of scheduling. Section 2.3. outlines the major classifications of E/T scheduling problems and graphically shows how these classes are interrelated. Section 2.4. outlines the literature which examines the common due date (CDD) problem while also summarising the unique characteristics of this type of problem. Section 2.5. examines the literature concerning the distinct due date problem and summarises its results. Section 2.6. discusses the relevance of scheduling theory to real world problems and explicitly examines the E/T scheduling literature to see how applicable it is to practical problems. The conclusions based on this analysis are then presented in Section 2.7.

2.2. Common Characteristics of E/T Scheduling

The tardiness scheduling problem recognises that costs are incurred by the organisation when orders are not delivered to the customer on time. Late production of an order can lead to a loss of goodwill with the customer and a potential loss of future business. The cost of producing a job late is normally considered to be proportional to the amount of time by which it is late. A full discussion on tardiness costs and the elements that make up the tardiness penalty can be found in Chapter 8.

During the 1960's the Toyota Motor Company started to develop what is now known as Just-In-Time (JIT) manufacturing. JIT is a philosophy that concentrates on increasing productivity by:

1. decreasing the material cost component of production, such as holding cost and setup costs,
2. decreasing scrap rates, and
3. increasing worker motivation (Schonberger 1982).

It has been defined as “the successful completion of a product or service at each stage of production activity from vendor to customer just-in-time for its use and at a minimum cost” (Schniederjans 1993, p 4). JIT is now respected by manufacturers as a winning strategy, causing many industries to implement JIT strategies to remain competitive (Schniederjans 1993).

The E/T Scheduling problem extends the Tardiness Scheduling problem to a JIT type scheduling environment where not only is producing an order late discouraged, but the production of an order early is discouraged as well (Baker and Scudder, 1990). In a JIT scheduling environment, customers may not accept the early delivery of goods and, even if they do, they may not be willing to be billed for the goods until the original due date. In either case, the firm incurs several costs when an order is produced early. For example the financial cost due to the extra investment in finished goods inventory, the cost involved with extra storage facilities and the cost of product spoilage. For a full discussion of the elements that make up the early and tardy penalties refer to Chapter 8.

Garey, Tarjan and Wilfong (1988) proved that the problem of finding minimum total deviation of the completion time from the due date is NP-complete. This problem is defined, using the notation outlined in Section 1.6, by an objective function of:

$$\min \sum_{i=1}^n |c_i - d_i|$$

Garey *et al.* (1988) also proved that when all the jobs had the same processing time the problem is easily solved. In this case earliest due date ordering is optimal.

2.3. Classification of E/T Scheduling Problems

E/T scheduling literature can be divided into several broad categories which overlap with each other. These categories deal with restrictions on due dates, processing times, earliness and tardiness penalties, the insertion of idle time and the consideration of sequence dependent setup times. A Venn diagram of the main interactions between these categories is presented in Figure 2-1, again using the notation outlined in Section 1.6.

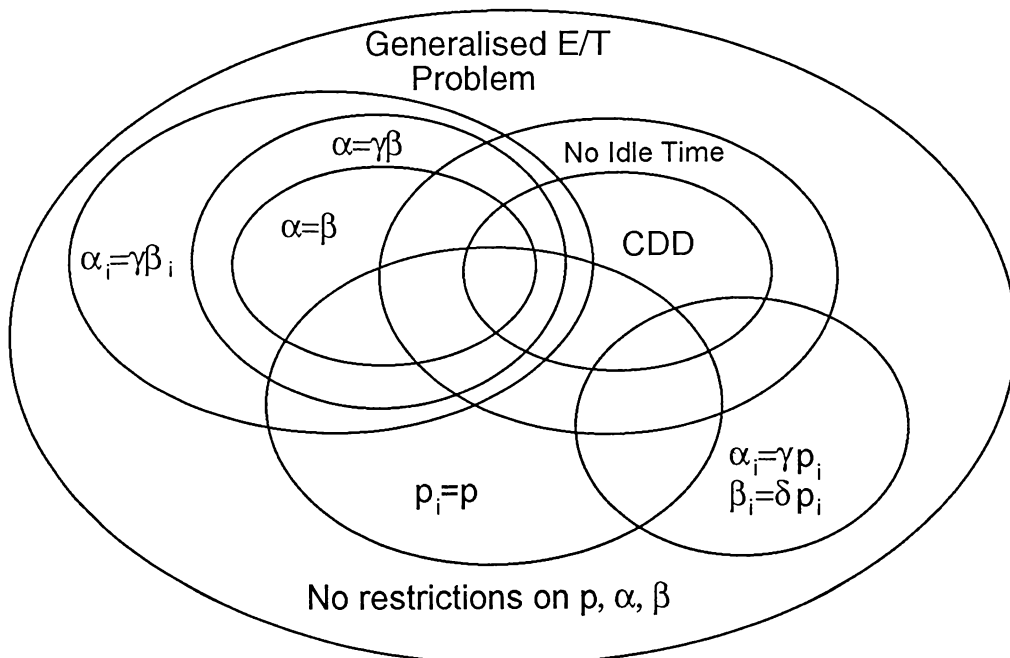


Figure 2-1. Venn Diagram of E/T Scheduling Problems

2.3.1. Due Date Considerations

In the generalised problem no restrictions are placed on the due dates of the job. A restricted model of this problem is the Common Due Date Problem which specifies that all jobs must have the same due date. This modification is popular as it introduces several unique properties which can be exploited by solution techniques.

A common derivation of the common due date problem is the Due Date Determination problem, whereby the common due date becomes a decision variable to be optimised along with the sequence. A survey of due date determination literature was carried out by Cheng and Gupta (1989) from which they developed a framework for classifying problems involving due date determination.

2.3.2. Idle Time Considerations

In the generalised problem, idle time is allowed within the schedule, however in some of the problems found in the literature this is not the case. Some examples of this type of problem are discussed by Abdul-Razaq and Potts (1988) and Ow and Morton (1989), however Baker and Scudder (1990) state that this assumption “is inconsistent with the E/T criterion”. In general the elimination of idle time speeds up the time required to evaluate a solution, as idle time does not need to be calculated, however the overall cost of the schedules produced may be higher than those which have idle time inserted. With this approach, the quality of the solution is sacrificed for the speed of the algorithm. For some types of problems, such as those with: very high tardiness penalties, very small earliness penalties and due dates which are very tight; this may be a valid assumption, but for the general problem there needs to be a balance between the speed of the algorithm and the solution quality.

To maintain a balance between speed and solution quality Robb and Silver (1994) implemented an idle time insertion routine that inserts idle time only at the start of the schedule and does not allow idle time to be inserted between jobs. Davis and Kanet (1993) discovered in their computational experiment involving 16 different problems, that only 2 contained idle time within the schedule, while the remaining 14 had either no idle time or only had idle time at the start. It must be noted that their 16 problems were based on the cases developed by Baker (1974) with a tardiness factor of 0.2 and due date range of between 0.087 and 0.778 (see Section 7.2 for a full description of data generation for these problems). The tardiness factor controls the amount of idle time at the start of the schedule while the due date range controls the amount of idle time within the schedule. Baker and Martin (1974) use due date range values of 0.2 and 0.95 in their experiments, values which are higher, and therefore more prone to idle time insertion within the schedule, than those used by Davis and Kanet (1993). The due date ranges used for these problems may not, therefore be representative of all E/T scheduling problems.

2.3.3. Penalty Considerations

In the generalised problem the early and tardy penalties can be unique for each job if required. Often in the literature, the E/T Scheduling problem has been simplified by making the early penalty and the tardy penalty the same for all jobs (Fry, Armstrong and Blackstone, 1987). A further simplification is also made by assuming that the early penalty and tardy penalty are the same (Garey *et al.*, 1988). This category of problems also includes those problems that ignore explicit early/tardy penalties but use a distance measure which assumes an earliness penalty of 1 and a tardiness penalty of 1.

Yano and Kim (1991) consider the problem where earliness and tardiness penalties are proportional to the processing time of the jobs, where it is assumed that the proportions are constant for all jobs. They claim that this assumption is valid as the longer the processing time is for the job, the greater the amount of 'value' added to the job, therefore the more valuable the job is and the more important it is. However the importance of a job does not depend entirely on its value, therefore this argument could easily be refuted in many circumstances. A full discussion on the elements that need to be considered when determining the early and tardy penalties is given in Chapter 8.

To generalise the E/T scheduling problem non-linear penalty functions can be considered. However only quadratic functions have been considered in the literature to date (Cheng, 1984; Bagchi, Sullivan and Chang 1987; De, Ghosh and Wells 1993). Linear functions remain dominant in the literature, presumably because of their more favourable mathematical properties.

2.3.4. Processing Time Considerations

The general model allows for unique processing times for every job. An assumption made in some of the literature, such as Garey *et al.* (1988), is that all jobs have the same processing time. This simplifies the problem considerably, enabling the problem to be solved in polynomial time (Garey *et al.*, 1988).

2.3.5. Other Extensions to the Model

Some of the extensions that could be applied to the E/T scheduling problem which have been reported in the literature are:

- The use of sequence dependent setup times. The sequence in which the orders are produced will in many cases influence the setup cost incurred for the job. Although JIT philosophy tries to eliminate setup costs (Schniederjans, 1993), in reality setup costs do occur and models that accommodate this factor have been developed by ten Kate, Wijngaard and Zijm (1991) and Coleman (1992).

- Inclusion of ready times for each job. In many situations not all orders can be produced immediately. Some orders will have to wait until raw materials arrive from suppliers or until they have been processed on other machinery (Keyser and Sarper 1991).
- Inclusion of waiting costs. The waiting cost incurred is proportional to the difference between the time when a job is ready to be processed and the time when it starts to be processed. Including waiting costs in the objective function is seen as a means of minimising work-in-progress inventory if the machine being scheduled is one in a series required for a job to be completed (Keyser and Sarper, 1991).
- The use of stochastic, rather than deterministic processing times. This tries to reflect the uncertainty of the processing that exists when producing an order. Examples of stochastic E/T scheduling can be seen in Sarin, Erel and Steiner (1991) and Robb and Silver (1994).
- The use of due date windows. These reflect the fact that as long as an order is delivered within a specified time range, no penalty will be incurred (Kramer and Lee, 1992).

All of these extensions to the E/T scheduling problem make it more applicable to practical scheduling problems. Unfortunately the literature in these areas has often extended the model in one area and placed other restrictions on it in another, for example ten Kate *et al.* (1992) introduce sequence dependent setup times but assume that the earliness penalty and tardiness penalties are the same for all jobs. When other assumptions have not been placed on the problem, the solution methods have only been proven on small problems, for example Keyser and Sarper (1991) proved their algorithm on problems with only 6 jobs.

2.3.6. Other Modifications to the General E/T Scheduling Problem

Many other modifications to the general E/T scheduling problem have been reported in the literature. One such modification is to consider minimising the maximum early/tardy penalty rather than minimising the total early/tardy penalty. Examples of research into this class of problem are Sidney (1977) and Ferris and Vlach (1992), who developed polynomial time algorithms to solve this problem when all jobs incur the same earliness and tardiness penalties. Cheng (1991) considers this problem combined with due date determination. He proves that the only sequences that need to be considered for this problem are the earliest due date (EDD) and the shortest processing time (SPT) sequences, when the due date is assigned using the total-work-content due date assignment method.

Hall, Sethi and Sriskandarajah (1991) consider E/T scheduling problems with “generalised due dates”. Jobs in this class of problem do not have specific due dates, instead there exists generalised due dates whereby before the first due date - one job must be completed, before the second - two jobs must be completed, and before the n th due date n jobs must be completed.

Chand and Schneeberger (1988) consider the problem that involves *deadlines* rather than due dates. A deadline differs from a due date in that if a job is scheduled after its deadline the schedule is infeasible. This is equivalent to the generalised E/T scheduling problem with relatively large tardiness penalties.

By making early/tardy penalties proportional to the processing time of the job, Ahmed and Sundararaghavan (1990) have developed an algorithm to solve the unrestricted common due date problem. A similar approach is taken by Yano and Kim (1991) to solve the distinct due date problem while also allowing for penalty differences when a job is scheduled early or late.

2.4. Common Due Date Literature

The common due date problem is an important subset of the generalised E/T scheduling problem. This form of the problem has been well researched and has some unique properties which assist in its solution (Baker and Scudder 1990). These properties are:

1. No idle time is inserted within the schedule.
2. The schedule is “V-shaped”; that is, early jobs are sequenced in non-increasing order of the ratio p_i/α_i and late jobs are sequenced in non-decreasing order of the ratio p_i/β_i . See Figure 2-2.

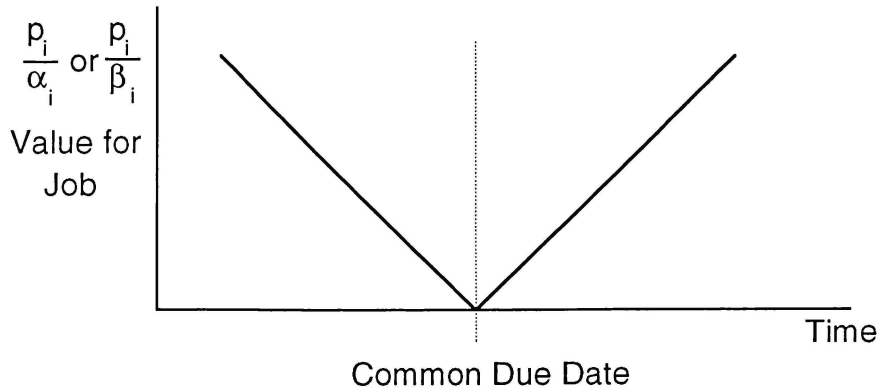


Figure 2-2. “V-Shaped” Property

3. One job is completed on the due date.
4. The b^{th} job in the sequence is completed on the due date, where b is the smallest integer satisfying the inequality:

$$\sum_{j=1}^b (\alpha_j + \beta_j) \geq \sum_{j=1}^n \beta_j.$$

Literature on the common due date problem can be classified according to the assumptions about the earliness and tardiness penalties of each job. The simplest form of the problem is to minimise the absolute deviation from the common due date. Effectively this problem has an implied earliness penalty of 1 and tardiness penalty of 1.

The unrestricted form of this problem, where jobs are not restricted in their orderings by the due date, was originally examined by Kanet (1981) who proved that jobs scheduled before the due date must be in longest processing time (LPT) order and those scheduled after the due date must be in shortest processing time (SPT) order. He also showed that at least one of the jobs must be scheduled on its due date. Sundararaghavan and Ahmed (1984) extended the work by Kanet (1981) to include multiple machines and also to consider schedules that begin at time 0. Hall (1986) further developed this model by designing an algorithm that finds multiple optima for the unrestricted case and then extended the theory to include multiple identical processors.

The restrictive form of the common due date problem was first addressed by Bagchi, Sullivan and Chang (1986). They developed an enumeration method for finding multiple optima for this problem. Szwarc (1989) developed bounds for the procedure of Bagchi *et al.* (1986) and implemented a branch and bound procedure to solve the problem. Hall, Kubiak and Sethi (1991) proved that this problem is NP-complete and then developed a dynamic programming approach to solving this problem which runs in pseudopolynomial time. De *et al.* (1993) also considered a general restricted case where the deviations were raised to some power. They developed a dynamic program which ran in pseudopolynomial time to solve the problem.

The next class of problem found in the literature is the weighted early/tardy scheduling problem, whereby the penalty of each job is individually weighted to provide a means of differentiating between important and less important jobs. Hall and Posner (1991) develop several properties for this problem, some of which are outlined above, while others are unique to this the weighted early/tardy problem. After proving that the unrestricted problem is NP-complete, Hall and Posner (1991) devise a computationally efficient dynamic programming algorithm to solve it and discuss special cases for which it can be solved in polynomial time.

Kubiak, Lou and Sethi (1990) use a transportation model to solve the mean absolute deviation problem where there are different weightings for the earliness penalties and the tardiness penalties. This type of problem can reflect the notion that to be early may not be as costly as being late, however in this formulation there is no method of distinguishing between important jobs and less important jobs.

The general form of the common due date E/T scheduling problem, when each job has its own early penalty and its own tardy penalty, is discussed in Baker and Scudder (1990) where properties 2 and 4 of the common due date model, outlined on page 12, are developed into their general form. Current literature dealing with this problem has

concentrated on the determination of the common due date rather than scheduling to a given due date. This is clearly an area of the literature that has potential for further research.

2.5. Distinct Due Date Literature

The distinct due date early/tardy machine scheduling problem differs from the common due date problem in that each job can have a different due date. Clearly the common due date problem is a subset of the distinct due date problem and therefore an algorithm that can solve the distinct due date problem efficiently will also be able to solve the common due date problem. Baker and Scudder (1990) point out that there has not been as much progress made with the distinct due date problem when compared to the research into the common due date problem. Distinct due date research is more fragmented than the common due date literature with most of the literature developing from either modifications of tardiness problem solution methods or the introduction of completely new methods rather than the refinement of other research which is often found in the common due date literature.

Garey *et al.* (1988) were the first to show that the distinct due date early/tardy scheduling problem, which minimises the total deviation of the completion time from the due date of each job, is NP-complete. This is the simplest form of the distinct due date model, and hence more complex problems involving job weights are also NP-complete.

Research in this area can be classified into two distinct groups, those that consider idle time and those that do not. Excluding idle time simplifies the problem and enables algorithms to achieve faster solution times. It is clear that by excluding idle time, the results achieved will not be optimal because of the non-regular objective function of the problem, however in certain circumstances, when there is little excess production capacity, this approach may be justified.

Abdul-Razaq and Potts (1988) investigate the distinct due date problem without idle time. They develop a branch and bound approach which uses a dynamic programming formulation to derive good lower bounds on the problem. With this formulation they were able to solve problems involving up to 25 jobs. They note that problems with more than 30 jobs would require excessive computational time.

Ow and Morton (1989) also investigated the distinct due date problem without idle time. They developed two different priority rules to determine the sequence of jobs, one was called LIN-ET, the other was called EXP-ET. They also developed another technique based on a filtered beam search which is similar to a branch and bound procedure but investigates k different schedules in parallel, where k is the 'length of the beam'. Each

sequence is selected on the basis of an evaluation function which estimates how good the sequence is. In their experiment they used one of their priority rules, EXP-ET, as their evaluation function. They solved problems involving up to 25 jobs and found that their dispatching rule performed significantly better than the earliest due date dispatching rule, while the performance of the filtered beam search was superior to all of the dispatching rules.

Literature that considers the distinct due date problem with idle time insertion has often concentrated on inserting idle time in a given sequence of jobs. Linear programming formulations have been provided to solve this problem (Fry, Armstrong *et al.*, 1987; Fry, Leong and Rakes, 1987), while other more efficient algorithms have also been developed. Fry, Armstrong *et al.* (1987) developed a one pass method of the simplex algorithm. Garey *et al.* (1988) developed an algorithm that would run in $O(n \log n)$ time. Yano and Kim (1991) developed a dynamic programming approach to the problem, while Davis and Kanet (1993) develop an algorithm similar to that proposed by Fry, Armstrong *et al.* (1987) which ran in $O(n^2)$ time.

Fry, Armstrong *et al.* (1987) examined the distinct due date problem when all earliness penalties are the same and all tardiness penalties are the same, ie. $\alpha_i = \alpha$ and $\beta_i = \beta$. In their paper they use a starting sequence given by either the earliest due date or the minimum slack dispatching rules. After optimally inserting idle time they check if adjacent jobs should be swapped to obtain a schedule with a lower cost in a process commonly called Adjacent Pairwise Interchange (API). To speed this process up they use dominance properties which they develop for this particular problem which, if met, mean the two jobs are swapped straight away and, if not met, the move is tested by performing the swap. This procedure continues for all jobs until a local minima is reached. The procedure was tested on problems with between 8 and 15 jobs and was found to provide good quality results.

Fry, Leong *et al.* (1987) consider a similar problem to Fry, Armstrong *et al.* (1987) but also include in their objective function a penalty for waiting time, that is the time between when a job is processed and when a job is ready to be processed. This adds to the objective function an allowance for the cost of work-in-progress inventory. They developed a branch and bound procedure to solve this problem which was able to solve problems with up to 15 jobs but found that solution times grew exponentially as problem sizes increased, making this method infeasible for large problems.

Yano and Kim (1991) examine the distinct due date problem when the earliness penalty and the tardiness penalty are both proportional to the processing time of the job, ie. $\alpha_i = \alpha' p_i$ and $\beta_i = \beta' p_i$ where α' and β' are constants. They develop five different heuristics, four of which are based on dispatching rules and one which takes the best solution from the four other

heuristics and then uses API to decrease the cost of the schedule. These heuristics were then used to solve problems ranging in size from 7 to 40 jobs. The solutions from these heuristics are then compared with the best solution found by a branch and bound algorithm which had a time limit of 1 CPU hour. The combined heuristic was found to produce very high quality results, only failing to find the optimal solution in one of the one thousand test problems.

Keyser and Sarper (1991) examine the general distinct due date problem which includes job dependent penalties for waiting time and early/tardy penalties. They develop a heuristic to solve this problem which is based on trying three different dispatching rules and performing adjacent pairwise interchange on the schedules to obtain a local minima. Two different methods of handling idle times are used, one ignores inserting idle time, the other inserts idle time so that the next job in the sequence is scheduled on or after its due date. Neither method provides optimal idle time insertion. The schedule with the lowest cost is chosen as the final solution. The results of the heuristic are then compared with a mixed integer programming formulation. The technique was tested on 48 randomly generated 6 job problems, larger problems could not be tested due to limitations on computational resources. It was found that the heuristic had an average deviation of 0.354% from optimal with a worst case deviation of 7.66% deviation.

Table 2-1 summarises the distinct due date E/T scheduling literature, their assumptions and their method of solution.

Paper	Penalty Assumptions	Idle Time Assumptions	Solution Method
Abdul-Razaq and Potts (1988)	None	No Idle Time	Dynamic Programming
Ow and Morton (1989)	None	No Idle Time	Filtered Beam Search
Fry, Armstrong <i>et al.</i> (1987)	$\alpha_i = \alpha,$ $\beta_i = \beta$	Idle Time Inserted	Heuristic with API gradient descent
Fry, Leong <i>et al.</i> (1987)	$\alpha_i = \alpha,$ $\beta_i = \beta$	Idle Time Inserted	Branch and Bound
Yano and Kim (1991)	$\alpha_i = \alpha'p_i,$ $\beta_i = \beta'p_i$	Idle Time Inserted	Heuristic and API gradient descent
Keyser and Sarper (1991)	None	Idle time for MIP, Heuristic Idle Time Insertion for Heuristic Solver	Mixed Integer Programming, and Heuristic API gradient descent

Table 2-1. A Comparison of Distinct Due Date, E/T Scheduling Literature

Baker and Scudder (1990) noted that the distinct due date problem has two main parts, the sequencing of jobs and the insertion idle time. All literature to date has followed this model

in the implementation of heuristics hence the emphasis on idle time insertion algorithms. As Yano and Kim (1991) noted, “the complexity of the idle time insertion algorithm is important since it may have to be executed many times to find an optimal sequence”. Baker and Scudder (1990) also suggest that the distinct due date problem may be attacked by decomposing it into subproblems with equal due dates making it possible to exploit the “tractability of problems with a common due date”.

2.6. Discussion

2.6.1. The Relevance of Scheduling Theory to the Real World

McKay, Safayeni and Buzacott (1988) challenged the relevance of job-shop scheduling theory to real world scheduling problems by stating, “... the theoretical formulation of the [job-shop scheduling] problem may be irrelevant. It does not appear to capture the essence of the job-shop scheduling problem faced by schedulers, and consequently, the research results have little applied value”. They then outline six issues which are not addressed in theoretical job-shop scheduling.

1. Schedules must meet many stated and unstated conflicting goals which must be formulated from incomplete, ambiguous or unreliable data.
2. Unexpected events are common on the shop floor such as machine break downs, staff becoming sick or injured or problems with raw materials.
3. Schedulers have many options available to them when they construct their schedules, most of which are not included in theoretical job-shop scheduling research, such as providing a short-term increase in capacity by outsourcing processing performed by a bottle-neck machine.
4. Schedulers use intuition to gain insight into what is actually happening on the shop floor.
5. Processing issues, such as the quality of work provided by different shifts, can affect the scheduling of different parts of the shop at different times for different reasons.
6. Schedulers already use simple dispatching logic to avoid specific long term scheduling which is contrary to the dynamic nature of the shop floor.

Grant (1986) comments on how Operations Research (OR) methodologies tend to make people forget their existing way of doing things and consider from scratch what is required to be done. This, Grant claims, “tends to downgrade the subjective element of human activities”. What OR methodologies can do is trivialise complex real problems so that they can be solved by mathematical modelling and therefore ignore the many other factors which need to be considered in the problem. It is the author’s opinion that this problem is evident in the area of scheduling.

Oakland and Sohal (1987) carried out a survey on the usage of production management techniques within organisations. They found that the three major barriers to the acceptance of production management techniques existed where people:

1. had no knowledge of the technique,
2. were successful without using the technique and
3. felt that the technique was not applicable to their organisation.

It is clear that for scheduling systems to become more relevant to actual applications they have to be more realistic in their assumptions, more flexible in their approach and better understood by schedulers. Clearly any OR model cannot reflect all the complexities of the real world and this is especially true in the highly dynamic world of production scheduling. Due to this fact OR scheduling systems are not the complete answer on their own, they need to be incorporated into a Decision Support System (DSS), whereby schedulers can access the recommendations of the algorithms and then modify the results to obtain a schedule which incorporates all the “soft” information that cannot be incorporated into the algorithm. The objective of an OR scheduling system is therefore not to produce a perfect recommendation of how the final schedule should look, but rather to provide a sufficiently good solution for the scheduler to work from.

2.6.2. Evaluation of E/T Scheduling Literature

Research on the E/T scheduling problem is not excluded from the criticism of irrelevance to the real world. Although the E/T scheduling problem closely models a JIT manufacturing environment, the assumptions placed on models used to solve it have rendered many of them ‘irrelevant’ to the problems faced by schedulers. This is not to say that these models cannot provide the building blocks for future research which can solve more realistic problems, however if this is to happen, any further research needs to be closely aligned with real world problems. If a change in focus is not made, there is a danger that the field will continue to remain theoretical. Some of the main areas for concern are:

1. A preoccupation with the Common Due Date Model. The common due date model has been a popular simplification due to its unique mathematical properties. It is a useful testing ground for new theories but is very limited in its applicability to the real world.
2. The relevance of more research into the CDD Unrestricted Due Date model. As pointed out by Baker and Scudder (1990), “These models are now well understood”.
3. Restrictions placed on Early/Tardy Penalties. In the literature to date many restrictions have been placed on early/tardy penalties giving rise to solution methods that can exploit the unique properties of these problems (Fry, Armstrong *et al.*, 1987; Yano and Kim, 1991). These restrictions, perhaps applicable in some industries, do not address the general problem faced by the majority of organisations.

4. The non-inclusion of idle time. Baker and Scudder (1990) note that this type of model is “inconsistent with the E/T criterion”. By not inserting idle time a major element of the E/T scheduling problem is removed.

The majority of literature to date has been involved with single machine scheduling, however in reality schedulers are more likely to be coordinating many different machines on the shop floor in their schedules. Goldratt and Cox (1986) point out that production scheduling problems can often be solved by concentrating on the bottleneck operations. Optimised Production Technology (OPT) is another technique that is based on dealing with bottlenecks. OPT “correctly recognised that bottlenecks control the rate of manufacturing throughput ... [therefore] the system attempts to optimise flow through the bottlenecks” (Gue, Heard, Maranka and McEneny, 1987). Scheduling bottleneck operations could therefore become the principle aim of a single machine E/T scheduling system.

Clearly the closer the E/T scheduling model is to reality, the more likely it is to provide a good starting schedule for the scheduler to work from. In this respect it is necessary to develop models and objective functions which closely resemble the scheduling problem being solved in the real world.

2.7. Conclusions

The E/T scheduling area is made up of many different classes of problems which are differentiated by the presence or absence of certain restrictions and assumptions.

From the literature it is clear that the unrestricted version of the CDD problem is well understood and needs little further research, however there is still room for the restricted version of the CDD problem to be developed further to aid the development of solution methods for the more general, distinct due date problem.

Relatively little work has been done on the distinct due date problem. The papers in this area have either dealt with very small problems or with special classes of the problem. Those that have dealt with special cases of the problem have either ignored the insertion of idle time or have imposed limitations on the earliness and tardiness penalties. It is the objective of this research to address this imbalance by considering techniques that could solve the general E/T problem, which considers the insertion of idle time and places no constraints on the early or tardy penalties, processing times or due dates. A particular focus of this research will be to consider solving large scheduling problems which involve up to 250 jobs.

No literature to date has been able to decompose the distinct due date problem into common due date sub-problems as suggested by Baker and Scudder (1990). This remains an area for future research.

The complexities of the job-shop are numerous, many of which are qualitative and therefore more difficult to incorporate into an OR scheduling model. To address this problem and the concerns raised by McKay *et al.* (1988), the goal of researchers in this area should be to provide high quality solutions which can then be incorporated into a DSS for schedulers to work from. Researchers should not assume that OR models will find the 'real' optimal solution to the scheduling problem due to these complexities.

3. Search Techniques Literature Review

3.1. Introduction

Heuristics are algorithms which do not “focus on finding an optimal solution, but instead try to find a ‘good’ solution within an acceptable amount of time” (Garey and Johnson, 1979, p 122). They are generally used on NP-hard problems which can not be solved efficiently through exact techniques, such as integer programming. Heuristics can be broadly categorised into two distinct groups, construction heuristics and improvement heuristics. Construction heuristics apply predefined “rules of thumb” to produce a single solution to the problem being solved. Improvement heuristics, on the other hand, start with an arbitrary, feasible solution and modify it repeatedly in an attempt to improve the overall solution. Search techniques form the major part of the latter group.

The purpose of a search technique is to find the best possible solution to the problem. Only two special types of searches can be guaranteed to find the optimal solution:

1. *Complete Enumeration* searches the entire solution space for the optimal solution. This technique is very inefficient and, as a result, can only be used on very small problems.
2. *Branch and Bound Enumeration* searches only those areas of the solution space where the optimal solution could be. For minimisation problems, upper bounds on the optimal solution for the problem are calculated so that partial solutions to the problem which have a cost higher than the upper bound are not developed any further, effectively eliminating areas of the search space. Often, however, branch and bound can only be used on small problems due to the exponential increase in the size of the solution space as the problem gets larger.

To solve larger problems other techniques have been developed which, although they are not guaranteed to give the optimal solution, can provide acceptable solutions within a reasonable amount of time. The three search techniques examined in this thesis; simulated annealing, genetic algorithms and tabu search; are used extensively in the literature to solve a wide variety of NP-hard problems, including scheduling problems. These three techniques have been chosen as they are claimed to be ‘intelligent’ search techniques which ‘learn’ from the environment which they are in, and apply what is learnt to help find better solutions. These claims of intelligence, as with all claims about artificial intelligence, are open to debate as to whether they are true intelligence or not. Whether these techniques are truly intelligent is not the issue, what is more important is that these techniques have been used successfully in solving many different NP-hard problems and could therefore provide a method of solving the E/T scheduling problem.

One of the unique features of improvement heuristics is that they define how a modification can be carried out on a given solution to obtain a new solution. This new solution is called

a *neighbour* to the current solution. The rules that define what new solutions can be generated from the current solution constitutes the *neighbourhood scheme*. A complete discussion on neighbourhood schemes is carried out in Section 3.2.

The simplest improvement heuristic is called a ‘gradient descent’ method. This heuristic starts from a given point and explores all the neighbours defined by the given neighbourhood scheme. When the objective is to minimise cost, check if there exists a neighbour whose cost is less than the cost of the present solution. If there is then move to the neighbour which has the least cost and repeat the process again. If there are no neighbours with a cost less than the present solution then the current solution becomes the final solution and the search terminates. The main problem with this type of heuristic is that it will find local minima but not necessarily the global minima. A Local minima is a solution which has no neighbour with a lower cost. Consider Figure 3-1.

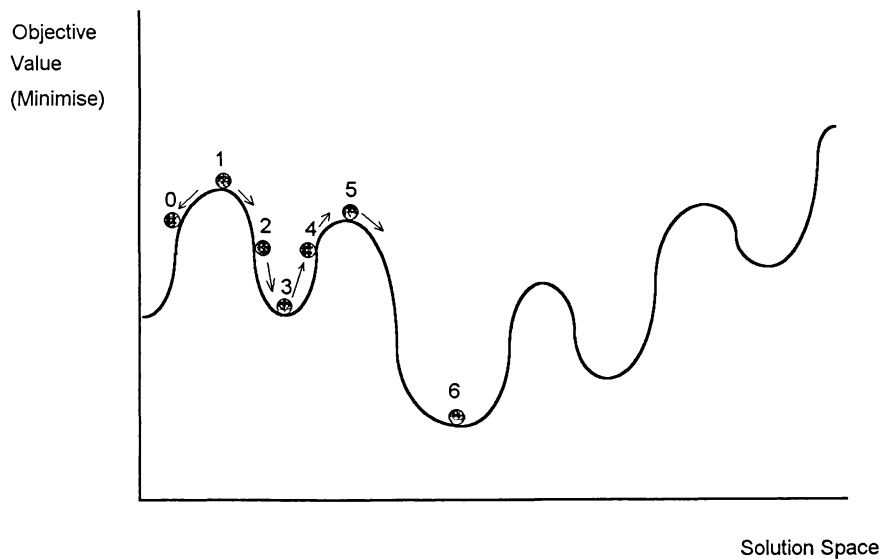


Figure 3-1 Local Minima in a Solution Space

The graph represents the solution space surface and the ball at point 1 represents where the starting point for the search. The search may take the ball down to position 2 (alternatively it could have taken the ball down to position 0) and then to position 3. At this point there is no neighbour to the solution at position 3 which has a lower objective value. This point is a local minima. The global minima or optimal value, at position 6, can only be achieved by allowing the search to move to a solution with a worse objective value, up to position 4 and ultimately to position 5, and then down to position 6.

The objective of search techniques is to avoid becoming stuck in local minima by allowing moves to worse solutions, under certain conditions, so that better subsequent solutions can be found.

Section 3.2. outlines several different neighbourhood schemes that can be used for scheduling problems. Section 3.3. introduces simulated annealing, Section 3.4. introduces genetic algorithms while Section 3.5. introduces tabu search. Each of these sections explains the particular search technique and some of the common extensions to them. Literature which has used these techniques for scheduling applications are also discussed, and conclusions are drawn from this discussion on the use of the particular technique in solving the E/T scheduling problem. The final section, Section 3.6., summarises the major conclusions made in this Chapter.

3.2. Neighbourhood Schemes

3.2.1. Sequencing Neighbourhood Schemes

The standard neighbourhood schemes used in the scheduling literature are derived from neighbourhood schemes used to solve the “travelling salesman” problem. These neighbourhood schemes determine the order in which jobs are processed, which for many types of scheduling problems is all that is required. The most common neighbourhood schemes found in the literature are adjacent pairwise interchange, swap, and insert.

The ‘Adjacent Pairwise Interchange’ (API) neighbourhood scheme allows for any two jobs which are sequenced next to each other to be swapped. This neighbourhood scheme has $(n-1)$ neighbours for any one solution. An example of an API neighbour is shown in Figure 3-2.

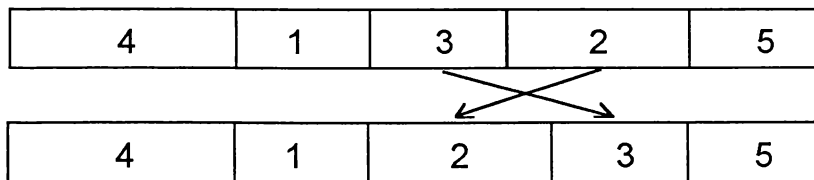


Figure 3-2 An Example of a Adjacent Pairwise Interchange Neighbour

The ‘Swap’ or ‘All Pairwise Interchange’ neighbourhood scheme allows for any two jobs from anywhere within the schedule to be swapped with each other. This neighbourhood scheme includes all the neighbours in the API neighbourhood scheme along with many others. The total number of swap neighbours to a solution is therefore $n(n-1)/2$. An example of a swap neighbour to a solution is shown in Figure 3-3.

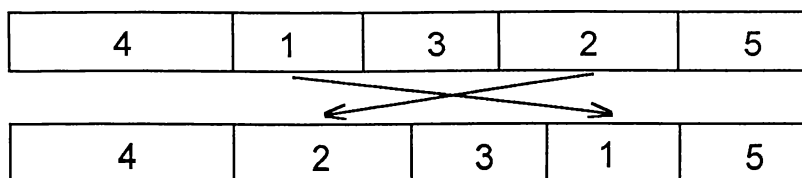


Figure 3-3 An Example of a Swap Neighbour

The ‘Insert’ neighbourhood scheme moves one job in the schedule and places it before or after another job in the schedule depending on the job’s position. If the job is moved forward in the schedule, then the job will be placed before the other job selected, whereas if the job is moved back in the schedule then the job will be placed after the job selected. Jobs cannot be placed before or after themselves as this does not change the sequence of jobs. This neighbourhood scheme also produces $n(n-1)/2$ neighbours to a given solution. An example of inserting job 2 in front of job 1 is shown in Figure 3-4.

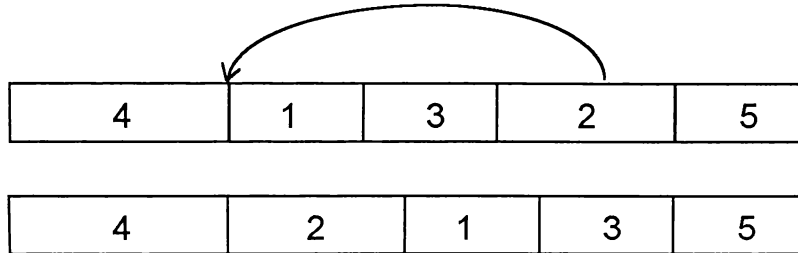


Figure 3-4 An Example of an Insert Neighbour

From experiments with scheduling problems using these different neighbourhood schemes, it was found that the swap neighbourhood scheme consistently gave better results than the API neighbourhood scheme (Tsubakitani and Evans, 1992), while the insert neighbourhood scheme consistently produced better results than the swap neighbourhood scheme (Laguna, Barnes and Glover, 1991). Neighbourhood schemes that combined both the swap and insert neighbourhoods into a single neighbourhood were also found to produce superior results to those that had only swap or insert neighbourhood schemes (Laguna, Barnes and Glover, 1991). From this it could be concluded that results improve as the neighbourhood scheme becomes more diverse, as solutions that are difficult to reach under one scheme will be easy to reach under another and vice versa (Barnes and Laguna, 1991a).

3.2.2. Problem Space/Heuristic Space Neighbourhood Schemes

Storer, Wu and Vaccari (1992) introduced the concept of using other types of search spaces, rather than the sequence of jobs solution space, to solve scheduling problems. Storer *et al.* (1992) developed two different types of solution spaces derived from the notion that a heuristic maps a given set of problem data to a specific solution in the solution space. Figure 3-5 illustrates this transformation.

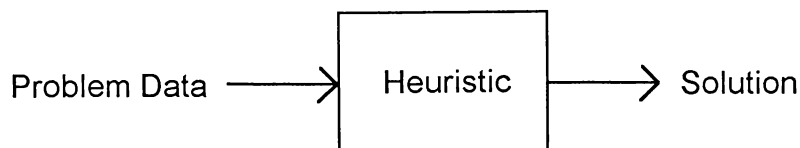


Figure 3-5 Mapping Proposed by Storer *et al.* (1992)

Storer *et al.* (1992) recognised that by modifying either one of the two “input” elements in this mapping, a different solution may be produced. Instead of having neighbourhoods based on modifying the solution, which is the standard approach, neighbours could now be generated by modifying either the problem data, therefore searching what has been termed

“problem space”, or the way in which the heuristic transforms the data, searching what has been termed “heuristic space”.

A neighbourhood which uses problem space defines its neighbouring solutions by changing a specific element of the problem data in some way by a given “delta” value. The modified problem is then solved by using a predetermined heuristic to generate a sequence which is designated as being the neighbouring solution’s sequence. The original problem is then reevaluated using this new sequence to determine the value of this neighbouring solution.

The way a heuristic space neighbourhood is implemented can vary. One method proposed by Storer *et al.* (1992) was to use a weighted sum of various dispatching heuristics to give a new dispatching rule. Changing the weights of the dispatching heuristics by a given delta value will generate different neighbouring sequences. Alternatively the schedule could be split into different sections of jobs, or “windows”. Each window would then be solved by a given heuristic from a suite of heuristics.

Storer *et al.* (1992) tested these solution spaces on job shop scheduling problems with the objective of minimising makespan. The problem space method was tested using a partial gradient descent method whereby only a predefined number of randomly selected neighbours were evaluated, rather than the entire neighbourhood, before a decision was made about a move. Recall that for gradient descent searches the current solution is only updated to the best neighbour if it has a cost which is lower than the current cost. If there are no neighbours with a lower cost then the search terminates. The heuristic space was implemented using a window based approach to the solution space. One of six different dispatching rules was able to be applied to each of these windows. The experiments were carried out using different numbers of windows within the problem. The search technique for the heuristic space used a simple hill climbing heuristic which selected a random neighbour, evaluated it and moved to it if it has a cost lower than or equal to the cost of the current solution. A combined approach, which takes the best solution from the heuristic space search and uses it as the starting point for the problem space search, was also tested.

Computational experiments on problems ranging in size from 5-15 machines and 10-50 jobs found that the problem solution space performed extremely well, producing results that were half the deviation from optimal when compared to the shifting bottleneck procedure of Adams, Balas and Zawack (1988). The combined approach produced higher quality results at the expense of extra computational time, whereas the heuristic space techniques performed poorly when compared to the problem space techniques. No results showing computational times were given hence a time/quality analysis could not be performed on the different heuristic procedures.

We now consider three different search techniques which can utilise these different neighbourhood scheme: Simulated Annealing, Genetic Algorithms and Tabu Search.

3.3. Simulated Annealing

3.3.1. Introduction

Simulated Annealing (SA) was first developed by Kirkpatrick, Gelatt and Vecchi (1983) from a simulation model of the annealing of metals developed by Metropolis, Rosenbluth, Rosenbluth, Teller, and Teller (1953). This model proposed a mathematical way of modelling the changes in the properties of a metal as it reaches a thermal equilibrium, that is the stable composition of the metal at a given temperature. The model was based on simulating the movement of atoms within the metal whereby small changes in the positions of these atoms changes the overall “energy” level of the metal. Metropolis *et al.* (1953) proposed that if changes produced a lower energy level then the changes were immediately accepted and this new point was used as the starting point for the next iteration. However if the changes produced a higher energy level then the change was accepted with a probability of:

$$e^{-\frac{\delta}{k_B T}}$$

Where

δ is the change in energy level of the metal,

T is the temperature of the system the metal is being exposed to,

k_B is a scaling constant for the system, called the Boltzmann constant.

Kirkpatrick *et al.* (1983) suggested that an analogy existed between the cooling of metals and the minimisation of an objective function in combinatorial optimisation problems. They applied the method of Metropolis *et al.* (1953), used for simulating the movement to an equilibrium in metals, and applied it to the behaviour of a search technique employed to identify the global minima of an optimisation problem. The energy level of the metal is represented by the objective function to be minimised and, rather than stabilising at a minimum energy level, the system stabilises at a minimum objective value. The method will not get trapped in local minima as long as the temperature of the system, T , is greater than zero. However the lower the temperature, the harder it is for the method to escape from a local minima.

3.3.2. Overview of the Method

Pseudo-code for the basic SA algorithm, proposed by Eglese (1990) is shown in Figure 3-6

```

Select an initial state,  $i \in S$ ;
Select an initial temperature,  $T > 0$ ;
Set temperature change counter  $t = 0$ ;
Repeat
    Set repetition counter  $n = 0$ ;
    Repeat
        Generate state  $j$ , a neighbour of  $i$ ;
        Calculate  $\delta = f(j) - f(i)$ ;
        if  $\delta < 0$  then  $i := j$ 
        else if  $\text{random}(0, 1) < \exp(-\delta/T)$  then  $i := j$ 
         $n := n + 1$ ;
    until  $n = N(t)$ ;
     $t := t + 1$ ;
     $T := T(t)$ ;
until stopping criteria true.

```

Figure 3-6 Basic Simulated Annealing Algorithm (Eglese, 1990)

This algorithm, however, is not all that is required to implement a SA algorithm. Johnson, Aragon, McGeoch and Schevon (1989) point out that SA is “more an *approach* than a specific algorithm” similar to other general, non-specific approaches such as dynamic programming, genetic algorithms and tabu search.

Johnson *et al.* (1989) describe four problem-specific issues that must be resolved before SA can be implemented. These are to:

1. Explicitly define what a solution is,
2. Define what a neighbour to a solution is,
3. Define the cost function to evaluate the solution, and
4. Define an initial starting solution for the algorithm.

Each of these issues are important to the overall success of the SA algorithm. The neighbourhood scheme is especially important as it determines the overall efficiency of the algorithm by defining the topography of the solution space (Eglese, 1990). If the topography of the solution space is ‘bumpy’, this indicates that there are many local minima that must be overcome during the search, which will slow down the search. Typically, a bumpy topography will occur when there are a small number of neighbours for each solution. However if the neighbourhood scheme is too large, the probability of the algorithm finding the neighbours which will reduce the overall cost is substantially reduced, again reducing the effectiveness of the algorithm.

van Laarhoven and Aarts (1987) identified four generic issues that need to be defined before a SA algorithm can be implemented. These are to:

1. Determine the initial values of the Boltzmann Constant, k_B , and the initial temperature T which form the “control parameter” $k_B T$.

2. Determine the number iterations that are spent at each temperature.
3. Determine the rule used for changing the temperature.
4. Determine the final temperature when the algorithm will stop.

The answers to these four issues form, what is known as the SA algorithm's "cooling schedule", as this determines every aspect of how the temperature of the system will be used to control the search. These parameters must therefore be "tuned" to each individual application to obtain the best possible results.

Kirkpatrick *et al.* (1983) propose that the initial control parameter be determined by using the algorithm in Figure 3-7.

1. Choose a large value for the initial control parameter.
2. Perform a predefined number of iterations.
3. If the ratio of the number of accepted iterations to the total number of iterations is less than a particular acceptance ratio χ_0 (proposed to be 0.8 by Kirkpatrick *et al.* 1983) then double the current value of the control parameter and go to step 1.

Figure 3-7 Algorithm to Determine the Initial Control Parameter
(Kirkpatrick *et al.* 1983)

The length of time spent at each temperature allows the system to reach an equilibrium point for that temperature, called a "quasi-equilibrium" by van Laarhoven and Aarts (1987).

There are many different methods for decreasing the temperature explained in the literature. The most common method, according to van Laarhoven and Aarts (1987), is to use the rule:

$$t_{k+1} = \delta t_k,$$

where δ is a constant, typically between 0.8 and 0.99 (Eglese, 1990).

Lundy and Mees (1986) reduced the number of parameters required in the cooling schedule by specifying that the temperature should be decreased after each iteration. They proved that if the temperature is modified according to a particular rule, then the annealing routine would still be guaranteed to find the optimal solution given the correct parameters. The rule they developed is:

$$t_{k+1} = \frac{t_k}{(1 + \gamma t_k)},$$

where γ is a constant which is much less than $1/U$, where U is the maximum depth of the largest local minima in the solution space.

van Laarhoven and Aarts (1987) outline three different methods for determining the final temperature at which the algorithm will stop. The alternative methods stop the algorithm if:

1. A fixed number of temperatures have been used,
2. The solutions obtained from the last two consecutive temperatures are identical, and
3. The number of moves accepted at the temperature is smaller than a given value.

Lundy and Mees (1986), however, propose that the final value of the control parameter should depend on the amount of acceptable error ϵ , the probability of error ϕ and the size of the solution space X according to the formula:

$$k_B T_f \leq \frac{\epsilon}{(\log(X-1) - \log \phi)}.$$

3.3.3. Extensions to the Standard Method

Many extensions to the base SA algorithm have been proposed in the literature to either increase the quality of the solutions obtained or to speed up the algorithm. These two elements interact as the faster the algorithm, the higher the quality of the solution obtained within a same amount of time.

Some of the extensions proposed in the literature are:

1. Storing the best solution found so far.

This is a simple extension that is incorporated into almost every version of SA in the literature. This ensures that if the SA algorithm moves away from a good solution and into other areas of the solution space then the good solution obtained previously is not lost.

2. Combining SA with other methods.

SA can be combined with other methods by either having another method provide the initial starting point for the SA or by having SA provide the initial solution for another type of search technique.

3. Choosing moves with consideration of previous iterations.

When the temperature of the system becomes low, the majority of solutions generated by the SA algorithm are rejected. Johnson *et al.* (1989) investigated a method of recording all the moves that had been tried previously so that they would not be retried if selected again by the random selection procedure. They found that the quality of the solutions obtained were increased by this extension.

4. Approximating the exponential function.

Calculating the value of the exponential function, $e^{-\delta/T}$, takes a large amount of computational time. Johnson *et al.* (1989) tested approximations to the exponential function, one of which was the function $1-\delta/T$. It was found that this simpler function took 1/25th of the time it took to compute the exponential function, providing a 33% increase in speed with no significant decrease in solution quality.

5. Limiting conditions on the number of iterations spent at a particular temperature.

One condition tried in the literature was to spend a fixed number of iterations at a temperature or until a fixed number of moves were accepted. Johnson *et al.* (1989),

however, found that this modification made little difference to either the solution quality or the run time.

6. Parallel processor versions.

The SA algorithm can be run on parallel processors by either splitting up the calculations for each trial between processors, or by having processors evaluate multiple solutions in parallel. Abramson (1992) found that parallel processing would speed up the algorithm by around 50%. Abramson (1992) also describes a specially designed computer, called an *annealing machine*, which is designed around the control structures found in simulated annealing. The result was a massive increase in speed. With a special annealing machine board installed in an IBM AT, it was found to be 33 times faster than a Cray computer (recognised at the time to be one of the most powerful computers in the world).

7. Nested Simulated Annealing.

John and Whitaker (1992) proposed a new method of overcoming the slow convergence of the standard SA algorithm, called Nested Simulated Annealing. The technique starts the algorithm off at a high temperature. After a specified number of iterations at this temperature, the best solution found so far is stored so that the algorithm can return to this level. The temperature is reduced by a factor, F , which is between 4 and 10. The process continues at the new temperature. If the best solution found at this level is better than the current best solution the temperature is reduced by the factor F . This continues until the temperature becomes minimal or no further improvements are made. Once a local minima has been reached, then the routine is restored back to its original level and continues its search at a lower base temperature. In their experiments into row/column designs for crop studies, John and Whitaker (1992) found that nested SA produced near optimal results efficiently, but marginal improvements took a lot longer than with standard SA.

3.3.4. Simulated Annealing Machine Scheduling Applications

Due to the overall simplicity of SA, it has been used to solve many NP-hard problems including some machine scheduling problems.

The flow shop problem requires n jobs to be processed on m machines in the same sequence. Both Osman and Potts (1989) and Ogbu and Smith (1990) have investigated the use of SA to solve the flow shop scheduling problem with the objective of minimising makespan.

Osman and Potts (1989) compare two different construction heuristics, which also have a 'gradient descent' phase implemented once completed, with SA. They experimented with both swap and insert neighbourhood schemes (see Section 3.2. for a complete discussion on neighbourhood schemes) along with the selection of jobs in either a given or a random

order. They found that randomly selected jobs using the insert neighbourhood scheme produced the best results. Ogbu and Smith (1990) compared the same two neighbourhood schemes and also found that the insert neighbourhood scheme was superior to the swap neighbourhood scheme.

The initial starting point used by Osman and Potts (1989) was a first-come-first-served algorithm, while Ogbu and Smith (1990) used two different construction heuristics to give good quality starting points. Their computational experiments showed, however, that there was no statistical difference between the results obtained from the SA algorithms which used heuristics to generate an initial starting solution and the results obtained from the SA algorithms which had random starting points.

The implementation of the simulated annealing algorithm, however, was different between these two papers. Osman and Potts (1989) adopted the Lundy and Mees (1986) form of SA which decreases the temperature after each iteration. The initial temperature was proportional to the average processing time for a job on a machine, while the final temperature was given as being 1. The constant γ is calculated so that a given number of iterations will be executed.

Ogbu and Smith (1990) use a SA which does not involve the Metropolis function, but instead uses an acceptance criteria which does not consider the proportional change in the cost function. The acceptance probability function, $AP(k)$, is therefore defined as:

$$AP_{ij}(k) = \begin{cases} AP(1)(pfac)^{k-1} & \text{if } C_j > C_i \\ 1 & \text{otherwise} \end{cases}$$

Ogbu and Smith (1990) defined $pfac$ as being like the temperature of the system in iteration k , and must be less than 1. The initial acceptance probability, $AP(1)$, was set at 0.2. A comparison of this implementation of SA with standard SA implementation found that both techniques found good quality solutions, with under 2% relative error for both techniques. However the modified SA approach was found to be more efficient, as it took less computational time.

Vakharia and Chang (1990) applied SA to the flow shop scheduling for a manufacturing cell, where jobs are grouped into families and the objective is to minimise the overall makespan. They use a neighbourhood scheme which uses adjacent pairwise interchange of jobs and/or families of jobs and starts from a random sequence of jobs. A standard SA algorithm is used with a constant decrement in the temperature, which is calculated from the number of temperatures that are to be used which, in the experiment, is set at 25. The algorithm incorporates a further probability factor which determines whether jobs or families are to be considered.

Vakharia and Chang (1990) conclude from their experiment that SA provides “good results as [*sic*] compared to existing heuristic methods”. They state that the solutions obtained for larger problems were better, with up to 50% less deviation from the lower bound, and required less computational effort, nearly 75 times faster, than using integer programming methods which were restricted in computational time. The SA approach also produced far better solutions (up to 33% closer to the lower bound) than two standard heuristics for this problem, one of which was developed by Campbell, Dudek and Smith the other by Nawaz, Ensore and Ham. This was, however, at the expense of computational time which was almost 100 times slower. In contrast to Ogbu and Smith (1990), Vakharia and Chang (1990) found that using a heuristic to find an initial starting solution for the SA did increase the overall quality of the final solution without any increase in computational time.

van Laarhoven, Aarts and Lenstra (1992) examine the use of SA to solve the job shop scheduling problem with the objective of minimising the makespan. In this problem we have n jobs and m machines, with each job being comprised of one or more operations which are to be carried out on a specific machine. This process can be represented by a graph, called a disjunctive graph, which contains two types of arcs, one type representing the jobs, the other type, called disjunctive arcs, representing the machines on which job operations will be performed. Normal arcs can only flow in one direction and can not be changed, while disjunctive arcs can flow in either direction and can change as appropriate. Consider Figure 3-8. From this diagram each of the arrowed paths indicate one of 3 jobs, while the dashed lines between the operations are the disjunctive arcs which represent the order in which each operation is performed on the machines by the direction they are assigned. From Figure 3-8 we can see that jobs 1, 5 and 9 are all processed on one machine, 2, 8 and 4 on another and 3, 6 and 7 on a third machine.

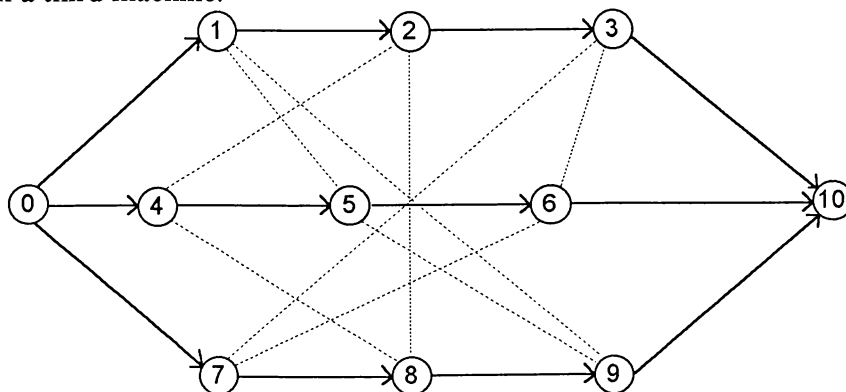


Figure 3-8. A Disjunctive Graph of the Job Shop Problem
(van Laarhoven *et al.* 1992)

By using this formulation of the problem, the longest path from the starting operation, 0, to the finishing operation, 10, is the critical path which is to be minimised. van Laarhoven *et al.* (1992) use this feature of the problem to reduce the size of the neighbourhood to be considered, eliminating many nonbeneficial moves immediately, as only modifications to the critical path need to be considered since only changes to these arcs will have any effect on

the overall makespan. A neighbour is defined by van Laarhoven *et al.* (1992) as reversing a disjunctive arc on the critical path in such a way as to perform an adjacent pairwise interchange between two adjacent operations scheduled on the same machine.

van Laarhoven *et al.* (1992) use the standard SA approach, using an initial temperature of 0.95, a final temperature of 10^{-6} and the number of iterations spent at a particular temperature is set to the longest path within the graph. The rule to decrease the temperature is defined as:

$$c_{k+1} = \frac{c_k}{1 + [c_k \ln(1 + \delta) / 3\sigma_k]}$$

where c_k is the control parameter for the k th temperature used, σ_k is the standard deviation of cost values found using the control parameter c_k and δ is a parameter which controls how quickly the temperature will decrease, which was tested with values between 10^{-1} through to 10^{-4} . The SA approach to the problem was then compared to the shifting bottleneck procedure of Adams, Balas and Zawack (1988) which is designed specifically for this problem. It was found that both schemes took approximately the same amount of computational time when δ was set at 0.1, however the average results obtained from the standard heuristic were considerably better than the SA approach, however when δ was set at 0.01, the average results were better for SA than for the standard heuristic but at the expense of longer processing times.

Matsuo, Suh and Sullivan (1989) used SA to solve the single machine weighted tardiness problem. Their experiments tested many different modifications to the standard SA approach and measured their impact on solution quality for problems involving 15, 50 and 100 jobs. All experiments used the API neighbourhood scheme. Four dispatching heuristics; shortest weighted processing time, earliest due date, cost over time and apparent urgency; were used to find an initial starting sequence. Tests found that the better the initial starting point, the better the final solution obtained by the SA algorithm. A second experiment investigated the use of several different acceptance probability functions which do not depend on the change in the objective function but solely on the iteration number. These functions included linear, logarithmic, linear piecewise concave, linear piecewise convex and exponential functions. From this experiment it was found that the linear and piecewise functions provided the best results and were computationally more efficient. When the linear form was compared with the standard Metropolis function, it was found that the SA with the Metropolis function solved just 3% more problems optimally than the SA with the linear function however this was at the expense of computational time. Matsuo *et al.* (1989) concluded from this that the performance of these two function forms is “nearly as good” as each other and hence they adopted the linear form. Another modification that was trialed was to search the neighbourhood in an order which places solutions which are known to have a higher cost, from previous tests, last rather than

randomly testing different neighbours. The results showed that the ordered approach was superior to the random approach, with the ordered approach finding 9% more optimal solutions from the set of test problems. A comparison between the modified SA and a standard SA was carried out and, although the solution quality of each was approximately the same, the standard SA took between 10 and 13 times longer.

Potts and van Wassenhove (1991) also used SA to find solutions to the single machine weighted tardiness problem. In their paper they compare several different heuristics, including simple dispatching rules as well as two different descent methods, to the SA method. Potts and van Wassenhove (1991) used the swap neighbourhood scheme and used an initial starting solution based on the earliest due date (EDD) dispatching rule if the problem has a tardiness factor less than 0.2, or an “apparent urgency” rule, developed by Morton, Rachamadugu and Vepsalainen, otherwise. In their experiments, they found that the starting point for the SA was important to the overall outcome of the SA algorithm.

Potts and van Wassenhove (1991) implement two different SA algorithms. The first follows the process outlined by Lundy and Mees (1986). They start with an initial temperature of 1, and end with a temperature of 0.334, while the constant γ is calculated to allow a certain number of iterations, which is set at 10,000 for the 50 job problems and 20,000 for 100 job problems. The second SA algorithm is similar, except that when a move that decreases the objective function is accepted a gradient descent algorithm is used to find the local minimum. The SA is then resumed from the local minimum found.

From computational experiments Potts and van Wassenhove (1991) found that in general SA was not able to significantly improve on the solutions found by descent methods unless a large number of iterations were allowed. Their modified version of SA did, however, perform significantly better than the descent methods for the weighted tardiness problem with improvements of up to 45% on the solution quality when a large number of iterations were allowed.

Mittenthal, Raghavachari and Rana (1993) use SA to solve the E/T scheduling problem with common early and tardy weights for all jobs and a common due date. The objectives used on this problem were to minimise the total variance in the completion times, minimise the total covariance in completion times, and to minimise a combination of the variance and the mean completion time. Due to the “V-Shaped” property of this problem (refer to Section 2.4 for a full discussion of this property) the neighbourhood can be reduced significantly for this problem. To exploit this feature of the problem they define two different neighbourhood schemes. The first is similar to the insert scheme whereby a job is moved from the early or late side of the schedule to the late or early side of the schedule respectively. The second scheme is similar to the swap neighbourhood scheme, whereby one late job, and one early

job are swapped. The SA algorithm was implemented using a starting temperature of 100, stopping at a temperature of 2, with a reduction factor of 0.8. A hybrid SA algorithm was also tested, which uses a gradient decent method to find a starting point for the SA algorithm. This is implemented by performing a gradient descent using the insert neighbourhood scheme until a minimum is reached, then performing a swap gradient descent until a minimum is reached. The SA algorithm is then started using this minimum point as its starting point and proceeds using a starting temperature of 20, ending temperature of 1 and a reduction factor of 0.8. From the computational experiments it was found that the hybrid SA algorithm had the best performance of all the heuristics including other SA procedures that started at various starting points, including LPT, SPT and a V-Shaped schedule based on all odd-numbered jobs being scheduled early and all even-numbered jobs being late.

3.3.5. Conclusions

From the literature on SA scheduling applications, it appears that the results are mixed. In some cases the reported results appear to contradict each other. Some claim that starting points are important (Potts and van Wassenhove, 1991; Vakharia and Chang, 1990; Matsuo *et al.*, 1989) while others claimed that they found no statistical significance between starting points (Ogbu and Smith, 1990). Explanations for this may be in the different types of scheduling problems, in the different ways SA was implemented, or simply in the amount of time that the SA algorithm was allowed to run. Another conflict appears to be whether SA is any more effective than simple gradient descent methods. Most of literature claims that it is, whereas Potts and van Wassenhove (1991) found that SA did not perform any better than descent methods, unless run for a considerable amount of time. Again these findings may be a feature of the type of problem that is being solved and the topography of the solution space being searched.

The implementation of SA algorithms is varied. Two techniques implement a gradient descent phase into the algorithm to improve solution quality, which is contrary to the analogy behind SA. Both van Laarhoven *et al.* (1992) and Mittenthal *et al.* (1993) exploit the nature of the problem that they were solving in order to reduce the size of the neighbourhood that needed to be searched and therefore increasing the efficiency of the SA algorithm. No comparisons have been carried out between implementing a SA algorithm based on a general neighbourhood scheme as opposed to a custom built neighbourhood scheme specifically designed for a particular problem. This, therefore, remains an area for further research.

3.4. Genetic Algorithms

3.4.1. Introduction

Genetic Algorithms (GAs) were developed in 1975 by Holland (Holland, 1975). Just as Simulated Annealing was inspired by the annealing process of metals, GAs were inspired by evolutionary theory, where each species makes adaptations to make it more suitable to its environment. The weaker strains of the species die out, while the stronger, better strains flourish.

The nucleus of a biological cell controls the characteristics and the growth of the cell. The information on the characteristics of the cell is contained in the *chromosomes* which make up the nucleus. The chromosome itself is made up of many different *genes*. A single gene can control many different features with several different genes making up the information about a single characteristic, such as eye colour. A slight change to a gene can have a major effect on many of the characteristics of the biological system of which it is a part.

Holland (1975) examined how to simplify these biological cells and model them mathematically. He suggested that complex chromosome structures could be represented by lists of 0's and 1's, called *bit strings*, which would, in turn, represent different characteristics. These bit strings would be similar to the chromosomes found in natural systems. He suggested three mechanisms for these bit strings to be improved: *reproduction*, *mutation* and *crossover*. The GA therefore simulates the processes in a real biological system where chromosomes are mated, mutated, and survival of the fittest is the key element of removing poor chromosomes. From a population of different chromosomes, parent chromosomes are selected and mated using a crossover transformation. The child chromosomes then form part of the next generation in the population. The weaker members die off, as they are replaced by the stronger chromosomes.

In the mathematical model of these biological systems, each solution to the problem being solved is represented by a different chromosome. By creating a population of chromosomes, or solutions, and applying a GA, the final population will contain more and more chromosomes, or solutions, which are “stronger”, effectively bringing the solutions closer to the optimal solution, where the “fitness” of the chromosome is determined by the objective function of the problem being solved.

GAs differ greatly from other search techniques in that they do not work from a single point in the solution space, but from multiple points. This enables GAs to avoid becoming trapped in local minima which is a problem with many other search techniques (Biegel and Davis, 1990). The focus of GAs is therefore on finding the global minima for a problem rather than finding different local minima.

3.4.2. Overview of the Method

An outline of the GA which could be used for optimisation problems is shown in Figure 3-9.

1. Create an initial population of chromosomes.
2. Evaluate the chromosomes.
3. Based on the evaluation of the chromosomes in the population, generate new chromosomes by mutating and recombining the parent chromosomes.
4. Delete some of the population to make room for new members.
5. Evaluate newly generated chromosomes and insert the best into the population.
6. If time is up stop, otherwise go back to step 3.

Figure 3-9. The Basic Genetic Algorithm (Lawton, 1992)

Davis and Steenstrup (1987) identified five different elements that all GAs must have. These are

1. A *chromosomal representation* of solutions to the problem.
2. A way to create an *initial population* of solutions.
3. An *evaluation function* that plays the role of the environment, rating solutions in terms of their fitness and therefore their probability of being selected for mating for the next generation..
4. *Genetic operators* that alter the composition of the children during reproduction.
5. *Parameter values* for variables such as population size and the probability of applying each genetic operator.

Much of the research on GAs has been concentrated on finding better methods to address each of these five elements (Davis and Steenstrup, 1987).

The most common form of chromosomal representation is the zero-one bit string introduced by Holland (1975). Under this scheme some means of encoding the original problem into the bit string is required. The advantage with bit string chromosomes is that many of the standard GA operators will work on these chromosomes without modification, enabling a standard GA to be used for many different problems. Davis (1991) proposed the use of strings of real numbers as an alternative, and found that this provided better results than bit strings for some applications along with having the advantage of being more understandable to the end user. For optimisation problems that involve determining a sequence of jobs, such as machine scheduling problems, the most common scheme makes the chromosome represent the sequence of jobs. To use this chromosome representation, special genetic operators must be developed so that no job is repeated within chromosome.

When considering how to generate the initial population, it is necessary to heed Grefenstette (1987) who warns “it is important to ensure that a great deal of variety exists in the initial population to maximise the number of opportunities for the GA to create the greatest number of combinations”. In his research on initial populations, Grefenstette (1987) found

that heuristics to create the initial population enabled good solutions to be found quickly, however when the GA was allowed to run long periods of time, the random schemes were better due to their more diverse populations.

Davis and Steenstrup (1987) highlight the importance of the evaluation function in the GA implementation as it is the evaluation function that will determine which types of chromosomes are encouraged and which are discouraged in the next generation. Typically the evaluation function is the objective function of the problem but may also incorporate some modifications, such as the conversion of the problem from a minimisation problem to a maximisation problem - which is the natural form for most GAs. If the evaluation function produces similar results for all chromosomes in the population, then overall progress of the GA to the global optima will be very slow. However, if too much emphasis is placed on improvements, then the diversity of the population will be lost. To overcome this problem the evaluation function is 'normalised' or 'scaled' so that this speed/diversity tradeoff can be controlled. The method used to perform this normalisation can therefore play a large part in the overall success of the GA.

Genetic operators can be classified into Reproduction, Crossover and Mutation operators. Reproduction is the process whereby individual chromosomes are copied with a probability related to the objective function. Crossover combines two chromosomes, called parents, to produce two new chromosomes, called children. Mutation changes one of the genes in the chromosome to produce a new chromosome. The way in which the crossover and mutation is carried out will depend on the chromosome representation adopted. For example, a bit string chromosome can be mutated by simply changing one of the 0's to a 1 or vice versa, whereas a sequence chromosome needs a mutation operator similar to the swap or insert function used in other search techniques. A full discussion on genetic operators and their variations is presented in Section 3.4.3.

The main parameters that need to be determined before implementing a GA are:

1. the probability of applying each different genetic operator,
2. the size of the population,
3. the types of operators that will be used, and
4. the stopping criteria, usually given as a specific number of iterations.

Goldberg (1989) reported that research into the appropriate values of these parameters found that GAs performed well with high crossover rates, low mutation rates which were inversely proportional to the population size, and moderate population sizes. Booker (1987) suggested that these parameters should be varied during the run to increase performance by initially setting the crossover rate at a low level and progressively increasing

it as the algorithm progresses. In his trials with this technique Booker (1987) found that this modification did improve the performance of the GA.

Wiggins (1992) stated that from his experience “only two relevant parameters make a strong difference in how quickly a maximum is found.” The first is the population size, for which he recommends a population of 10, and the second is that mutation should only be used to prevent member duplication. From his experience the other parameters have little effect on efficiency of the search. These claims, however, do not have quantitative evidence to support them. Syswerda (1991), however, indicated that a population size of 30 chromosomes provided good results, however he also found that the most effective population size is dependent on the problem being solved, the chromosomal representation being used and the operators manipulating the representation. It is clear that there is no one set of ‘best parameters’ that can be used for every type of GA application, and therefore parameter tuning is required to achieve the best results.

We now move from considering the general parameters of the GAs and now turn our attention to how the different genetic operators can be implemented.

3.4.3. Genetic Operators

One of the important elements of the GA identified by Davis and Steenstrup (1987) was deciding which genetic operators to use. This section outlines how each of the three genetic operators; reproduction, crossover and mutation; can be implemented.

The reproduction operator implements the “survival-of-the-fittest” mechanism into the GA. The main issue for implementing a reproduction operator is the way in which chromosomes for the next generation are selected.

One of the more common techniques for selecting the chromosomes to be involved in the reproduction process is called the “roulette wheel parent selection” which is outlined in Figure 3-10. With this scheme the probability of selecting a particular chromosome is directly proportional to the fitness of the chromosome, relative to all other chromosomes. Note that the selection method outlined in Figure 3-10 is for maximisation functions only, therefore for minimisation problems the evaluation function must subtract the actual cost from a sufficiently large constant. Determining this constant is part of the normalisation process.

1. Sum the fitness of all the population members; call the result total fitness.
2. Generate n , a random number between 0 and total fitness.
3. Return the first population member whose fitness added to the fitness of all the proceeding population members, is greater to or equal to n .

Figure 3-10. "Roulette Wheel Parent Selection" Procedure outlined in Davis (1991)

Some modifications to the reproduction process have been proposed to enhance the performance of the GA. These include:

1. Always copying the best chromosome from the current population into the next generation so that the chance of the best chromosomes not reaching the next generation is eliminated. This modification is called *Elitism*.
2. Instead of replacing the entire population, only a few of the chromosomes are replaced at any one time. This modification is called *Steady-State Reproduction*. With this scheme, the number of new offspring also needs to be determined.
3. A modification on the steady-state reproduction scheme is the *Steady-State without Duplicates* scheme. Under this scheme all duplicate chromosomes are eliminated from the population, usually by mutation, to ensure that the diversity of the population is maintained.

The standard crossover operator works on two randomly selected bit string chromosomes, interchanging a section of one chromosome with the other. The simplest form of this type of crossover is the one-point crossover. In this crossover method a random number is generated which becomes the point where the chromosomes are split. For example, if we have two chromosomes, 10010101 and 01011011, and the point at which it is to be split is position 3. The two resulting chromosomes would be 01010101 and 10011011. A two point crossover technique swaps a section of one chromosome with the same section in another. A third popular crossover technique is the uniform crossover, whereby each bit in the chromosome is considered individually and randomly selected from either of the parents.

For chromosomes that are based on sequences, such as scheduling problems, five different types of crossovers have been developed to avoid having duplicate values within the chromosomes.

1. The "Partially Matched Crossover" (PMX) takes a randomly defined section of two chromosomes and swaps them. Duplicates that are caused by the crossover are removed from the non-swapped areas of the chromosome and replaced by the value that was previously in the position occupied by the other duplicate, in the swapped area before the swap. For example if the crossover section is to be between positions 3 and 6 in the following chromosomes:

A	9 8 4 5 6 7 1 3 2 10
B	8 7 1 2 3 10 9 5 4 6

After the PMX the resulting chromosomes are

A	9 8 4 2 3 10 1 6 5 7
B	8 10 1 5 6 7 9 2 4 3

2. The “Order Crossover” (OX) starts by selecting a crossover range. Before the area is swapped, the duplicate values are removed from the original chromosome, and the non-swapped area of the chromosome is then rotated from right to left so that the entire non-swapped area of the chromosome contains the jobs and the swapped area contains no jobs. The two original sections are then swapped. Using the same example as before we start with:

A	9 8 4 5 6 7 1 3 2 10
B	8 7 1 2 3 10 9 5 4 6

The potential duplicates are then removed

A	9 8 4 5 6 7 1 X X X
B	8 X 1 2 3 10 9 X 4 X

The non-swapped areas are then moved from right to left until the swap area is free.

A	5 6 7 XX X 1 9 8 4
B	2 3 10 XX X 9 4 8 1

Each section is swapped:

A	5 6 7 2 3 10 1 9 8 4
B	2 3 10 5 6 7 9 4 8 1

3. Cycle Crossover (CX) is based on swapping cycles within two chromosomes. To find the cycles, the first position is kept the same while the corresponding value in the second chromosome is then found, and where that value appears in the first chromosome this position is fixed. This process is repeated until it loops back to the original gene. Once the cycle is determined, all genes that were not found in the cycle are taken from the second string. For example if we have two chromosomes:

A	9 8 2 1 7 4 5 10 6 3
B	1 2 3 4 5 6 7 8 9 10

fix the first position and find the cycles in the sequences,

A	9 - - 1 - 4 - - 6 -
B	1 - - 4 - 6 - - 9 -

then cross the remaining values.

A	9 2 3 1 5 4 7 8 6 10
B	1 8 2 4 7 6 5 10 9 3

4. Uniform Order-Based Crossover, introduced by Davis (1991), is based on using a bit string template to determine which parent will contribute to the make up of the children. The first child has all the genes in the positions marked with a 1 copied from parent 1, while the second child has all genes marked with a 0 copied from parent 2. A

list of all the genes marked with a 0 from parent 1 (1 from parent 2) is then made and inserted into the blanks of child 1 (child 2) in the order of appearance in parent 2 (parent 1). For example:

Parent 1	9	8	2	1	7	4	5	10	6	3
Parent 2	1	2	3	4	5	6	7	8	9	10
Template	0	1	1	0	0	1	0	0	1	0

Initial Assignment:

Child 1	-	8	2	-	-	4	-	-	6	-	(Not included are 9,1,7,5,10,3)
Child 2	1	-	-	4	5	-	7	8	-	10	(Not included are 2,3,6,9)

Final Children

Child 1	1	8	2	3	5	4	7	9	6	10
Child 2	1	9	2	4	5	6	7	8	3	10

- Whitley, Starkweather and Shaner (1991) developed the “Genetic Edge Recombination” crossover scheme which was originally designed for “travelling salesman” type problems but can also be used for any sequencing problem such as scheduling. The method is based on considering which jobs are adjacent to each other, rather than considering the positions of the jobs within the schedule. The crossover operator starts by building up a list of adjacent jobs for each job in the schedule. A starting job is chosen, called the current job, and this job is eliminated from the adjacency list of the other jobs. From the adjacency list of the current job, a job is chosen which has the least number of adjacent jobs (ties are broken randomly) and this becomes the new current job. If there are no jobs left in the adjacency list for the job, randomly choose a job which has entries left or if all jobs have been sequenced then stop. For example:

A	3	1	2	4
B	1	2	3	4

The adjacency list is:

1:	2	3	2:	1	3	4
3:	1	2	4	4:	2	3

Start with Job 3, Choose the job with least adjacencies (either 1 or 4), randomly choose 4, choose 2 as this is the only one left in the adjacency list, choose 1, then stop. The final sequence becomes

3 4 2 1

Note that this crossover only produces one offspring unlike the other crossover techniques which produce two.

Syswerda (1991) studied the effect of different crossover functions in a timetable scheduling application. From his study, which compared PMX, OX and genetic edge recombination with a random search, it was found that both PMX and OX both performed very well, with PMX only slightly better than OX. Genetic edge recomposition, however, was found to be only slightly better than the random search that was implemented. Genetic edge

recomposition was designed specifically for travelling salesman type problems which are cyclic, and therefore actual positions within the sequence are not important, whereas this is not the case with scheduling problems. This may be an explanation for its poor performance on scheduling applications.

Using bit string chromosomes, the mutation operator inverts one of the bits in the chromosome from 0 to 1 or 1 to 0. With sequence based chromosomes, however, a more complex mutation operator is necessary to maintain a valid sequence. Lawton (1992) outlines two different types of mutation. The first type of mutation is called position-based mutation which randomly selects two genes within a chromosome and swaps them in a way similar to a swap move in other search techniques. The second type of mutation, called order-based mutation, like position-based mutation randomly selects two genes, however under this scheme the second gene is placed before the first gene in a similar manner to the insert move used in other search methods. Syswerda (1991) developed another type of mutation, called “scramble mutation”. This method takes a random sub-list from the chromosomes and then randomly scrambles the order of the genes within the sub-list.

Syswerda (1991) also compared three different mutation operators, position-based mutation, order-based mutation, scramble mutation and compared the use of a GA with these operators with a random search. The results of this study showed that order-based mutation was clearly superior, with position-based and scramble mutation having a similar level of performance, while the random search performed poorly.

3.4.4. Genetic Algorithm Machine Scheduling Applications

Biegel and Davern (1990) have applied GAs to both the single and multi-machine flow shop scheduling problem with the objective of minimising mean flow time. They used a chromosome representation based on job sequences with the PMX operator and no mutation operator. They suggested that the steady-state without duplicates extension be used, however they provide no computational experience with their technique to evaluate its performance.

Whitley *et al.* (1991) uses GAs to schedule a flow shop which has manufacturing cells and jobs that can be processed in parallel. This model is a simplification of an actual problem faced by schedulers in one of Hewlett-Packard's manufacturing facilities. The GA determines the sequence of jobs entering the flow shop, while two different heuristics determine which job is processed on which machine. It is these heuristics that enable the sequence to be evaluated with a view to minimising costs, whereby setup, processing and idle time costs are all given. The paper uses genetic edge recomposition and from computational experience it is reported that the GA performs well. No comparisons of

different parameters or different operators were given, nor were there any comparisons with other search techniques, making a performance assessment of this technique impossible.

Vempati, Chen and Bullington (1993) used GAs to solve the flow shop scheduling problem with the objective of minimising makespan. Each chromosome represented a sequence of jobs, while the PMX operator was used, without a mutation operator. The termination criteria used was to stop when over half the population had values equal to the best solution found so far. The results from computational experiments with problems ranging from 6-25 jobs and 4-10 machines showed that the GA was within 7% of the optimal, while it consistently produced better results than standard heuristics.

Mason (1992) considered the use of GAs in the single machine job class scheduling problem with setup times with the objective of minimising makespan. He developed several different chromosome encoding schemes, designed specifically for this problem and compares this to the standard schemes. OX is used along with another crossover technique similar to order-based crossover which he develops. Order-based mutation is also used. A parent/child elitism strategy is implemented, whereby the parent is only replaced by the child if the child is an improvement on the parent. One result obtained from this research was that a chromosome representation which incorporated a problem specific coding scheme, called a class run encoding scheme, performed better than the standard sequence of jobs scheme. Using the class run encoding scheme, a larger, more difficult problem set were run and compared with a standard heuristic called the Ahn & Hyun heuristic. From these experiments it was found that when the number of classes was increased (which produces less complex problems), the GA performed better than the Ahn & Hyun heuristic. However, when there were a small number of different classes (which are more complex problems) the Ahn & Hyun heuristic performed better than the GA as it generated results with a lower overall deviation from optimal, however the GA did find the actual optimal more often. From these results, the performance of GAs could be seen as having a high variability, finding some very good solutions but at the same time finding some very poor solutions as well.

Dorndorf and Pesch (1992) used GAs to solve the minimise makespan problem for job shop scheduling. They developed two GA implementations. The first GA, called priority-GA, used a unique chromosome structure based on resources, rather than jobs. When conflicts arose at a particular machine, they were resolved by using one of 12 dispatching heuristics. The chromosome therefore represented which dispatching heuristic is used at each resource. Since all genes are independent of each other, standard crossover and mutation functions can be used. Their second approach implemented a GA in conjunction with the “shifting bottleneck” heuristic of Adams, Balas and Zawack (1988). The GA provided the sequence of jobs for non-critical machines, while the shifting bottleneck heuristic determined the

sequence for the critical machines. In this implementation the CX operator was used exclusively without any mutation operators. Computational experiments were carried out on problems ranging in size from 5-15 jobs and 5-15 machines. The results from these experiments showed that the shifting bottleneck GA produced the best results overall, although it did take longer than the standard shifting bottleneck procedure. The priority-GA technique did not perform as well as the standard shifting bottleneck procedure, however it was speculated that it may perform better on larger problems.

Dagli and Sittisathanchai (1993) used a combination of both GA and neural networks to solve the job shop scheduling problem. The GA part of the model performed the search, while the neural network evaluated the schedules in terms of several different objective criteria. Chromosomes were used to indicate the sequence of jobs through each of the machines with the CX and order based mutation operators being used to manipulate the chromosomes. The technique was tested on small problems with known minimums. In all cases the GA found the minimum “within a few generations”, the exact number of generations is not given. Larger problems were then tested and compared with the solution obtained from the SPT heuristic, which demonstrated that it was superior. This result, however, is not surprising considering the complexity of the GA/neural net formulation in comparison to the simplicity of the SPT heuristic. The success of this approach on larger problems is therefore inconclusive.

Herrmann and Lee (1993a) investigated the use of a GA based on the “problem space” rather than sequence-of-jobs space for the class scheduling problem. A full explanation and discussion of the of concepts of heuristic space and problem space is contained in Section 3.2.2. The GA uses a six bit binary encoding scheme (representing a number from 0 to 63) for each job which represents a change in the deadline of the job from 0 to the maximum deadline in the problem. Using this type of chromosome representation allows standard crossover and mutation operators to be used, although the actual operators used in the GA are not stated. To convert the given problem and deviations contained in the chromosome into a final schedule, a “minimum waste” algorithm is used (refer to Herrmann and Lee (1993a) for a full description of this algorithm). Results from computational experiments on 30 job problems with 4 classes and 50 job problems with 10 classes found that the GA produced better results than the Ahn & Hyun heuristic.

Herrmann and Lee (1993b) used the same “problem-space” GA as Herrmann and Lee (1993a) to solve three different single machine scheduling problems, minimise makespan, minimise number of tardy jobs and minimise weighted tardiness. For each different problem, a different base heuristic was used to incorporate the major element of the objective function. The GA operators uniform crossover, one point crossover and standard mutation are all used with a steady-state reproduction scheme on a population of 100

chromosomes. In computational tests it was found that the GA approach improved the solution obtained by the base heuristic by between 2% and 90%, however no comparisons were carried out with other heuristics which are known to produce high quality solutions to these problems, hence our conclusions must be reserved.

Herrmann, Lee and Hinchman (1993) investigated the use of “heuristic-space” with GAs for the job shop scheduling problem. They used a machine chromosome representation which indicated what dispatching rule is to be used for each machine in order to determine its sequence. As each element is independent, standard crossover operators can be used in this problem. A total of fourteen different dispatching rules were used in these experiments. No computational experiments were carried out to measure the effectiveness of this type of GA, hence no conclusions can be drawn as to whether this model is an effective method of developing a job shop schedule.

Lash (1993) investigated the use of GAs to solve the common due date, weighted tardiness problem for a set of identical, parallel machines where the tardiness penalty is proportional to the processing time of the job. Two chromosome representations were tested and compared. The first scheme allocated each job a position within the chromosome, and the value of the gene indicates what machine that particular job will be processed on. This completely defines the schedule as once jobs are allocated to the machines, they are processed in LPT order, as this is a necessary condition for an optimal schedule for this problem. Using this scheme three different crossover techniques were trialed, these were: one-point crossover, two-point crossover and uniform crossover. Mutation under this scheme was to randomly choose a job and assign it randomly to a machine. The second scheme was based on each chromosome being a sequence of jobs. The jobs were then assigned to machines using a method called MULTIFIT. Once assigned to a machine, the jobs are sorted into LPT order. The crossover method used under this scheme is a one point order crossover, whereby the first part of child comes from one parent while the remainder of the child has values in the same order of appearance from the other parent. Computational experiments compared the two different GAs with a technique which generated a random sequence and determined the schedule using the MULTIFIT algorithm used in the second GA. Results showed that the first GA encoding scheme performed poorly, with the random technique producing better results. The second GA technique, however, performed as well, if not better than, the random approach and generally took less computational time.

Lee and Kim (1992) considered using GAs to solve the single machine, common due date, E/T scheduling problem. Due to the nature of the E/T scheduling problem, the entire schedule can be defined by assigning each job to be either early or late. A binary chromosome representation was therefore used to represent whether a job is early or late.

Standard one-point and two-point crossovers were used with a mutation operator which swaps a job from being early to being late or vice versa.

From computational experiments it was found that there was little difference between one-point and two-point crossover operators. Also experiments with varying crossover rates failed to produce results which were significantly different. Two different selection processes, the roulette wheel selection and a steady-state reproduction scheme, called N-Best selection, were used. It was found that roulette wheel selection was slightly better but N-Best selection was faster. Overall it was found that the error bound on problems with between 20 and 40 jobs was just 0.1-0.6%.

3.4.5. Conclusions

From the literature reviewed, as with SA, there appears to be many mixed results from the use of GAs for solving scheduling problems. Although many different chromosomal representations have been tried, the majority of successful schemes have used a sequence representation for their problems, and/or they have used algorithms to enforce the necessary conditions on the problem for optimality. Schemes that have been based on mixing dispatching heuristics have not performed well when compared to other approaches.

It appears, from the literature, that the best parameters for a GA are quite problem-dependent. Therefore any research into the use of GAs will require parameter tuning to obtain the best results.

The results of Lee and Kim (1992) are encouraging for the use of GAs in E/T scheduling problems. The native binary representation of GAs is clearly complementary to the common due date E/T scheduling problem, and therefore these two techniques have worked well together. One challenge is therefore to adapt the distinct due date problem so that it too can be manipulated in the same manner as the common due date problem.

3.5. Tabu Search

3.5.1. Introduction

Tabu Search (TS) is a search technique that is superimposed on other local search techniques to stop them becoming trapped in local minima. It was originally developed by Glover (1986) to solve integer programming problems however elements of TS were first proposed by Glover (1977). Since then many introductory papers to TS have appeared (Glover 1989, 1990a, 1990b, 1990c; Glover and Laguna, 1993). de Werra and Hertz (1989) liken TS to a deterministic form of SA, as both SA and TS both incorporate local search procedures within a *metaheuristic*. Unlike SA, TS does not accept the movement to a poorer solution on the basis of probabilities but makes its decision based on its *memory* of previous moves it has made. It is the incorporation of memory within the search technique that differentiates TS from other search techniques. The purpose of the memory in TS is to stop it from revisiting solutions it has already evaluated, and is therefore the primary means of stopping the search from cycling, unlike SA which has no such means.

There are three different types of memory that have been proposed for TS; short term memory, intermediate term memory and long term memory (Glover, 1990c); however only short term memory is required to implement TS. The purpose of these different types of memory is to help achieve a balance between intensifying the search in areas where there is potential for higher quality solutions to be found, called *intensification strategies*, and diversifying the search to new areas where there could be even better solutions, called *diversification strategies*.

TS has been used extensively for solving many different types of NP-hard problems, including many different types of scheduling problems. A difficulty with TS is that if the neighbourhood of the problem is relatively large, the search may become slow in moving from one solution to another. If insufficient time is allowed for the search, then all that may happen is that a gradient descent may occur and the tabu search itself may never start (Reeves, 1993). To overcome this problem several modifications to the TS selection process, which limit the size of the neighbourhood, have been suggested.

3.5.2. Overview of the Method

The use of short term memory is the most essential element in TS. Its primary purpose is to diversify the search into areas of the solution space that have not been explored by placing restrictions on the moves that can be made. Ideally these restrictions would record the entire solution permanently so it could not be revisited, however this would require excessive amounts of memory for large problems, hence a *tabu restriction* is placed on the elements that make up a move in such a way that the move can not be “undone” by a subsequent move.

The key element in the TS short term memory is the “*tabu list*” which records the moves that cannot be performed and are regarded as “tabu”. The most common form of tabu list is to have an array which records a given number of tabu moves. As a new move becomes tabu it is added to the front of the tabu list. When the tabu list becomes full, the oldest tabu move is dropped from the tabu list, and in doing so loses its tabu status, and the new move is added to the front of the list. Glover and Greenberg (1989) call this process “strategic forgetting” which incorporates an interaction between the “learning” and “unlearning” processes. The key parameter which determines this interaction is the size of the tabu list. Glover and Greenberg (1989) note that many of the applications using TS have found the ideal tabu list size to be between 5 and 9, however Barnes and Laguna (1991a) point to other research which has found that the ideal tabu list size to be proportional to the size of the problem. Tsubakitani and Evans (1992) found that the size of the tabu list is also dependent on the size of the neighbourhood being used. If a large neighbourhood is being used, then there is less chance of the search staying in the same area of the solution space, hence the tabu list size can be smaller.

Tabu lists can also be made to vary in size, as in Skorin-Kapov and Vakharia (1992). By controlling the size of the tabu list, the search can control its intensification and diversification strategies. When promising new solutions are found, the tabu list size can be shortened to explore the area more fully - intensifying the search, whereas if no new promising solutions are found in an area, the tabu list size can be extended to diversify the search into other areas.

Short term memory also incorporates an intensification strategy which concentrates the search on a particular area by overriding the tabu restriction if certain “aspiration criteria” are met. The most common aspiration criterion used in the literature is to accept the move, regardless of tabu restrictions, if the solution is better than the best solution found so far. Clearly this solution has not been visited before as the best solution found so far would have to be equal to this value if it had been previously visited. Three other types of aspiration criteria have been suggested by Glover and Laguna (1993). These are:

1. Subdivide the solution space into regions, and accept the move if it is the best cost within the region.
2. If the last move made was an improvement in cost, then accept the move if it improves the cost.
3. If the change in the objective function is relatively small and there has been a relatively large change in the objective function since establishing the move’s tabu status, then accept the move. For this type of criterion, what is regarded as a small change and a large change needs to be defined.

The algorithm for the short term memory component is outlined in Figure 3-11.

1. Begin with a starting current solution from either an initialisation routine or from intermediate or long term memory components.
2. Create a list of candidate moves
3. Choose the best admissible candidate by:
 - 3.1 Evaluating each Candidate move
Does the move yield a higher evaluation than any other move found admissible so far from the current candidate list?
If Yes go to 3.2, otherwise go to 3.5
 - 3.2 Is the candidate Tabu?
If Yes go to 3.3 otherwise go to 3.4
 - 3.3 Does the move satisfy the aspiration criteria?
If Yes go to 3.4 otherwise go to 3.5
 - 3.4 The move is Admissible therefore designate as the best admissible candidate.
 - 3.5 Is there a “good probability” of better moves left, or should the candidate list be extended?
If Yes go to 3.1, otherwise go to 3.6
 - 3.6 Return the current best admissible candidate
4. Stop if a specified number of iterations has elapsed in total or since the last Best Solution was found. If Stopping criteria is met go to 6, otherwise go to 5.
5. Update the Admissibility conditions.
6. Terminate globally or transfer. A transfer initiates an intensification or diversification phase embodied in an intermediate or long-term memory component.

Figure 3-11. Tabu Search Short Term Memory Algorithm (Glover, 1990c)

Intermediate and long term memory are used to intensify the search and diversify the search respectively.

Intermediate memory operates by finding the common elements of the best solutions found during the search and using this to encourage search to explore solutions with these elements (Glover and Greenberg, 1989). The search is therefore intensified in a particular area of the solution space as movements which produce solutions inconsistent with these features are penalised.

The purpose of long term memory is to diversify the search by finding new starting points that are in entirely new areas of the solution space (Glover and Greenberg, 1989). This is done by penalising those solutions that are similar to those which have already been examined by the search, forcing the search into new areas of the solution space.

Two different memory structures have been proposed for intermediate and long term memories. The first scheme, called tabu time, was proposed by Glover (1990c). This scheme uses a tabu status array which records the iteration number when each move was last made tabu. These longer term memories can then use this information to determine

those attributes that have been made tabu recently and those attributes that have not been made tabu at all. Intermediate term memory would use the information to concentrate on incorporating those moves which are more recent, while long term memory would concentrate on incorporating those moves that have had little use. Another scheme, proposed by Glover (1991), is to use frequency based memory. This memory records the number of times a move has been made tabu. From the frequency of use, each attribute can be penalised appropriately according to the intensification or diversification goals of the memory. Intensification strategies would penalise low usage moves, while diversification strategies would penalise high usage moves. Glover (1991) also proposes the use of this type of memory within the tabu search's short term memory process so that when there are no non-tabu moves that provide an improvement, this memory can then be used to diversify the search away from those moves that have a high frequency count, and explore moves with low frequency counts.

In summary there are seven major issues that must be addressed when using TS. These are to:

1. Determine the *construction algorithm* to be used to create an initial starting point from which to start the search. Most TS applications use a simple heuristic to determine the starting point. Taillard (1990) notes that the number of iterations required to find the optimal solution will depend greatly on the initial starting solution, hence the importance of this part of the TS.
2. Determine which *neighbourhood scheme* to use. Research has shown that the neighbourhood scheme used can have a large impact on the success of the TS (Barnes and Laguna, 1991a).
3. Determine the way in which the tabu restrictions operate, called the *tabu conditions*. This will define the way in which the tabu list will operate.
4. Determine the *tabu list size* and the *type of tabu list*. Both fixed and varying list sizes have been experimented with in the literature.
5. Determine the *Aspiration criteria*. The most common criterion is to accept a move if it is better than the best move so far.
6. Determine the *stopping criterion* for the search. Common options are an absolute number of iterations, a number of iterations without improvement or a specific amount of time.
7. Determine how *long term* and *intermediate term memory* are incorporated into the search. Issues such as how this memory will operate and how it is integrated into the short term memory need to be resolved.

3.5.3. Tabu Search Extensions

In some problem formulations the optimal solutions may not be easily found from a regular search path due to the way the feasible regions are formed within the solution space or

because of a tendency for the algorithms to stay around a fixed area. Glover (1989, 1990) proposed a diversification strategy called “*strategic oscillation*” to overcome this problem. The underlying principle behind strategic oscillation is that it forces the search to move away from and then back to the search area that the standard TS would have normally stayed in. This movement can take one of two forms:

1. Crossing feasibility boundaries, then being forced to return, and
2. Moving away from areas where the search has become stuck due to high costs involved in getting out of the area around the local minima.

One possible way to implement such a scheme, suggested by Glover (1990), is to use “*adaptive penalties*” which allow some constraints to be temporarily violated as the search moves through infeasible regions. These penalties will change as the solution moves toward the problem boundary, allowing a certain number of moves into the area of infeasibility, before forcing a return. Strategic oscillation does not appear to have been used in any of the TS scheduling applications, however Glover (1989) mentions that it has been used successfully in other applications.

To overcome the problem of tabu search being slow with large problems and large neighbourhoods, three different techniques have been proposed in the literature. The first technique was used by Laguna, Barnes and Glover (1991) and Laguna and González Velarde (1991) and it effectively places a search “*window*” on the neighbours that are examined. The window restricts how far apart, in terms of a given number of jobs, that any two jobs selected by the neighbourhood scheme can be within the sequence. The second technique, called “*Revised Tabu Search*”, was proposed by Reeves (1993). This technique again only considers part of the neighbourhood before making a move, however this time the tabu search randomly orders the jobs in the schedule and, from this random ordering, selects the first job and evaluates all its neighbours. It continues with the next job until a specified number of jobs have been processed. The ratio of the number of jobs to process to the total number of jobs, is again referred to as the “*window size*”. As the window size is reduced, the search moves to new solutions faster than standard tabu search, however the ‘quality’ of the move is unlikely to be as high. There is therefore a time/quality tradeoff in the problem which is dependent on the size of the problem being solved. The third technique, called the “*restricted neighbourhood*” was proposed by Adenso-Díaz (1992). This scheme based on an exponentially reducing window which starts with testing the entire neighbourhood, that is a window size of 1, then after a given number of iterations the window size exponentially reduces with each iteration to a minimum size. This type of window was found to reflect the distances in the neighbours used for the type of flow shop problem being solved hence was found to increase the search speed with little effect on solution quality.

There are many other extensions to TS proposed in the literature, however these are two common extensions which address problems which are specifically relevant to the distinct due date E/T scheduling problem.

3.5.4. Tabu Search Machine Scheduling Applications

This section outlines some of the TS scheduling applications which have been discussed in the literature.

Widmer and Hertz (1989) investigated using TS to solve the flow shop scheduling problem with the objective of minimising makespan. In their implementation of TS they used a swap neighbourhood scheme, and a tabu list which recorded both jobs involved in the swap. The tabu list size was set at 7 with a stopping criteria based on computational time. Computational experiments compared TS with other heuristics on 500 different problems with 4-20 jobs and 4-20 machines. In these experiments the TS method found the best solution to the problem 80% of the time, whereas the best, standard heuristics found the best solution only 55% of the time. The performance of the TS also improved as more time was allowed for it to search, raising its accuracy to around 90%.

Kim (1993) used the TS of Widmer and Hertz (1989) and compared it to other standard heuristics and some other search techniques on problems ranging from 15-50 jobs and 5-25 machines. The results of this experiment showed that TS outperformed all the other standard heuristics trialed, however this came at the experience of a large amount of computational time. An interesting result in this research was that although the overall average performance of TS was the best of all the heuristics, in almost 25% of the test problems TS did not obtain the best final solution, as these came from other heuristics which had a higher result variability.

Taillard (1990) investigated the use of TS for the same minimise makespan flow shop problem as Widmer and Hertz (1989), however his implementation is quite different. Taillard (1990) uses the insert neighbourhood scheme with a tabu list which records the last value of the objective function rather than the move. Moves that have the same objective value are all regarded as being tabu, rather than just the one specific move. This eliminates the need for testing for the standard aspiration criteria as if a move has a lower cost, then by definition this will not be tabu. The TS is tested using 200, 9 job, 10 machine problems and 550 problems ranging from 10-50 jobs on 10 machines, all randomly generated. A varying tabu list size was used, starting with a list size of 7 and ending with a list size of 100. Three different stopping criteria were trialed:

1. using a fixed number of iterations,
2. terminating after a given number of iterations without an improvement to the best solution, and

3. running several TS searches in parallel each starting from a different starting point, comparing the results of each search after a given number of iterations and terminating if two of the searches have the same best solution found.

It was found that terminating using a number of non-improvements and the parallel tabu search methods both produced similar results when using a non-parallel computer. Although not explicitly stated within this paper, it appears that the starting point used by the search is generated by a standard heuristic, and this forms the basis for Taillard's statement that TS can get better solutions than standard techniques at the expense of computational time.

Reeves (1993) considered the same flow shop scheduling problem as Widmer and Hertz (1989), except in his implementation he used the revised version of TS and then compared this to standard TS and simulated annealing. Computational experiments involving 20 to 500 jobs and 5 to 20 machines showed that, on average, the solutions from the Revised TS had half the deviation from the best known solutions that standard TS had, and almost a third when compared with SA. This result, as a consequence, also shows that both TS were superior to SA for this particular problem.

Skorin-Kapov and Vakharia (1992) used TS to schedule a flow shop which produces families of jobs with the objective of minimising makespan. This problem has two different levels, the scheduling of jobs within a given family group and the scheduling of the family groups themselves. To accommodate this, two tabu search lists and neighbourhoods were used, one dealing with families, the other with jobs within families. The adjacent pairwise interchange neighbourhood scheme was used on both jobs and families, with the tabu list preventing jobs or families of jobs being swapped back to their original position. Both fixed and varying length tabu lists were used. The varying tabu list was controlled by the number of iterations without improvement, initially being decreased if no improvement was made after a given number of iterations (intensifying the search), then increased if still no improvement is made (diversifying the search).

Skorin-Kapov and Vakharia (1992) also implemented intermediate and long term memory structures using frequency based memory. These memories recorded the number of times a family or job had occupied a particular position within the schedule. Intermediate term memory operated by placing families in the sequence based on the highest frequency values, whereas long term memory operated by placing families in the sequence based on the lowest frequency values. In the computational experiments, only one of these two memories was implemented at any one time. Computational experiments were carried out on various small problems, and the results were compared with a SA technique. From these computational experiments it was found that the TS outperformed SA in terms of both solution quality and time. The TS which incorporated long term memory was found to take more computational

time than the TS which incorporated intermediate term memory, showing that diversifying the search into areas which have poor quality solutions can have a large impact on the overall performance of the search. In general, both intermediate and long term memories improved solution quality but at the expense of computational time. In two-thirds of the problems tested the variable tabu list model found better results than the fixed length tabu list model, hence this type of tabu list was preferred for this type of problem.

Adenso-Díaz (1992) used TS to solve the permutation flow shop problem with the objective of minimising total weighted tardiness. For this problem a restricted neighbourhood scheme is used on problems with between 10 and 35 problems and 5 and 20 machines. This technique starts with a swap neighbourhood scheme and if after a given number of iterations there has been no improvement in the best solution using this neighbourhood it switches to the insert neighbourhood scheme until it finds a better solution, then it reverts back to the swap neighbourhood again and continues this switching until the maximum number of iterations has been reached. Computational experiments found that when this heuristic was compared to the TS of Widmer and Hertz (1989) (presumably with the different objective function) it was found that the solution quality was slightly better, but the speed was improved by up to 50%.

Widmer (1991) used TS to solve the job shop scheduling problem with setup times. This problem has several objectives which are weighted and summed in order to produce a single objective function. These are: minimise makespan, minimise the number of tooling changes, minimise the number of late parts and minimise the number of machines not able to complete their work during the normal production hours. The insert neighbourhood scheme was used, with the tabu condition which stops the job moving back to its original position. The search is stopped after a given number of iterations without improvements. Experiments comparing TS, with various parameters, and a standard technique, which was using a single objective criteria of minimising tool switches, showed that the TS found results that were the same as those of the standard technique. Other experiments using 100 randomly generated problems found that the best tabu list size was 1, whereas the number of iterations without improvement required before stopping varied between 100 and 1000 depending on the size of the problem.

Barnes and Laguna (1991b) considered solving the identical parallel machine scheduling problem, with the objective of minimising the weighted delay in processing of all jobs, by using TS. This problem has a property, called a natural ordering, in which all jobs on a machine must be in nondecreasing order of the ratio of the job's processing time to the job's weight. The neighbourhood used is a combination of both swap and insert moves between machines. Due to the natural ordering of this problem the actual sequence of the jobs is determined automatically. A tabu list size of 8 was used with the search terminating after

500 iterations for smaller problems and 1000 iterations for larger problems. An interesting comment made by Barnes and Laguna is that intermediate and long term memory were not implemented as preliminary results showed they were “not helpful”. No other information was given on these preliminary tests of other memory structures, however this serves as a warning that these other memories may not increase the performance of TS. In computational experiments TS was compared to a branch and bound solution. In each case the TS found the same solution to the branch and bound but in far less computational time.

Hübscher and Glover (1992) examined the use of TS to solve the identical parallel machine scheduling problem with the objective of minimising makespan. They used swap and insert neighbourhood schemes that operate between machines along with, what is termed, a *dynamic tabu list*. This tabu list recorded the machines involved and the processing times of the jobs involved hence making tabu all other moves involving the jobs of the same length on these machines. The tabu list had both a static part and a dynamic part. The static part of the tabu list was like a standard tabu list, whereas the dynamic part of the tabu list changes in what it recognises as being present in the tabu list. Initially all of the dynamic part of the tabu list is recognised, next none of it is recognised, then only the earlier half of the list is recognised, then the later half is recognised, then the first and last quarters are recognised, then finally the middle half is recognised. The parts of the dynamic tabu list recognised changes after a given number of iterations into its next phase, and once it has gone through all the phases it then starts from the first stage again. The purpose of this is to vary the diversification and intensification strategies during the search. The method was tested using problems with between 2 and 50 machines and 50 to 2000 jobs. The static part of the tabu list was set at 12 while the dynamic part was set at 9 for all the test problems. Another modification made to the standard technique was to include a diversification strategy which, after a given number of steps (3000 and 6000 steps were used), will randomly reassign jobs between two different machines. No comparative tests were done to show the effectiveness or otherwise of the dynamic tabu list, however the diversification strategy did provide better solutions at the expense of more computational time. The TS method was compared to only one standard problem in the literature, for which TS performed well achieving a solution lower than the standard technique but higher than the theoretical optimum.

Laguna and González Velarde (1991) also dealt with the identical parallel machine problem, however their objective was to minimise the weighted earliness penalty with deadlines. A modified Smith-heuristic (Chand and Schneeberger; 1988) was used to obtain a starting point for the search. Both swap and insert neighbourhood schemes were used. To decrease the size of the neighbourhood a window was used on the neighbourhood. A further modification was done to the search by implementing a branch and bound procedure after the TS had terminated to find optimal job sequences on the machines. Tests comparing the

standard TS with the TS and branch and bound showed that the solution quality increased with very little increase in time, however Laguna and González Velarde note that this result is only possible because the problems they were testing were small. Various tabu list sizes were experimented with, however the value $((\sqrt{n}) + 5)$ was found to give the most consistent results for differing problem sizes. No comparison was made between this technique and any other search or optimal techniques. The only comparison provided was with the standard heuristic which was used as the starting point. From this point the TS produced solutions which were, on average, 10% better.

Tsubakitani and Evans (1992) used TS to solve the total tardiness single machine scheduling problem. A greedy heuristic was used to construct the initial schedule for the search, while the adjacent pairwise interchange and the swap neighbourhood schemes were compared. The tabu condition used was that the job moved forward in a move is recorded in the tabu list and is not allowed to be moved back. Computational experiments with 100 job problems found that a tabu list size of 84 was best for the API neighbourhood while a tabu list size of 12 was best for the swap neighbourhood. The stopping criteria used was based on a 3 minute time limit. The TS techniques were then compared with the initial greedy heuristic and two gradient descent heuristics, each based on the two neighbourhoods used in the TS. Results showed that the swap-based TS was superior to all the other techniques tried, followed by the gradient descent method based on the swap neighbourhood scheme, however the better results from the TS were obtained at the expense of greater computational time. The percentage improvement in the solution was just over 3.5% by the TS technique and just under 3% for the local search technique, hence overall the results do not indicate large improvements in solution quality, with little gain being obtained after the initial gradient descent.

Laguna, Barnes and Glover (1991) used TS to solve the single machine scheduling problem with the objective of minimising the sum of setup costs and delay penalties, where each job had a sequence dependent setup cost and a unique waiting penalty. Swap, insert and a combination of swap and insert neighbourhoods were used, but the neighbourhoods were restricted by a 50% window. The tabu condition used was that a job moved forward by a swap could not be moved back in the schedule, while a job moved by an insert move could not be moved at all. Tabu list sizes from 6 to 10 were tried, while a time based stopping criteria was used. No long term memory was used, as preliminary work on this type of memory with this problem gave inferior results to techniques without these memories. Once again no details were given regarding this earlier research. Two standard heuristics are used to generate the initial starting solution. Using the swap based TS, an experiment was carried out, calculating the best, worst and average values obtained from the TS using the different parameters. In no case did the deviation for the TS become greater than 6.5% from the optimal or best known solution. When the three different TS neighbourhoods

were compared, all using a tabu list size of 7, the TS using both swap and insert neighbourhoods gave the best results, followed closely by the insert neighbourhood scheme. The swap neighbourhood scheme was the least effective technique.

Laguna and Glover (1993) extended the work by Laguna *et al.* (1991) by using TS with another technique called Target Analysis (TA). The TA portion implements a long term memory function to diversify the search by using frequency based memory which records the frequency of moves involving different jobs. This information is then integrated into the TS by considering it when selecting the next candidate, with moves that have a high frequency being discouraged. Computational experiments comparing the standard TS method with the TS/TA approach found that TS/TA found superior solutions to TA in the same amount of time.

3.5.5. Conclusions

Barnes and Laguna (1991a) carried out a review of scheduling literature which used TS. From this review, seven conclusions were made about implementing TS in scheduling problems. These are presented below with comments from the literature reviewed in this chapter which support or contradict these conclusions.

1. It is beneficial to the eventual success of the TS to obtain one or more good starting solutions through heuristics rather than having a random starting point, especially in larger problems although this is not quite as important for TS as it is for other methods, such as SA. Taillard (1991) notes that the number of iterations required by tabu search depends strongly on the initial starting solution.
2. The efficient selection and evaluation of moves is the single most important factor for TS production scheduling problems. Evaluation procedures need to look at the structure of the problem so that move evaluation can be streamlined. This type of streamlining is evident in Barnes and Laguna (1991b) and Laguna and González Velarde (1991) who used the properties of the identical parallel machine problem to reduce the size of the neighbourhood.
3. It is beneficial to incorporate more than one class of moves in the heuristic as this widens the neighbourhood base. The advantage of several different types of neighbourhoods was evident in Laguna *et al.* (1991) which found that a combination of two different neighbourhood schemes was superior to these neighbourhoods performing on their own. This modification will, however, increase the time taken to execute an iteration due to the larger number of possibilities to calculate.
4. Simpler tabu list attributes are preferred to complex attributes. In the literature reviewed in this chapter, no comparative studies were made between different tabu list attributes and their performance difference, however in general the majority of papers used simple tabu list attributes, which can support this conclusion.

5. Attention must be paid to the ideal tabu list size. Barnes and Laguna (1991a) reject the notion of a universal tabu list size and state that the list size will vary with each application and neighbourhood definitions. Glover (1991) however maintains that the size of seven appears to be a good choice for many TS applications, Laguna and González Velarde (1991) promote the ideal tabu list size as being around $(\sqrt{n})$ and Tsubakitani and Evans (1992) suggest that the tabu list size should be proportional to the neighbourhood size. There is clearly no consensus on this issue.
6. Diversification strategies were found to be essential to the success of TS. This conclusion by Barnes and Laguna (1991a) is not as well supported by the literature reviewed here. The issue of diversification is two fold. Within the short term memory element of TS, the diversification issue is part of the tabu list size and type of tabu list debate. With other diversification strategies, based on long term memory, some found that it was not of any benefit (Barnes and Laguna, 1991b; Laguna and González Velarde, 1991), others found that diversification schemes were beneficial to solution quality but at the expense of computational time (Hübscher and Glover, 1992; Skorin-Kapov and Vakharia, 1992), while others found diversification schemes beneficial without any time penalty (Laguna and Glover, 1993). The only conclusion that can be made on this issue is that the success or otherwise of diversification strategies may be greatly influenced by the way in which they are implemented and the particular problem being solved. More research is needed in this area.
7. Hybrid methods which combined TS with other techniques have also caused greater improvements. An example of this is the improvements found by Laguna and González Velarde (1991) when TS was used in conjunction with branch and bound.

The underlying theme of the results through all of the literature about TS and modifications to the standard technique is the time versus quality tradeoff. Modifications that have achieved better results will often cause an increase in the computational time required to solve the problem. Examples of this are the widening of the neighbourhood scheme to include multiple neighbourhood moves and the inclusion of long and intermediate term memory. Notable exceptions to this rule, however, are the development of the Revised Tabu Search (Reeves, 1993) and the use of Target Analysis as a form of long term memory (Laguna and Glover, 1993).

The benefits of many of these modifications will greatly depend on the problem being solved and the properties of the problem itself. What is encouraging is that when TS has been applied to many scheduling problems, it has performed successfully, outperforming both standard heuristics and other search techniques, such as SA.

A major criticism of much of TS literature to date is that it does not provide adequate, fair comparisons of TS with other heuristics or other search techniques. Too often no ‘control’

is used in the computational experiments to allow an objective measurement of how good the TS, or a modification to the standard TS, is to a standard technique. The majority of papers compare a standard heuristic which is also used as the starting point for the TS, ignoring the fact that because of this the TS is never going to perform worse than the standard heuristic and that the TS is actually spending a great deal more computational time in solving these problems. In essence the comparisons made are not fair in that they are comparing a low quality/fast technique with a high quality/slow technique then judging performance on quality of the solutions produced! Often these papers do not determine the optimal solutions either, therefore not revealing how well the TS is performing in a global sense.

TS is a search technique that has been applied to many different scheduling problems and has been successful in providing solutions which are superior to both standard heuristics and other search techniques. None of the TS literature to date has addressed the single machine E/T scheduling problem and therefore this provides a fertile ground for research.

3.6. Summary and Conclusions

The literature shows that SA, GA and TS search techniques have been used to solve a variety of scheduling problems. The SA and GA literature appears to have had some mixed results, however most of the literature using TS reports good results. The literature that provides comparisons between different techniques has shown that TS is superior to SA, however this may be problem dependent.

A critical element of success, common to all search techniques, is in the way the problems are represented, that is either the neighbourhood scheme or the chromosomal representation that is used for the problem. Neighbourhood schemes that were customised to the particular problem being solved; such as in Mittenthal, Raghavachari and Rana (1993), Lee and Kim (1992), and Laguna and González Velarde (1991); were particularly successful.

Research to date on the effect the starting point (or initial population) has on the quality of the final solution is mixed, indicating that this may be problem and/or implementation dependent. Care is therefore needed when implementing new types of searches or searches on new problems as the effect the starting point has on the end result is unknown.

Lee and Kim (1992) addressed the common due date E/T scheduling problem, however none of the search technique literature has addressed the distinct due date E/T scheduling problem. Therefore this area has potential for further research. Examples of how each of these search techniques can be used to solve the E/T scheduling problem using the neighbourhood and chromosome schemes explained in this chapter are outlined in Appendix I

4. Idle Time Insertion for E/T Scheduling Problems

4.1. Introduction

Research into the distinct due date E/T scheduling problem with idle time has split the problem into two distinct phases: a sequencing phase and an idle time insertion phase (Baker and Scudder, 1990). Figure 4-1 shows these two phases pictorially, whereby the sequence provided may be, as in this case, 4-1-2-3. The idle time insertion phase then takes this sequence and assigns optimal completion times to each job, effectively inserting idle time as necessary. In Figure 4-1, for example, idle time has been inserted between job 1 and job 2.

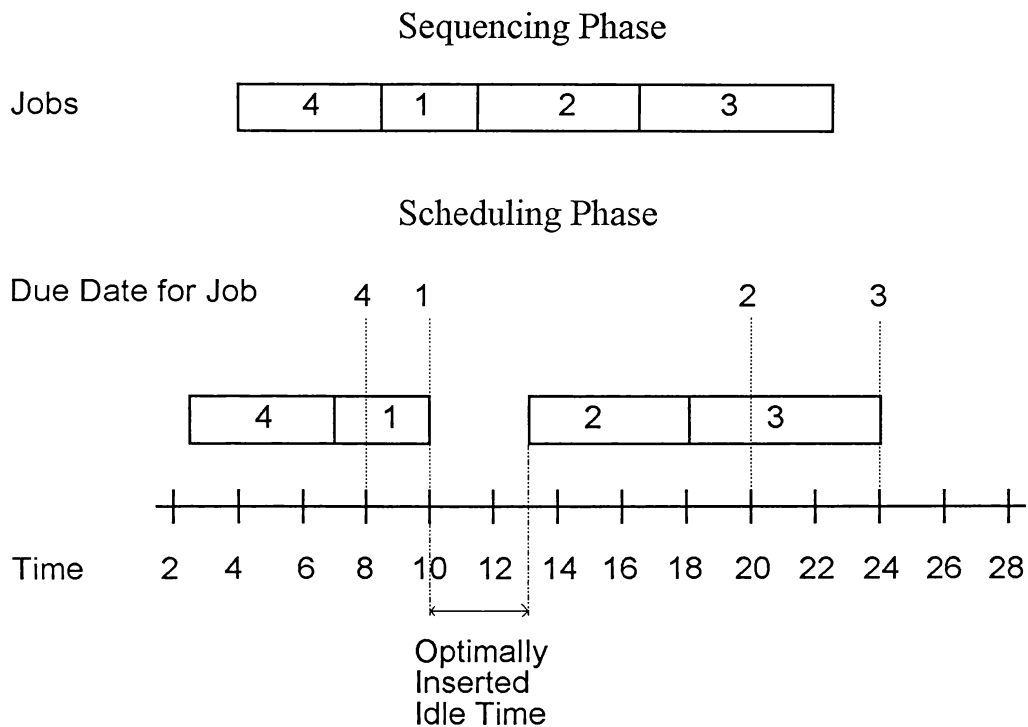


Figure 4-1. The Two Phases in E/T Scheduling

Methods of determining and manipulating this sequence were outlined in Chapter 3. In this chapter we now focus on the idle time insertion phase. The various idle time insertion algorithms outlined in the literature are presented in Section 4.2. Section 4.3. develops a new idle time insertion algorithm based on the concept of 'conflict resolution' which starts with an infeasible solution and progressively resolves the conflicts between jobs that are present in the schedule until the algorithm reaches an optimal solution. The conclusions from this chapter are then presented in Section 4.4.

4.2. Literature Review

The distinct due date E/T scheduling problem, unlike many scheduling problems, is not completed once the sequence of jobs is determined. Once a sequence is given, completion times for each job need to be determined. The optimal completion times for the job sequence may cause the machine to be idle at various times throughout the sequence. The process of determining the completion time of each job is commonly referred to in the literature as *inserting idle time* into the schedule.

Yano and Kim (1991) note that, since the idle time insertion routine may have to be executed many times during a search to find a high quality solution, it is very important that this routine is efficient.

Some researchers, (Abdul-Razaq and Potts 1988, Ow and Morton 1989) have avoided the issue of inserting idle time by assuming that idle time is prohibited. Once the sequence is defined only the starting time for the sequence needs to be determined. This assumption was discredited by Baker and Scudder (1990) who stated that this form of the problem is “inconsistent with the E/T criterion”. It is clear that for the general problem idle time is essential for optimisation, however, in certain classes of E/T scheduling problems, for example where the majority of jobs are going to have to be scheduled late, it may be possible to ignore idle time without a deterioration in solution quality. Robb and Silver (1994) implemented a partial idle time insertion routine which inserts idle time only at the start of the schedule, not between jobs. This provides a compromise between time and solution quality, which may prove adequate for some problem classes (refer to Section 2.3.2). As it is the general E/T scheduling problem being addressed in this thesis, it is assumed that it is possible to insert idle time at any point, not just at the beginning of the sequence.

Fry, Armstrong and Blackstone (1987) formulated the idle time insertion problem into a linear programming model. They then took this model and developed it into a one-pass implementation of the simplex algorithm (Dantzig, 1963), which will find the optimal schedule for a given sequence. The author found that the algorithm of Fry, Armstrong *et al.* (1987) contained three errors and omissions. Once these errors and omissions in the algorithm are corrected, the algorithm works efficiently. The corrected algorithm is presented in Figure 4-2 with the corrections in bold.

Davis and Kanet (1993) developed an algorithm, called TIMETABLER, that is very similar to the one-pass implementation of the simplex algorithm developed by Fry, Armstrong *et al.* (1987). This algorithm schedules the jobs from the last job in the sequence to the first job in the sequence. A job is scheduled at the point where the cost of scheduling the job any later would increase the overall cost of the schedule. Other jobs already scheduled may also be

moved to the right (i.e. later) if necessary. Once all the jobs have been scheduled in this manner, an optimal solution is obtained. Due to their similar nature the two algorithms have the same complexity.

Let

- P = position of the current job being considered.
- A_k = the set of jobs coming after idle time k and before $k+1$
- A_k^+ = all the early jobs in A_k
- A_k^- = all the late jobs in A_k
- I_k = the length of each inserted idle time period.
- W_k = the change in the penalty for increasing the k th idle time by one unit.

1. Set $P = n$, $k = 1$, $W_k = 0$, I_{k-1} = a large number, $I_k = 0$, $A_k = \{\}$; $A_k^- = \{\}$
Compute c_i based on zero idle time.
2. $A_k = A_k + P$; if $c_i > d_P$ then go to 5
3. $W_k = W_k - \alpha_P$; $A_k^- = A_k^- + P$
If $W_k \geq 0$ then idle time is not to be inserted and go to 6, otherwise idle time is to be inserted and go to 4.
4. Set $\Delta = \min \left[\min_{l \in A_k^-} (d_l - c_l); I_{k-1} \right]$
Set $c_j = c_j + \Delta$, $j \in A_k$
Case 1: $\Delta = d_l - c_l$, $l \in A_k^-$
Set $W_k = W_k + \alpha_l + \beta_l$; $A_k^- = A_k^- - l$
 $I_k = I_k + \Delta$
 $I_{k-1} = I_{k-1} - \Delta$
if $W_k \geq 0$ then
 $k = k + 1$
 $I_k = 0$
 otherwise
 go to 4.
Case 2: $\Delta = I_{k-1}$
 $k = k - 1$
 $I_k = I_{k+1} + I_k$
 $I_{k+1} = 0$
 $W_k = W_k + W_{k+1}$
 $A_k^- = A_k^- + A_{k+1}^-$
 $A_k = A_k + A_{k+1}$
 if $W_k < 0$ go to 4.
 $k = k + 1$; $I_k = 0$
go to 6
5. Set $W_k = W_k + \beta_P$
6. Set $P = P - 1$; if $P \geq 1$ then go to 2, otherwise stop.

Figure 4-2 Corrected Idle Time Insertion Algorithm of Fry, Armstrong *et al.* (1987)

Yano and Kim (1991) proposed a dynamic programming approach for inserting idle time. They schedule each job in reverse order, either on its due date or just before the currently scheduled jobs. If it is placed just before the currently scheduled jobs, then all the jobs from

this point through to the next section of idle time are rescheduled. In the process, the idle time may be eliminated in which case, the jobs are rescheduled up to the next subsequent section of idle time. This process is very similar to that of Fry, Armstrong *et al.* (1987).

The corrected Fry, Armstrong *et al.* (1987) was implemented as it was well documented and the other techniques do not appear to have any speed advantage by having a lower time complexity.

4.3. Theoretical Development of an Idle Time Insertion Algorithm

In this section an idle time insertion algorithm based on ‘conflict resolution’ is developed. The algorithm starts with an infeasible schedule with a cost of 0, and progressively resolves the conflicts within that schedule. Through this process we also develop some properties of the distinct due date problem.

Consider a schedule where all jobs are scheduled on their due date. This schedule has a cost of 0, the best possible cost, however this schedule may be infeasible due to conflicts between several different jobs as they need to be processed at the same time. The conflicts between these jobs can be resolved by forming these jobs into *groups*. A group is defined as ‘a set of jobs (between 1 and n) which are processed consecutively and without delay, that is no idle time exists between any of the jobs within the group’.

Using the concept of groups several theorems can be developed.

Theorem 4.1

If each group is scheduled optimally without consideration for any other groups within the schedule and if two consecutive groups are in conflict because they both need to be processed at the same time, then an optimal schedule will exist which does not have any idle time inserted between these two groups of jobs.

Proof

We know that the cost function of a group of jobs is convex (Yano and Kim, 1991; James, 1992). To make the schedule feasible, three alternatives are available: produce one group earlier, produce one group later, and produce one group earlier and the other group later. The convex nature of the objective function will ensure that it will never decrease as the groups are moved further away from their current optimal position. If idle time is inserted then the objective function will never decrease as this will move at least one of the groups further from its optimal position. Therefore a necessary condition for the global optima is that there is no inserted idle time between these two groups. ■

This leads to a second theorem:

Theorem 4.2

The optimal schedule will consist of groups of jobs which are optimally scheduled and are not in conflict.

Proof

For the schedule to be feasible, no group is allowed to be in conflict with another group. Since no group is in conflict, then each group is independent of each other and can be treated as a sub-problem to be minimised. The global minimum must therefore consist of the schedule which has all of its groups optimally scheduled. ■

From Theorem 4.2, each of the conflicting groups must be combined and rescheduled as a single group, as evident in Theorem 4.3.

Theorem 4.3

If several optima exist for a group, no matter which optima is used to resolve the conflicts between groups, a global optima will be reached.

Proof

In cases where there is more than one global optima there may be a case where by using one optima the group stays independent of other groups, but when using another optima the group becomes in conflict with the another group. Using either of these points will produce a global optima, since producing the group just before or just after the colliding group will also be optimal due to the convex nature of the objective function when applied to the group and hence it could be treated as an independent group. ■

From Theorem 4.1, all jobs that are in conflict are placed together into a group of jobs without any idle time between them. This group of jobs can be optimally scheduled by moving the group of jobs so that at least one job is completed on its due date, as the convex nature of the objective function indicates that the minimum must occur when at least one job is on its due date (or its earliest possible schedule date). By calculating the value of the penalty function at each of the possible schedule positions the optima can be determined. The process is then repeated except this time groups of jobs are joined together if they are in conflict and then rescheduled. When there are no conflicts in the schedule, a global optimum has been reached.

An example of how this algorithm works is presented in Figure 4.3.

Iteration

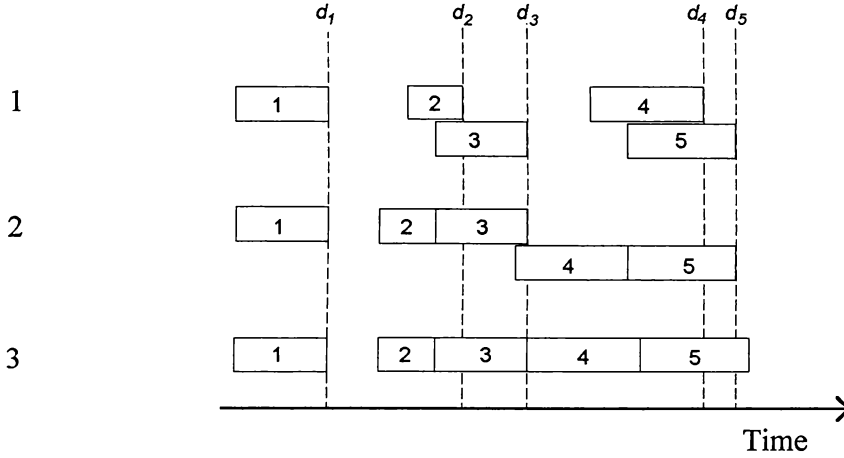


Figure 4-3. An example of the Idle Time Insertion Algorithm

The most time consuming part of this algorithm is finding the optimal starting position for the group once the groups have been merged. In its most basic form this will take $O(n^2)$ time, where n is the number of jobs, as each job has an alternative solution for which the cost of all the other jobs needs to be calculated. This can be reduced to $O(n \log_2 n)$ time by calculating the gradient of the cost function.

Let

$$\begin{aligned}
 c_i &= t + k_i \\
 f(t) &= \sum_{i=1}^n \left(\alpha_i (d_i - c_i)^+ + \beta_i (c_i - d_i)^+ \right) \\
 &= \sum_{i=1}^n \left(\alpha_i (d_i - k_i - t)^+ + \beta_i (k_i + t - d_i)^+ \right)
 \end{aligned}$$

where

k_i is a constant equal to $\sum_{k=1}^{i-1} p_k$

$f(t)$ = the total cost of the schedule when it begins at time t .

The gradient for the cost function is therefore defined as:

$$f'(t) = \sum_i \begin{cases} -\alpha_i & \text{if } (d_i > k_i + t) \\ \text{undefined} & \text{if } (d_i = k_i + t) \\ \beta_i & \text{if } (d_i < k_i + t) \end{cases}$$

The minimum point is found when the gradient is 0 or changes sign. The algorithm used to find the minimum point is shown in Figure 4-4.

1. Schedule all the jobs so that the starting time for the group is equal to 0.
2. Sort jobs in increasing order of $|d_j - c_j|^+$, late jobs are ignored.
3. Calculate the total net penalty, $(\alpha - \beta)$, for this initial schedule.
4. Repeat until the net penalty is less than or equal to 0, then exit
 - 4.1. Subtract the α penalty for the current job and add on its β penalty to the total net penalty.
5. The current job is to be completed at its due date, and the schedule is realigned to this date.

Figure 4-4. Group Scheduling Algorithm

As long as an efficient sorting routine is used in Step 2 of this algorithm, then the complexity of the algorithm is $O(n \log_2 n)$. The computational time is also reduced to the smaller number of calculations required at each iteration.

Comparisons between this idle time insertion algorithm and the Fry, Armstrong *et al.* (1987) algorithm, outlined in Section 7.3, found that the latter was on average over 8 times faster than the idle time heuristic developed in this section. This result is due to this new algorithm requiring that all the groups are rescheduled at each iteration whereas the Fry algorithm works from the latest 'groups' to the earliest 'groups'. The algorithm could be modified to work in a similar manner, however it is unlikely that this would make the idle time insertion process any more efficient than that proposed by Fry, Armstrong *et al.* (1987). The Fry algorithm was therefore used when idle time insertion routine was required, throughout this thesis.

4.4. Conclusions

Baker and Scudder (1990) have suggested that the distinct due date E/T scheduling problem be split into two different phases, a sequencing phase and an idle time insertion phase. Idle time insertion is an important issue for the distinct due date E/T scheduling problem, as pointed out by Yano and Kim (1991), search techniques solving E/T scheduling problems may require an idle time insertion routine be executed many times before a final solution is found, therefore the efficiency of the routine is critical to the overall speed of the search.

Several different idle time insertion algorithms have been developed in the literature. An iterative approach using the concept of 'conflict resolution' and 'groups' was also developed although preliminary tests showed that this approach was not as efficient as the approach taken by Fry, Armstrong *et al.* (1987). This new approach does, however, introduce the concept of breaking down the schedule into groups of jobs which can then be dealt with in turn.

5. Early/Tardy Solution Space

5.1. Introduction

In this Chapter we develop a new type of solution space for the distinct due date E/T scheduling problem which, instead of being based on the manipulation of a sequence of jobs, is based on selecting whether each job is to be scheduled before or after its due date. Section 5.2. develops the theory behind this new solution space, which is then outlined in Section 5.3. Sections 5.4. and 5.5. develop two different heuristics which can be used to implement this solution space. Section 5.6. discusses the advantages and disadvantages of each of the heuristics and conclusions are provided in Section 5.7.

5.2. Theoretical Development of the E/T Solution Space

In this section we develop theoretical results for the E/T scheduling problem which are derived from properties of the common due date, early/tardy, single machine scheduling problem. These results are then used to develop a heuristic which can be used to solve the E/T scheduling problem once each job is defined as being either early or late.

5.2.1. The Common Due Date Scheduling Problem

The single machine, common due date, early/tardy machine scheduling problem is much simpler than the distinct due date E/T scheduling problem because it involves only one due date, rather than a different due date for each job. Baker and Scudder (1990) outline four important properties of an optimal schedule for this simpler problem. Using the notation outlined in Section 1.6, these are:

1. No idle time is inserted.
2. The schedule is “V-shaped”; that is, early jobs are sequenced in non-increasing order of the ratio p_i/α_i and late jobs are sequenced in non-decreasing order of the ratio p_i/β_i
3. One job is completed on the due date.
4. The b^{th} job in the sequence is completed on the due date, where b is the smallest integer satisfying the inequality,

$$\sum_{j=1}^b (\alpha_j + \beta_j) \geq \sum_{j=1}^n \beta_j.$$

The common due date scheduling problem is solved by initially scheduling each job either early or late. The new schedule is then devised by using this early/late assignment for each job and the properties outlined above. The size of the solution space for this problem is effectively 2^n , as each job has only two alternatives - being scheduled early or being scheduled late. This compares favourably with the standard scheduling problem solution space size of $n!$ where every job has to be considered in every position in the sequence.

5.2.2. Distinct Due Date Problem

Do these four properties apply to the distinct due date scheduling problem? If these properties could be applied, in some form, to the distinct due date problem then the solution space could again be compressed from $n!$ to 2^n in a similar manner to the common due date problem, thus increasing the efficiency of the search. However Baker and Scudder (1990) state that none of the properties outlined for the common due date problem hold for the distinct due date problem as the optimal schedule may require idle time to be inserted between some jobs. Property 1 is therefore violated. The proof for Property 2 no longer holds as it cannot accommodate idle time or the use of different due dates. Properties 3 and 4 no longer hold, as multiple due dates cannot be accommodated. To make use of these properties within the distinct due date model it is necessary to change our definition of a *schedule*.

We now define a *schedule* to be one or more groups of jobs separated by idle time. A group is defined, in the same manner as Chapter 4, as a sequence of jobs with no idle time inserted between them. For example, there are two groups in Figure 4-1, jobs 4 and 1, and jobs 2 and 3. Although the four properties outlined by Baker and Scudder (1990) cannot be applied to the entire schedule, they can be applied to each group of jobs which make up the schedule. We now consider the four properties of an E/T schedule, for groups within the distinct due date scheduling problem.

Property 1. No idle time is inserted

Given the definition of a group of jobs, this property holds.

Property 2. The schedule is “V-shaped”

This property can be applied to only certain jobs within the schedule. Bector, Gupta and Gupta (1988) show that, for an optimal schedule, all jobs scheduled before the earliest due date of the jobs within a group must be sequenced in non-increasing order of the ratio p_i/α_i . They also show that all jobs scheduled after the latest due date of the group must be scheduled in non-decreasing order of the ratio p_i/β_i . Ow and Morton (1989) generalised this property further by proving that all adjacent pairs of jobs in the optimal schedule must satisfy the following condition:

$$\beta_i p_j - \Omega_{ij} (\beta_i + \alpha_i) \geq \beta_j p_i - \Omega_{ji} (\beta_j + \alpha_j) \quad (5-1)$$

where job i immediately precedes job j and Ω_{ij} and Ω_{ji} are defined as

$$\Omega_{xy} = \begin{cases} 0 & \text{if } s_x \leq 0 \\ s_x & \text{if } 0 < s_x < p_y \\ p_y & \text{otherwise} \end{cases}$$

where $s_x = d_x - t - p_x$.

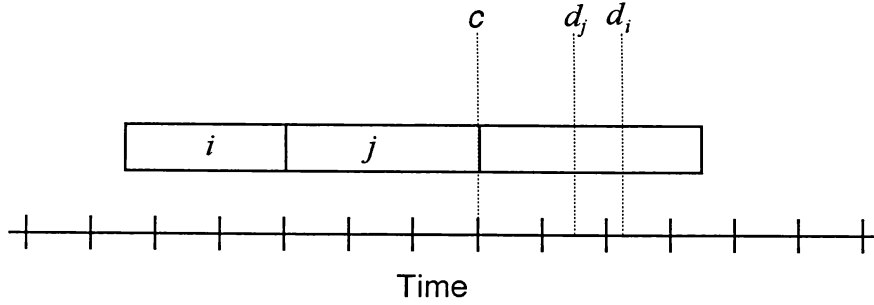


Figure 5-1. Determining the Order of Early Jobs

If we consider the situation where both jobs can be swapped without either job becoming late, as in Figure 5-1, equation 5-1 becomes

$$\beta_i p_j - p_j (\beta_i + \alpha_i) \geq \beta_j p_i - p_i (\beta_j + \alpha_j)$$

$$\frac{p_i}{\alpha_i} \geq \frac{p_j}{\alpha_j}$$

This is one of the requirements for a “V-shaped schedule”. The other requirement is derived by considering two late jobs which can both be swapped without either becoming early.

Property 3. One job is completed on the due date.

With the distinct due date model there is not one due date but up to n due dates. We can show, however, that for a given sequence of jobs at least one job within each group must be completed either on its due date or on its earliest completion time whichever is greater. This result is derived from the piecewise linear convex nature of the objective function for a group of jobs as shown by Yano and Kim (1991) and James (1992).

Let

G = A group of jobs,

E = The set of all jobs in the group G that are completed early,

T = The set of all jobs in the group G that are tardy,

$h_j = \sum_{k=1}^j p_k$, the earliest possible completion time for job j, and

$\varepsilon_j = |c_j - \max(d_j, h_j)|$; the deviation of the completion time of the job from either the due date or the earliest possible completion time, whichever is greater.

Theorem 5.1

If there exists a schedule S such that $\varepsilon_i > 0$, $\forall i \in G$, then there exists a schedule S' with an equivalent or lower cost with at least one $i \in G$ such that $\varepsilon'_i = 0$.

Proof

Let

$$\begin{aligned}\varepsilon_E &= \min_{i \in E}(\varepsilon_i) \\ \varepsilon_T &= \min_{i \in T}(\varepsilon_i)\end{aligned}$$

Consider rescheduling S so that it is completed ε_E units later. Then at least one job in the resulting schedule S' will have an i such that $\varepsilon'_i = 0$. The objective function will decrease by the combined decrease in earliness penalties from all early jobs but increase from the increased tardiness penalties from the tardy jobs. The overall reduction in the objective function is:

$$\sum_{i \in E} \alpha_i \varepsilon_E - \sum_{i \in T} \beta_i \varepsilon_E$$

If this movement is to result in an equivalent or improved value for the objective function then

$$\begin{aligned}\sum_{i \in E} \alpha_i \varepsilon_E &\geq \sum_{i \in T} \beta_i \varepsilon_E, \\ \Rightarrow \sum_{i \in E} \alpha_i &\geq \sum_{i \in T} \beta_i.\end{aligned}\tag{5-2}$$

Now consider scheduling S so that it starts completion ε_T units earlier. Then at least one job in the resulting schedule S' will have an i such that $\varepsilon'_i = 0$. Using the same argument as before we have the overall change in the objective function as:

$$\sum_{i \in T} \beta_i \varepsilon_T - \sum_{i \in E} \alpha_i \varepsilon_T.$$

Similarly, if this movement is to result in an equivalent or improved value for the objective function, we have

$$\begin{aligned}\sum_{i \in T} \beta_i \varepsilon_T &\geq \sum_{i \in E} \alpha_i \varepsilon_T, \\ \Rightarrow \sum_{i \in T} \beta_i &\geq \sum_{i \in E} \alpha_i.\end{aligned}\tag{5-3}$$

Either (5-2) or (5-3) must be true, therefore there must be a schedule S' with an equivalent or better value than the schedule S, but with at least one $\varepsilon_i = 0$. ■

Property 4. Determining the Best Job to be Scheduled

The method of determining the best job to be scheduled on its due date for the distinct due date problem is the same as that used for the common due date problem. This is again due to these problems having a cost function which, as it moves through time, is convex with no local minima (Yano and Kim, 1991; James, 1992).

5.3. The Distinct Due Date E/T Solution Space

While we have shown that the four properties outlined by Baker and Scudder (1990) in Section 5.2.1. can be applied successfully to groups of jobs within a schedule for the E/T scheduling problem, two factors remain unknown:

1. the group to which a particular job belongs, and
2. the precise order of the jobs in a group between the earliest due date and the latest due date.

Although the adjacency conditions of Ow and Morton developed some guidelines in the latter area, these are insufficient to fully define the sequence. No general-purpose rules exist to address these two remaining elements in any optimal way. However, heuristics can be used to fully define a schedule by specifying whether each job is early or late. In this context, we define a job as being *early* when it is completed on or before its due date, and a job as being *late* when it is completed on or after its due date.

With the four properties of Baker and Scudder (1990) redefined to be compatible with the E/T scheduling problem and an appropriate heuristic to address the remaining two problems, the E/T scheduling problem can be solved in a manner similar to the common due date problem. All that remains for a schedule to be defined is to determine whether each job is to be scheduled early or late. This definition of whether a job is to be early or tardy will be referred to as the *early/tardy job specification*. A schedule is therefore defined by an early/tardy job specification rather than the physical sequence of jobs.

This change in schedule definition can be taken one step further when viewed in a search heuristic context. Instead of using the physical sequence of jobs as the solution space through which a search heuristic moves, an abstract data space, based on early/tardy job specifications, can be used. This reduces the size of the solution space from $n!$ job sequences to 2^n early/tardy specifications, although this is still prohibitively large for problems of any practical size.

Figure 5-2 illustrates how the problem, outlined in Table 5-1, maps the abstract early/tardy solution space to the physical sequence data space. The early/tardy specification is represented by binary numbers, where a 0 in the i th position, numbered from *right to left*, implies that job i is early and a 1 means that job i is late. For example, the heuristic maps the point 0010, (ie. jobs 4, 3 and 1 are scheduled early and job 2 is scheduled late) onto the physical sequence 4123.

Job Number	Processing Time	Due Date	Early Penalty	Tardy Penalty
1	3	10	2	6
2	5	20	3	9
3	6	23	2	7
4	5	8	4	10

Table 5-1. Data for a Four Job Problem

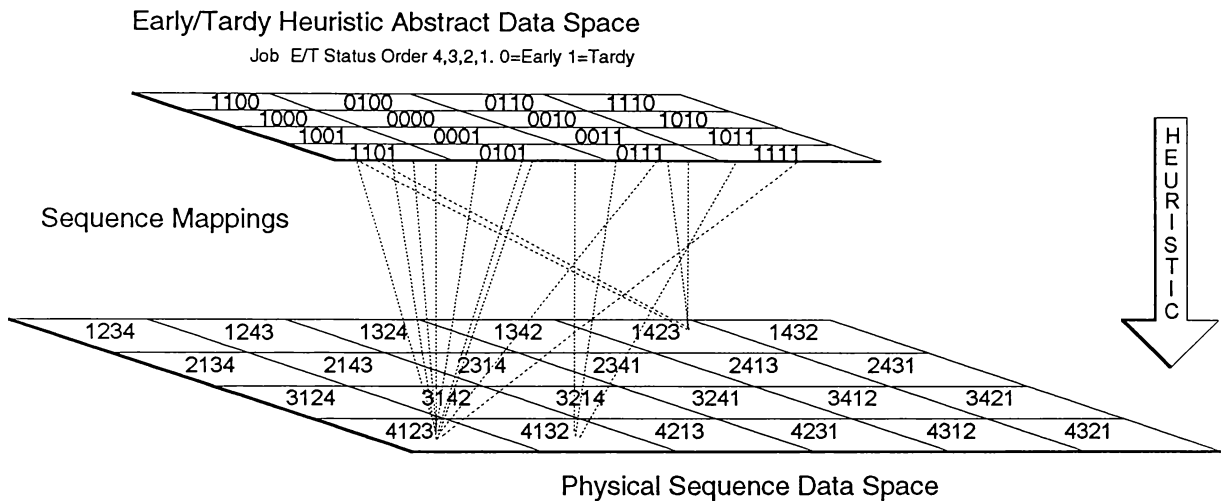


Figure 5-2. The Mapping between Solution Spaces

The entire early/tardy solution space (2^4 early/tardy specifications) for this example problem maps onto only three different job sequences. However within the same job sequence, several different schedules can be defined by changing the timing of the jobs so that the jobs adhere to the appropriate early/tardy specification. For example, the early/tardy specification of: 0000 and 1111 both map onto the physical sequence 4-1-2-3, however the timings for these jobs are different, as shown in Table 5-2.

Job	Early/Tardy Specification	
	0000	1111
4	7	8
1	10	11
2	18	20
3	24	26

Table 5-2. Completion Times of Two E/T Specifications which Map onto the same Sequence

The extent of sequence duplication is problem-dependent. Suppose the jobs within the schedule are spaced such that all jobs can be completed on their due date, then all points on

the early/tardy solution space map to the same physical sequence, regardless of whether the jobs were to be scheduled early or late. A more likely scenario occurs when a job that is scheduled on time is changed from early to tardy, or vice versa. The sequence of jobs does not need to change as both early/tardy job specifications are satisfied by the same schedule. Each job scheduled on time can halve the number of sequences which the heuristic maps onto. However the extent to which this type of duplication occurs will depend entirely on the heuristic that is used to complete the schedule.

Due to timing constraints, not all early/tardy job specifications will be valid and this will further reduce the size of the solution space, however the solution space will never be divided by invalid schedules.

Theorem 5.2

There are no divisions in the E/T solution space.

Proof

All schedules with one or more early jobs have a valid neighbour with one of its early jobs scheduled late. By repeating this same argument, ultimately all schedules degenerate to the schedule where all jobs are late, hence proving that there are no divisions in the E/T solution space.

The approach taken for this solution space is fundamentally different from other solution spaces that have been used in search techniques because it uses the unique structure of the problem being solved to construct the resulting schedule. However, unlike the neighbourhood scheme proposed by Storer, Wu and Vaccari (1992), this neighbourhood also has a solution space that is discrete and can be fully enumerated, although it may be very large when a large number of jobs are considered.

5.4. Five Point Insertion Heuristic

It was pointed out in Section 5.3. that to be able to create a schedule from an E/T job specification for the distinct due date problem two factors could not be determined with certainty. These two factors could, however, be addressed by the use of a heuristic. Two such heuristics were developed. The first heuristic, outlined in this section, uses a “constructive” heuristic approach, while the second heuristic, developed in Section 5.5., uses a “conflict resolution” approach.

The first heuristic, which is referred to as the Five Point Insertion Heuristic, defines a schedule from early/tardy job specifications for the distinct due date E/T scheduling problem using the following steps:

1. Separate the early jobs from the late jobs.

This step sets up the schedule so that all the early jobs and all the late jobs are examined in turn.

2. Insert all early jobs into the schedule.

The insertion of early jobs is done in ascending p_i/α_i order, placing each job in the schedule at the point with the lowest penalty for the job which does not interfere with already scheduled jobs. This ensures that the early jobs within the groups are in decreasing p_i/α_i order, as jobs with the highest priority are inserted first and jobs with the least priority are inserted last. This meets the “V-shaped” group requirement of Property 2. This insertion routine ensures that for each group at least one early job is completed at its due date or at its minimum completion time, satisfying Property 3. The insertion algorithm is outlined in Figure 5-3.

1. Find the position in the schedule for the job which is closest to and less than its due date.
2. Move back from this position until either some idle time is found or the position has gone past the earliest possible completion time for the job.
3. Calculate the completion time that will be used by looking at the position of the next job and setting the completion time of this job to the minimum of the starting time of the next job or the due date of this job.
4. Check that this time is valid in that inserting the job in this position will not violate the ready time constraints of previous jobs, if any.
5. Try all other idle time points for inserting the job in the schedule using the same procedure.
6. Choose the idle time point with the lowest cost and update the schedule, or if insertion is not possible classify the E/T job specification as being infeasible and exit.

Figure 5-3. Algorithm for the Insertion of Early Jobs

3. Insert all the late jobs

Insert each late job into the schedule in ascending p_i/β_i order. Inserting the highest priority jobs first gives priority to scheduling the more important jobs closer to their due dates, while less important jobs are more likely to be scheduled further away from their due dates. This again endeavours to satisfy the “V-shaped” requirement of Property 2.

To insert the late jobs into the schedule, five insertion points were identified as being the most common places where late jobs are found in good quality schedules. Each of these points directly addresses the questions of what group the jobs should belong to and the order the jobs. This is done by basing these decisions on the jobs that have already been inserted into the schedule and the due date and early/tardy penalties of the job being considered for insertion. Each insertion point always keeps at least one of the jobs within the group scheduled on its due date or its minimum completion time, maintaining Property 3. The five insertion points are described below and illustrated in Figure 5-4.

Jobs can be inserted:

1. At the end of the group that is scheduled over the due date of the late job being inserted or, if this is not possible due to interference from previously scheduled jobs, at the end of subsequent groups. This position causes the least disruption to the jobs already scheduled. If there is not enough room between the end of the job's appropriate group and the next group, then the job is scheduled at the end of the next group, which may be scheduled earlier to absorb the idle time between the two groups if this reduces the overall penalty. Figure 5-4(a) demonstrates the insertion of the late job 5 at the end of the groups within a schedule.
2. Directly after the job currently being processed on the late job's due date. This insertion point is used in the situation where the early jobs in the group incur high penalties if they are moved or, because of a late job that has been scheduled previously, the earlier jobs can not be scheduled earlier. Figure 5-4(b) illustrates this insertion process. We could not use this insertion point if job 4 in Figure 5-4(b) was originally scheduled on its due date, and was required to be scheduled early.
3. Before the job currently being processed on the job's due date. This insertion point is used when earlier jobs incur a high penalty if they are moved, while later jobs incur a lesser penalty. This point is only valid when the job that is currently scheduled on the due date is a late job or there is no job scheduled on the due date. Figure 5-4(c) illustrates the insertion of a late job into the example schedule.
4. On, or as close as possible to the due date, after the job being processed on the late job's due date. This position is used when the job scheduled on the late job's due date is to be scheduled early and the tardy penalty for the late job is high. Figure 5-4(d) illustrates the insertion of a late job into this position.
5. On, or as close as possible to the due date, before the job being processed on the late job's due date. This position is used when the job scheduled on the late job's due date is to be scheduled late and the tardy penalty for the late job is high. Figure 5-4(e) illustrates the insertion of a late job into this position.

Before evaluating each of the insertion points, we check if the early jobs which are in ascending p_i/α_i order are to be scheduled earlier because of the late job insertion and whether they can be swapped to correct their p_i/α_i order. If they can, we swap them before evaluating that insertion point.

The major difference between the approach of this heuristic and those proposed by Fry, Armstrong and Blackstone (1987), Yano and Kim (1991) and Baker and Scudder (1990) is that the sequencing and the scheduling components are carried out simultaneously rather than having both a sequencing phase and an idle time insertion or scheduling phase.

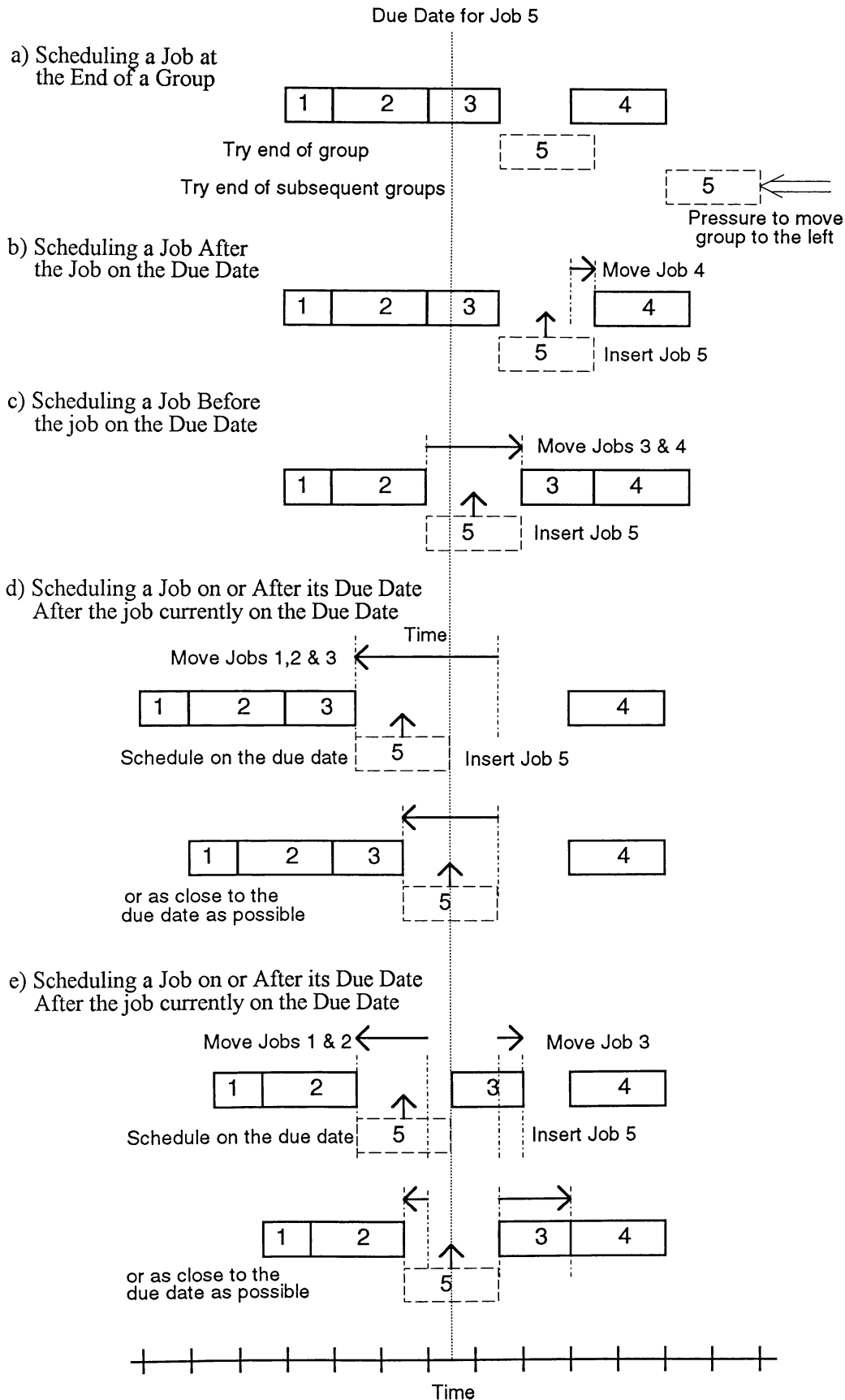


Figure 5-4. Insertion of Late Jobs Into the Schedule

5.5. Dynamic Loop Heuristic

A second E/T scheduling heuristic, called the ‘Dynamic Loop’ heuristic, is based on iteratively calculating a ‘critical point’ (or ‘critical time’) for each job in the schedule whereby if a job violates its critical point, a change in the sequence of jobs is required to lower the schedule’s cost. A critical point is violated when an early job is scheduled before its critical point, or a late job is scheduled after its critical point. Different types of moves are associated with the critical points, and each type of move is given a different level of importance allowing the most important critical point violation to be resolved first. This process continues until no critical points are violated, and hence the schedule reaches an ‘equilibrium state’.

5.5.1. Detailed Description of the Dynamic Loop Heuristic

Figure 5-5 outlines the stages which are used to implement a scheduling system based on the Dynamic Loop concept. At each of these stages when either an early job is moved back, or a late job is moved forward, the dynamic nature of this system comes into play and updates the sequence of jobs as needed.

1. Determine a schedule for all the early jobs without any regard for the late jobs by using a heuristic.
2. Determine a schedule for all the late jobs without any regard for the early jobs by using a heuristic.
3. Merge each of the schedules together.

Figure 5-5. The Top Level of the Dynamic Loop Heuristic

The first stage in the Dynamic Loop heuristic is to produce a schedule which is only feasible with regard to early jobs. This is done by using the heuristic outlined in Figure 5-6.

- 1.1. Place Jobs in Earliest Due Date Order.
- 1.2. Calculate the completion times by starting at the last early job in the schedule and progressively setting the completion time at the minimum of the job’s due date or the latest available time before the due date.
- 1.3. Dynamically adjust the early jobs for changes to the job sequence until an equilibrium is reached.

Figure 5-6. Stage 1 of the Dynamic Loop Heuristic - Early Job Scheduling

The second stage is similar to the first except that it deals with late jobs rather than early jobs. The heuristic to schedule late jobs is shown in Figure 5-7.

- 2.1. Place Jobs in Earliest Due Date Order.
- 2.2. Calculate the completion times by starting at the first late job in the schedule and progressively setting the completion time at the maximum of the job’s due date or the earliest available time after the due date.
- 2.3. Dynamically adjust the late jobs for changes to the job sequence until an equilibrium is reached.

Figure 5-7. Stage 2 of the Dynamic Loop Heuristic - Late Job Scheduling

The final stage merges the early schedule and the late schedule produced by the two previous stages until the resulting schedule contains no further conflicts. This is done by resolving the conflicts between the early job schedule and the late job schedule starting with the earliest jobs and working through to the latest jobs. In this way it is always true that all jobs scheduled before the 'current job' are never in conflict. This procedure is outlined in Figure 5-8.

- 3.1. Define the current late job as being the first job in the schedule of late jobs.
- 3.2. Determine which group of early jobs (if any) is in conflict with the current late job and the amount of overlap between the late job and the early group.
- 3.3. Calculate the amount of room that is available to move the early jobs back without scheduling jobs before they are ready to be produced.
- 3.4. Calculate the smallest movement of both the early and late groups before one of their jobs will violate a critical point or before they will collide with another group.
- 3.5. If the current late job has no conflict with any early job in the schedule then proceed to the next late job until finished.
- 3.6. Calculate the cost of moving the early group back and the cost of moving the late group forward using the heuristics outlined below.
- 3.7. If the cost of moving the late jobs back is less than the cost of moving the early jobs forward then move the late jobs back by the minimum amount and repeat from step 3.2. otherwise move the early jobs back by the minimum amount
- 3.8. If there is a conflict between the early jobs and the late jobs that had been previously scheduled, then try to move them both back. If this is not possible, try moving the necessary number of early jobs over the conflicting late job, if this is cost effective. The cost of this movement is determined by the heuristic outlined below.
- 3.9. Repeat the entire process using the same job until it is no longer conflicting with early jobs.

Figure 5-8. Stage 3 of the Dynamic Loop Heuristic - Merging Early and Late Schedules

To obtain comparable costs at step 3.6 the cost of the entire conflict resolution needs to be calculated even though conflict may only be partially resolved by the move. To accurately determining the cost of the schedule would require solving the schedule for each particular option of moving a group of early jobs back or a group of late jobs forward. This would clearly require excessive computational time and effort hence heuristics are used to determine the cost of the schedule. Two heuristics are used, one determines the short range costs of moving early jobs while the second estimates the long term costs. These heuristics are detailed in Figure 5-9.

Short Range Early Cost Heuristic

- Calculate the penalty incurred when pushing the early jobs back to the first critical point, call this *push_ep*.
- Calculate the size of idle time to be inserted due to the movement at this critical point, call this *idle_size*.
- Calculate the early penalty rate that is incurred while the idle time inserted at this point is absorbed, call this *idle_ep*.

Let

movement = the total movement required to resolve the conflict.

Then

$$\text{short_range_cost} = (\text{movement} - \text{idle_size}) * \text{push_ep} + \text{idle_size} * \text{idle_ep}$$

Long Range Early Cost Heuristic

For early jobs assume that only the last job in the early group may be moved and that all other jobs will remain in the same order. It is therefore calculated by determining the sequence once the first job is moved and then the cost of moving each of the jobs in this new sequence back by the relevant amount, taking into account the absorption of idle time.

Long Range Late Cost Heuristic

In a similar manner to the Long Range Early Cost Heuristic, assume that only the first job in the late group may be moved and that all other jobs will remain in the same order. The cost is again calculated in a similar manner.

Figure 5-9. Heuristics for Calculating Schedule Costs at Step 3.6

To determine if it is worthwhile moving early jobs over a conflicting late job in step 3.8, the heuristic outlined in Figure 5-10 is used. This heuristic is based on calculating the overall penalty of moving the early jobs back in the schedule.

- Calculate the overall penalty for moving early jobs back by the required amount. For further late jobs which are in the schedule estimate their cost in the same way as early jobs - even though their cost may decrease due to a movement back, however it may also increase if further jobs need to be moved over the top.
- Calculate the cost of moving the early job over the late jobs immediately before it in the schedule.
- Calculate the cost of moving late jobs back due to insufficient room earlier in the schedule.
- Sum all previous costs to obtain an estimate of the total cost of moving an early job back.

Figure 5-10. Heuristic to Estimate the cost of Moving an Early Job Over Late Jobs

5.5.2. Implementation of the Dynamic Loop

All of the top level steps involve the use of a dynamic loop to determine the sequence of jobs within the schedule and to fine tune the timings for the jobs. This keeps modifying the schedule until an equilibrium schedule is reached, which is a state where no further movement can be made to reduce the overall cost of the schedule.

The procedures that make up the dynamic loop can be broken into two different types, those that deal with early jobs and those that deal with late jobs. These procedures are almost an exact mirror image of each other with the exception that ready time violations need to be considered for early jobs, while late jobs have no equivalent constraint.

The functions that implement the dynamic loop move either the early jobs back or the late jobs forward. If no movement is required then a movement of zero can be given to initiate the loop.

The two main functions required in this process involve moving early jobs back given the last job in the group and to move the late jobs forward given the first late job in the group. The algorithm for moving early jobs back is shown in Figure 5-11. The procedure for moving late jobs forward is implemented in an analogous manner.

1. Search back in the schedule until the start of the group is found, i.e. slack is found in the schedule between the early jobs, without considering late jobs.
2. Define the maximum possible movement as the minimum of the amount requested or the amount to absorb the slack to the next early or late job in the schedule.
3. Calculate and set the new times for each of the jobs.
4. Dynamically update the schedule until an equilibrium is found by:
 - 4.1. Finding all the out of sequence jobs, i.e. those which are not in WLPT order in the schedule.
 - 4.2. Calculate the Critical points using this out of sequence job information
 - 4.3. For the critical point which is the most important, implement the move that is required.
 - 4.4. Check for possible improvement by late jobs because of idle time insertion. Update late job times as required.
 - 4.5. Update early job times.
 - 4.6. Repeat this sub-process until no more critical points are violated.

Figure 5-11. Algorithm to Move Early Jobs Back

5.5.3. Critical Point Calculations

The calculation of 'critical points' enables the dynamic loop to decide which moves to perform so that a better quality solution can be obtained. The four different critical points which need to be considered are outlined below.

1. Swap Two Adjacent Jobs

Swap one job with an adjacent job checking the costs of other jobs taking into account the elimination of idle time. Figure 5-12 illustrates how idle time is absorbed then jobs i and j are swapped.

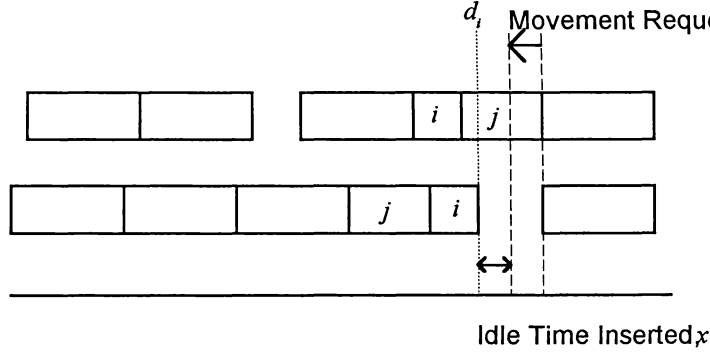


Figure 5-12. Swap Job Critical Point

The amount of idle time that can be inserted into the schedule needs to be calculated to determine this critical point.

Let

x = the amount of idle time being inserted

k_0 = the group of jobs directly behind job i in the schedule back to the start of the group.

k_1 = the first group of jobs behind the group being moved, and

k_l = the l th group behind the group being moved.

I_l = the amount of idle time between groups k_l and k_{l-1} . $I_0 = 0$

N = the number of groups behind the group being moved which will be affected by the insertion of idle time, ie. $\sum I_l < x$

If jobs i and j are swapped then the extra cost would be

extra cost of job $i = \alpha_i(x - p_j)$

extra cost of job $j = \alpha_j(x + p_i)$

extra cost for all other later jobs = $\sum_{m=0}^N \left[\sum_{l \in k_m} \alpha_l \left(x - \sum_{p=0}^m I_p \right) \right]$

Therefore the critical point will occur when the extra costs are 0, hence overall the equation becomes:

$$\begin{aligned} \alpha_j(x + p_i) + \alpha_i(x - p_j) + \sum_{m=0}^N \left[\sum_{l \in k_m} \alpha_l \left(x - \sum_{p=0}^m I_p \right) \right] &= 0 \\ \alpha_j(x + p_i) + \alpha_i(x - p_j) + \sum_{m=0}^N \left[\sum_{l \in k_m} \alpha_l x \right] - \sum_{m=0}^N \left[\sum_{l \in k_m} \left(\alpha_l \sum_{p=0}^m I_p \right) \right] &= 0 \\ x \left(\alpha_j + \alpha_i + \sum_{m=0}^N \sum_{l \in k_m} \alpha_l \right) &= \left(\alpha_i p_j - \alpha_j p_i + \sum_{m=0}^N \left[\sum_{l \in k_m} \left(\alpha_l \sum_{p=0}^m I_p \right) \right] \right) \end{aligned}$$

giving

$$x = \frac{\left(\alpha_i p_j - \alpha_j p_i + \sum_{m=0}^N \left[\sum_{l \in k_m} \left(\alpha_l \sum_{p=0}^m I_p \right) \right] \right)}{\left(\alpha_j + \alpha_i + \sum_{m=0}^N \sum_{l \in k_m} \alpha_l \right)}$$

From this, the critical point for job j can be calculated as

$$cp_j = d_i + x$$

A check is included in the idle time calculation to ensure that the movement requested is actually valid for each of the jobs involved. If there is insufficient slack due to limited idle time then this critical point is not calculated.

2. Swap Two Jobs

Swap two jobs over an out of sequence job as detailed in Figure 5-13, where jobs i and j are swapped over an out of sequence job h . Idle time is inserted at the appropriate places, depending on the restrictions placed on the schedule.

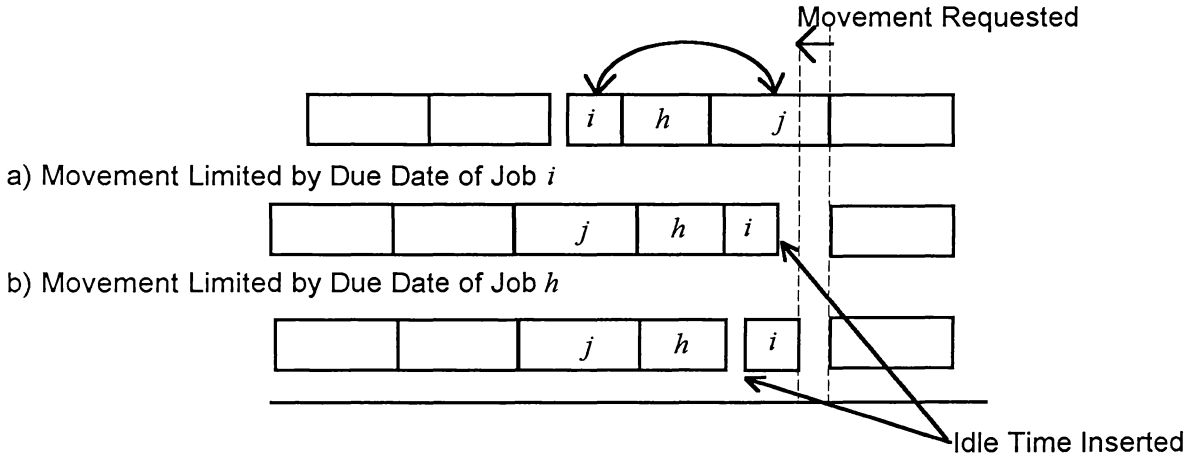


Figure 5-13. Critical Point for Swap with Out of Sequence Job

There are two different scenarios for this type of move, one where the move is going to be limited by the due date of job i and the other where the move is going to be limited by the due date of job h .

Scenario A: Movement limited by the due date of job i .

$$\text{extra cost of job } i = \alpha_i(x - p_h - p_j)$$

$$\text{extra cost of job } j = \alpha_j(x + p_i + p_h)$$

$$\text{extra cost of job } h = \alpha_h(x + p_i - p_j)$$

$$\text{extra cost for all other later jobs} = \sum_{m=0}^N \left[\sum_{l \in k_m} \alpha_l \left(x - \sum_{p=0}^m I_p \right) \right]$$

Note that the only differences between these equations and those involved in swapping adjacent jobs is the inclusion of the p_h factor and an extra cost for the movement of job h , hence the value for x is

$$x = \frac{\left(\alpha_i(p_j + p_h) + \alpha_h(p_j - p_i) - \alpha_j(p_i + p_h) + \sum_{m=0}^N \left[\sum_{l \in k_m} \left(\alpha_l \sum_{p=0}^m I_p \right) \right] \right)}{\left(\alpha_j + \alpha_i + \alpha_h + \sum_{m=0}^N \sum_{l \in k_m} \alpha_l \right)}$$

and the critical point for job j becomes

$$cp_j = d_i + x$$

Scenario B: Movement limited by due date of job h .

$$\text{extra cost of job } i = \alpha_i(x - p_h - p_j)$$

$$\text{extra cost of job } j = \alpha_j(x + p_i + p_h)$$

$$\text{extra cost for all other later jobs} = \sum_{m=0}^N \left[\sum_{l \in k_m} \alpha_l \left(x - \sum_{p=0}^m I_p \right) \right]$$

Hence the only difference in the formulation for this scenario and the previous scenario is that the cost for job i no longer includes idle time, hence the idle time calculation becomes

$$x = \frac{\left(\alpha_i(p_j + p_h) + \alpha_h(p_j - p_i) - \alpha_j(p_i + p_h) + \sum_{m=0}^N \left[\sum_{l \in k_m} \left(\alpha_l \sum_{p=0}^m I_p \right) \right] \right)}{\left(\alpha_j + \alpha_h + \sum_{m=0}^N \sum_{l \in k_m} \alpha_l \right)}$$

but the critical point for job j changes to

$$cp_j = d_h + x + p_i$$

3. Move a Job

Swap an in-sequence job over out-of-sequence jobs, as outlined in Figure 5-14, where job i is moved over the out of sequence jobs into its right place, and in the process inserting idle time into the schedule.

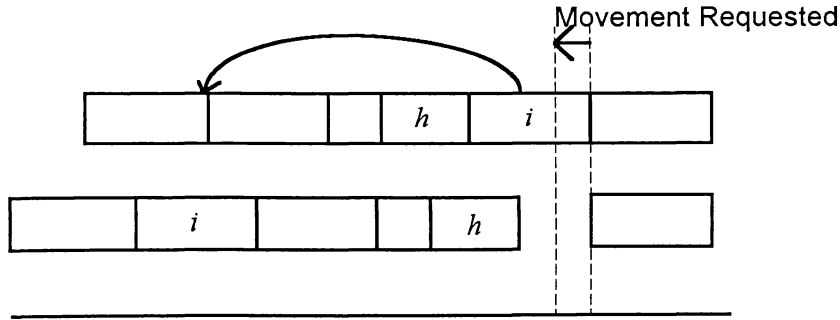


Figure 5-14. Critical Point to Swap Over Out of Sequence Jobs

We define j to be the position where job i is to be inserted, and R to be the group which j is part of.

$$\text{extra cost of job } i = \alpha_i(\sum p_l + x)$$

$$\text{extra cost of job } j = \sum \alpha_l(x - p_i - \sum I_p) \quad \text{where } l \in k_0 \text{ and } p < R$$

$$\text{extra cost for all other later jobs} = \sum_{m=1}^N \left[\sum_{l \in k_m} \alpha_l \left(x - \sum_{p=0}^m I_p \right) \right]$$

By adding all these costs together and rearranging the formula, we can calculate the amount of idle time that can be absorbed, hence

$$x = \frac{\left(\sum_{l \in k_0} \alpha_l p_i - \alpha_i \left(\sum_{l=j}^{i-1} p_l + \sum_{m=0}^R I_p \right) + \sum_{m=R+1}^N \left[\sum_{l \in k_m} \left(\alpha_l \sum_{p=0}^m I_p \right) \right] \right)}{\left(\alpha_i + \sum_{m=0}^N \sum_{l \in k_m} \alpha_l \right)}$$

The critical point for job i becomes

$$cp_j = d_h + x$$

4. Swap Two Adjacent Jobs to Absorb Idle Time

Absorb idle time in the schedule by swapping jobs, as shown in Figure 5-15 where jobs i and j are swapped to absorb the slack s .

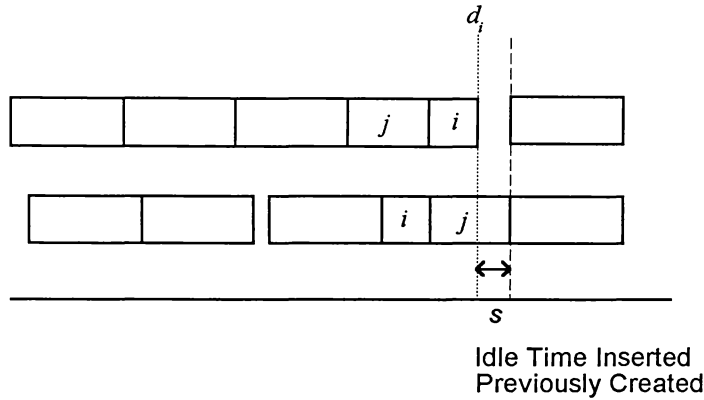


Figure 5-15. Critical Point to Absorb Idle Time

For this critical point, there is no insertion of idle time, just an absorption of idle time that is presently in the schedule. There is therefore no calculation of a critical point, just a calculation of the extra cost of swapping the jobs. If this cost is negative, ie. there is a cost reduction, then the critical point for job i is set to the new completion time for job j causing the swap to be done immediately.

The cost calculation is:

$$\text{cost} = \alpha_i(p_j - x) - \alpha_j(p_i - x) - x \sum_{l \in k_0} \alpha_l$$

where

s = the amount of slack available.

$$x = \min(d_j, c_i + s) - c_i$$

The priority given to each of these critical points in order of the most important to the least important are:

1. Idle time absorption moves, with the last one found executed first.
2. Skip Moves, in the order of the lowest α ratio of job i to the highest.
3. Adjacent swaps which do not insert any idle time into the schedule, with these in order from lowest alpha ratio to highest alpha ratio.

4. Skip swaps which do not insert any idle time into the schedule, with these in order from lowest alpha ratio to highest alpha ratio.
5. Adjacent swaps which insert idle time into the schedule, with these in order from highest alpha ratio to lowest alpha ratio.
6. Skip swaps which insert idle time into the schedule, with these in order from highest alpha ratio to lowest alpha ratio.

When idle time is involved, we want the most important jobs dealt with first, as idle time needs to be minimised within the schedule. However when no idle time is involved, low priority jobs are dealt with first, as higher priority jobs will tend to dominate the lower priority job moves and hence not enable the movement of a lower priority job out of their current position in the schedule causing local minima.

5.6. Evaluation of the Heuristics

Table 5-3 outlines the advantages and disadvantages of the two different E/T scheduling heuristics proposed in this chapter.

Heuristic	Advantages	Disadvantages
Five Point Insertion	Has a low polynomial relationship between the number of the jobs and the amount of time required.	The heuristic tries to find the optimal solution at each stage of the insertion process. The heuristic is limited to just 5 different insertion points and that late jobs must be inserted in WSPT order.
	Accurately evaluates the entire schedule when considering scheduling options without the need for heuristics.	The technique is myopic in its view of the schedule as it only considers scheduled jobs ignoring other unscheduled late jobs.

Table 5-3. Advantages and Disadvantages of the two E/T Heuristics

Heuristic	Advantages	Disadvantages
Dynamic Loop	Conceptually the Dynamic Loop heuristic has the potential to obtain all optimal solutions. Its accuracy is limited by the costing heuristics that are used throughout the technique and the priority given to each of the critical points. The heuristic has a holistic view of scheduling the jobs, considering all jobs at the same time rather than inserting one job at a time.	Due to the dynamic nature of the technique considers and reconsiders different jobs many times causing the computational time required for merging groups of jobs to be large. The evaluation of the different move options requires many calculations hence longer computational time. This heuristic has a myopic focus when evaluating the costs of each move. To overcome this deficiency, long range cost estimation of the effect of pushing early jobs back or late jobs forward has been incorporated but the amount of time that would be needed to accurately determine these moves is prohibitive. These heuristics therefore speed up the calculation at the expense of solution accuracy. The computational time of the heuristic will tend to be exponential/high polynomial with regard to the number of jobs being scheduled, making the technique unsuitable for large problems. The accuracy of this technique is less than that obtained by the five point insertion method, refer to Section 7.3 where computational comparisons of the two techniques are performed.

Table 5-3. Advantages and Disadvantages of the two E/T Heuristics cont..

We can summarise the two main features of the heuristics by comparing the way each heuristic schedules jobs and how accurately each heuristic determines the costs when doing the scheduling. This is summarised in a diagrammatic form in Figure 5-16.

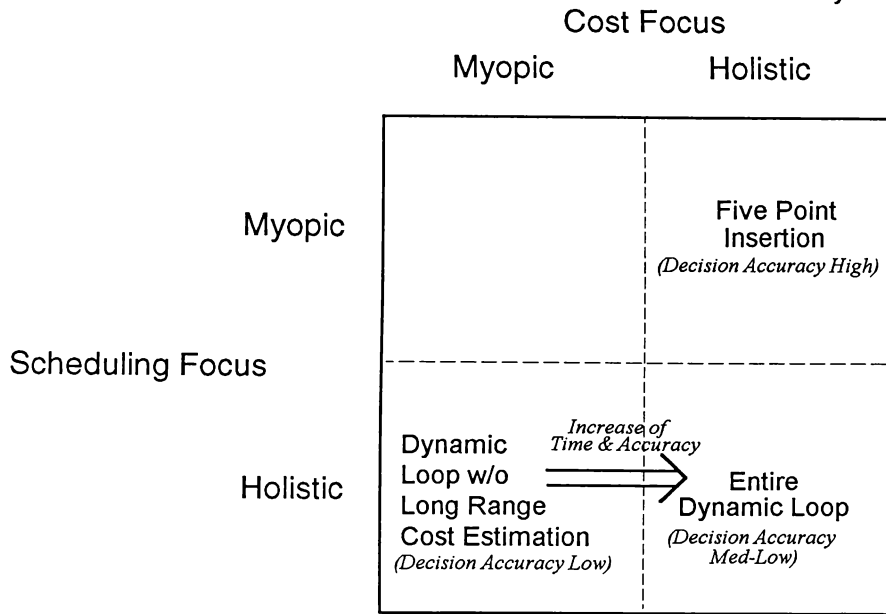


Figure 5-16. Comparison of E/T Scheduling Heuristics

The Dynamic Loop heuristic starts with an invalid schedule and incrementally resolves the conflicts to obtain its solution. This approach always considers every job in the schedule at each iteration however, to speed up the scheduling process, cost estimates are used to make decisions on resolving conflicts between jobs. The accuracy of the decisions that are made from this cost information, called decision accuracy, is therefore low. In the Dynamic Loop implemented in this chapter, long range estimates of costs were included to help increase the accuracy of these decisions. This produced decisions that were better but not highly accurate, hence this type of dynamic loop was considered to have a Medium-Low decision accuracy. On the other hand the decision accuracy for the Five Point Insertion heuristic is high, as at each step, up to five different schedules are constructed and evaluated accurately, however these schedules only include the early jobs and some of the late jobs. Therefore the scheduling focus is myopic.

From the computational experiments, outlined in Section 7.3, it was found that the Five Point Insertion heuristic performs better than the Dynamic Loop heuristic. This indicates that the cost focus emphasised in the Five Point insertion technique is more important than the scheduling focus that is emphasised in the Dynamic Loop heuristic.

5.7. Conclusions

In this chapter the properties of the common due date E/T scheduling problem were developed further and applied to the distinct due date problem E/T scheduling problem. From these properties and with the use of an appropriate heuristic, a schedule can be fully defined by specifying whether each job is to be scheduled early or late. A solution space can therefore be developed using early/tardy job specifications which can be searched using any appropriate search technique. Two different heuristics were developed, one based on late job insertions, the other based on dynamically resolving conflicts between early and late jobs within the schedule. The first of these techniques was highly accurate on calculating the cost of the schedule, whereas the latter was more concerned with taking a holistic approach to scheduling the jobs. Heuristics were then categorised by two different factors, the accuracy of their cost estimation and the number of jobs considered concurrently when scheduling jobs. When computational experiments were carried out, in Section 7.3, it was found that the heuristic with a more accurate cost estimation produced better results than the heuristic which considered all the jobs at once, highlighting the importance of determining the cost accurately when making scheduling decisions.

The development of this type of solution space has not been documented in any of the public literature to date to solve the distinct due date E/T scheduling problem. The generalisation of the common due date properties to distinct due date problems provides a major step forward in the solution of this class of problem.

6. Search Technique Implementation

6.1. Overview

This chapter describes the implementation of the three main search techniques for the generalised E/T scheduling problem, simulated annealing (SA) (Section 6.2.), genetic algorithms (GAs) (Section 6.3.) and finally tabu search (TS) (Section 6.4.). Section 6.5. outlines the implementation of tabu search techniques specifically designed for common due date problems. These common due date problems will be used to compare the performance of an E/T neighbourhood scheme with a standard neighbourhood scheme. All the techniques are summarised in a table in Section 6.6. They are then tested in computational experiments which are reported in Chapter 7.

6.2. Simulated Annealing

6.2.1. Standard Simulated Annealing

The standard SA search method uses a solution space based on sequences of jobs. A hybrid neighbourhood scheme is used, which will select a neighbour from either the swap or insert neighbourhood schemes with equal probability. The sequence of jobs is transformed into a schedule by the idle time insertion algorithm of Fry, Armstrong and Blackstone (1987). The initial starting point for the search is generated by sequencing the jobs in earliest due date (EDD) order and then optimally inserting idle time.

The cooling schedule developed by Lundy and Mees (1986) is used, whereby at each iteration the temperature is reduced. A ‘standard’ initial temperature was calculated as:

$$temp_0 = \frac{n((n-1)\bar{p})^{\frac{\bar{\alpha}+\bar{\beta}}{2}}}{2}.$$

This is an estimate of the upper bound on the depth of local minima within the solution space. The actual initial temperature multiplies this value by a parameter, enabling the effect of starting temperature on the final solution quality to be measured. The temperature was reduced by the formula:

$$temp_{i+1} = \frac{temp_i}{(1 + \Delta temp_i)},$$

where Δ is defined as

$$\Delta = \frac{x}{temp_0},$$

and x is a parameter controlling the decrement rate.

The final temperature is not set explicitly but determined by the amount of time that is allowed to find a solution. This allows for fair comparisons between techniques.

The standard SA search method uses the extension which stores the best solution found so far. No other extensions are implemented, so as to provide a valid comparison of all the techniques in their most basic form.

A second standard SA method is also implemented which, instead of starting at the EDD sequence, starts at either the E/T approximation to the EDD sequence or the EDD sequence with idle time optimally inserted, whichever has the lowest cost. This provides a comparison on the effects of starting points for SA searches.

6.2.2. E/T Simulated Annealing

The E/T simulated annealing search techniques use the early/tardy job specification solution space with the five point insertion heuristic to create the physical schedule. A neighbour is therefore defined as any schedule that has the early/tardy specification of one of the jobs changed, that is an early job changes to a late job, or vice versa. Since some of these neighbours may produce invalid schedules these neighbours are ignored. The initial starting point is an approximation of the E/T heuristic to the EDD schedule with idle time optimally inserted. This is found by sequencing the jobs in EDD order and inserting idle time then each of the early/tardy job specifications is then set according to whether the job is early or late. The E/T heuristic is then used to determine the starting point for the search using this E/T job specification.

This method also uses the cooling schedule adopted by Lundy and Mees (1986) which decreases the temperature after each iteration, hence the amount of time spent at each temperature is not an issue. In the same manner as the standard SA method, the initial temperature and the cooling rate are both determined by parameters which enable a comparison of the effects of these variables on the quality of the search results. Again, the final temperature is not determined explicitly, rather the search is terminated after a given amount of time.

As with the standard SA, the only other extension implemented is to store the best solution found so far, as this maintains comparability between this technique and the standard SA method.

6.2.3. Early/Tardy Smart Simulated Annealing

The E/T Smart SA is identical to the E/T SA except that it uses a slightly different E/T neighbourhood scheme.

The 'smart' neighbourhood scheme attempts to adjust the schedule being analysed so that large changes in the physical schedule do not occur when a job is changed from being early to being late, or vice versa.

Recall how a sequence is optimally scheduled. The optimal position occurs when the following two conditions are present:

$$\sum_{i \in E} \alpha_i + \alpha_k \geq \sum_{i \in T} \beta_i$$

and

$$\sum_{i \in E} \alpha_i \leq \sum_{i \in T} \beta_i + \beta_k$$

where job k is the job that is scheduled on its due date.

By changing just one job, as proposed in the E/T SA method, the balance of the group is clearly disrupted. This imbalance can cause large changes to the physical schedule, especially to those jobs that have been changed are on the outskirts of the group. Clearly what is needed is a technique that will not disrupt the balance to same extent and will allow slight modifications to the schedule. The ‘smart’ E/T neighbourhood scheme tries to overcome this problem by maintaining the same number of early and tardy jobs within the group. When one job is changed from early to late (or late to early) then another job that is already late (early) is also changed within the same group, so that it becomes early (late) to maintain this balance. Since diversity is also required in any search, just changing one job is also part of this neighbourhood scheme.

6.3. Genetic Algorithms

6.3.1. Standard Genetic Algorithm

The standard GA is based on the general algorithms outlined by Goldberg (1989) and Lawton (1992). To speed up the process of evaluating large size scheduling problems, the ‘steady-state reproduction without duplicates’ extension with a new off-spring size of 1 is also implemented. This extension also provides a comparable iteration counter to those in the other search techniques. It uses a solution space based on sequences of jobs while the idle time insertion algorithm of Fry, Armstrong *et al.* (1987) is used to complete the schedule.

The initial population has one chromosome set as the EDD sequence, one chromosome set as the E/T approximation to the EDD sequence, and the rest of the population is generated randomly. Checks are made to ensure that every chromosome in the population is unique, with those that are not unique being mutated until they are. The population size is controlled by a parameter so that the effect of population size on the solution quality can be measured. Tests were also performed on GAs whose population was completely random to measure the effect of good population members on solution quality.

The fitness function for a particular chromosome in the population is the schedule cost of the worst chromosome's in the population less the chromosome's schedule cost. Thus chromosomes with the least cost have the greatest chance of being selected. Sigma truncation (Goldberg, 1989) is also used so that all chromosomes that have a cost which is worse than two standard deviations from the mean are excluded from the selection process. Once the chromosomes have been evaluated, standard roulette wheel parent selection is used on those chromosomes available for selection.

A form of elitism was also used whereby the new chromosome will replace the worst chromosomes in the population if, and only if, they have a cost lower than the worst chromosome in the population. When a chromosome created by crossover is the same as one already in the population, that chromosome was mutated until it was unique, hence no duplicates can exist within the population.

Four different crossover operators are used: Partially Matched Crossover (PMX), Order Crossover (OX), Cycle Crossover (CX), and Uniform Order-Based Crossover (UOBX). When two chromosomes are selected they will be crossed with one of these four crossover operators with equal probability.

Mutation is carried out by using an order-based mutation scheme. This was shown to be the best type of mutation by Syswerda (1991). This result is also supported by Barnes and Laguna (1991a) who found that the 'insert' neighbourhood scheme, which is effectively the same as order-based mutation, provided superior solutions to the 'swap' neighbourhood scheme. This later scheme is effectively the same, in principle, to position-based mutation.

The termination criteria was based on finishing after a given amount of time, so comparisons could be carried out between different search techniques.

6.3.2. E/T Genetic Algorithm

The E/T GA search method is similar to the standard GA search method, with the major difference being in the use of a different chromosome representation. The chromosome representation used replicates the early/tardy job specification by using a binary coding method. Here a "0" indicates that a job is to be scheduled early and a "1" if the job is to be scheduled late. The E/T GA method implements a steady-state reproduction scheme, however duplicates are allowed. This modification was necessary because the size of the valid solution space, that is the size of the solution space which the heuristic can solve, is unknown and it may not be possible for the population to be filled with unique valid schedules, as a schedule is guaranteed to have only one valid member.

The initial population has the E/T approximation to the EDD sequence placed in it, while the rest of the population is randomly generated. If a randomly generated early/tardy specification is found to be invalid, then a new random sequence will be chosen but with a heavier bias towards making the job late rather than early. This bias of early : tardy selection increases from the original 1:1 chance, to a 1:2 chance, to a 1:3 chance and so on, until a valid schedule is found. Tests were also performed on E/T GAs whose population was completely random to measure the effect of good population members on solution quality.

Crossovers are carried out by using a uniform crossover, whereby each of the children has an equal chance of getting the early/tardy setting from each of their parents. This scheme was adopted since there is essentially no relationship between the ordering of jobs within the chromosome and how they are scheduled, a feature consistent with uniform crossover. A standard mutation operator is used, which changes one of the genes in the chromosome from a 0 to a 1, or vice versa, effectively changing one job's early/tardy status. If the new schedule is invalid the mutations will continue until a valid schedule is found.

The fitness function for a particular chromosome in the population is the schedule cost of the worst chromosome in the population less the chromosome's schedule cost, again giving lower cost schedules a higher chance for reproduction. As for the standard GA, sigma truncation is again used to eliminate all chromosomes which are worse than two standard deviations from the mean from the selection process. This allows for a form of elitism and ensures that those chromosomes which are involved in the crossover do not have extremely bad schedules. Once the schedules are evaluated, a standard roulette wheel parent selection is used.

In the same manner as for the standard GA, a form of elitism was also used, whereby the new chromosome replaces the worst chromosomes in the population if, and only if, they have a lower cost than the worst chromosome in the population, however checks are not necessary to ensure that there are duplicates. Once again the termination criteria was based on finishing after a given amount of time so comparisons could be carried out between search techniques.

6.4. Tabu Search

Tabu Search, like Simulated Annealing and Genetic Algorithms, is used to compare the effect of different solution spaces on the search technique's performance. Unlike the other two techniques, extra experimentation is proposed which will determine the effect of using different starting points on the search techniques performance and the effect that the starting point has with the use of different solution spaces. The "standard" starting point adopted for all the searches was the EDD sequence with idle time optimally inserted or the E/T approximation to this for the heuristic solution space searches. Other starting points include randomly chosen points and the EDD sequence optimised by the Ow and Morton (1989) adjacency conditions. These other starting points are only incorporated into the most promising search techniques so that the sensitivity of the search to the starting point can be determined.

6.4.1. Standard Tabu Search

The Standard TS algorithm is implemented by using only the short term memory component as outlined by Glover (1990c). The search uses the solution space of sequences of jobs with the Fry, Armstrong, *et al.* (1987) idle time insertion algorithm to schedule the jobs.

The initial starting point for the search is the EDD schedule with idle time inserted optimally. The Standard TS uses a combination of both 'swap' and 'insert' neighbourhood schemes as this was found to be the most effective neighbourhood scheme, for scheduling problems, by Barnes and Laguna (1991a).

The tabu conditions are that the job that is moved in the insert neighbourhood scheme, and one of the jobs moved in the swap neighbourhood scheme are deemed 'tabu' and placed on the tabu list. This job cannot move from its current position in the schedule unless the aspiration criteria is met. This, however, does not stop the completion time of the tabu job being changed, as at each stage idle time is inserted optimally which will change completion time of most jobs. The size of the tabu list is fixed at a size determined by a parameter which was also used to determine the effect the tabu list size had on solution quality. The aspiration criteria used is that a tabu move is accepted if it produces a schedule with a cost lower than the cost of the best schedule found so far. The stopping criteria for the search was that the search was to terminate after a given amount of time. This enabled a fair cross comparison of different search techniques. None of the other extensions proposed in the literature were implemented to provide a comparison between the basic search techniques.

A second Standard TS technique was identical except that, instead of starting at the EDD sequence, it started at the best schedule of the E/T approximation to the EDD sequence and the EDD sequence.

6.4.2. Standard Tabu Search With “Windows”

The Standard Tabu Search with Windows is implemented in the same way as the Standard Tabu Search but the neighbourhood is reduced to a ‘window’ of the full neighbourhood scheme. A window is defined as a specific number jobs on either side of the current job. Consider the sequence shown in Figure 6-1, with a window size of 4. The neighbours which involve job 5, would involve swap and insert moves which also involve jobs 3, 4, 6 and 7. The neighbours which would involve job 1 would be swap and insert moves which also involve jobs 2 and 3.

1	2	3	4	5	6	7	8	9	10
---	---	---	---	---	---	---	---	---	----

Figure 6-1. Sample Sequence

This effectively reduces the size of the neighbourhood, a feature which is very important for solving larger scheduling problems.

Again two different versions of this technique are implemented, one starting at the EDD sequence, the other starting at the best of the EDD sequence and the E/T approximation to the EDD sequence.

6.4.3. Revised Tabu Search

The Revised Tabu Search method is identical to the Standard Tabu Search technique except that it includes the extension proposed by Reeves (1993). This extension chooses a window of jobs to be evaluated before choosing a move and updating the current move. Unlike the window modification, the revised tabu search does not restrict the size of the neighbourhood for each job, rather it restricts the number of jobs that are evaluated. In this sense, these two techniques are the exact opposite of each other. If we split a neighbourhood move into two types of jobs, source jobs (jobs that are to be moved) and destination jobs (where the source job is to be moved to or the job which it is to be swapped with), the window extension restricts the number of the destination jobs while the revised extension restricts the number of source jobs. The random method of selecting ‘source’ jobs was adopted, as Reeves found that this method was slightly more efficient than the linear scheme that was also proposed by Reeves (1993). The Revised TS was implemented by using three different starting points:

1. the EDD sequence with optimally inserted idle time,
2. the best schedule out of the EDD sequence and its E/T approximation, and
3. the EDD sequence which has been optimised to conform to the adjacency conditions outlined by Ow and Morton (1989).

6.4.4. Revised Window Tabu Search

The Revised Window Tabu Search combines the Revised Tabu Search and the Window Tabu Search into a single technique. In this search we not only restrict the number of

source jobs, but we also restrict the number of destination jobs as well. This type of tabu search is designed specifically for larger problems where neighbourhood sizes need to be severely restricted. This technique was also implemented with three different starting points:

1. the EDD sequence with optimally inserted idle time,
2. the best schedule out of the EDD sequence and its E/T approximation, and
3. the best schedule out of the EDD sequence which has been optimised to conform to the adjacency conditions outlined by Ow and Morton (1989) and its E/T approximation.

To test the significance that the starting points have on the final solution quality, two other random starting points were also tested, one which evaluated 100 different randomly generated schedules and chose the best solution as the starting point, while the other search chose the worst solution as its starting point.

6.4.5. Standard E/T Tabu Search

The Standard E/T Tabu Search implements the short term memory function of TS but instead of using the sequence of jobs solution space, it uses the E/T job specification solution space. The initial starting point is an E/T approximation to the EDD schedule with idle time optimally inserted which is found in the same manner as the E/T SA method. The neighbourhood scheme used takes each job in turn, changes its E/T job status from being early to late, or vice versa and evaluates this new E/T job specification by using the five point insertion heuristic. Since the E/T solution space can contain invalid moves, a special two level tabu list was constructed for this type of search to stop the search becoming trapped in areas of the solution space where the only valid moves are tabu. The tabu list contains not only the list of moves that are tabu but also the number of times the search has violated that tabu status to avoid getting stuck in an area of infeasibility. The search will, if there are no non-tabu moves that can be made, use the tabu job with the least number of tabu violations. This enables the search to back-track to an area in the solution space that has not been searched and proceed from there. Figure 6-2 illustrates this situation and how the search handles it. Once the search has entered the place within the solution space whereby it can not make any further moves due to invalid solution spaces (move 4), it back tracks on tabu moves (moves 5 and 6) to allow it to get back to an area where it is not trapped (move 7).

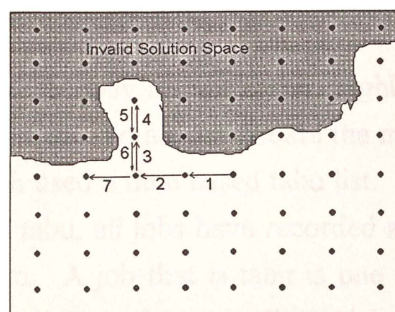


Figure 6-2. Searching With Invalid Solution Space

The search uses a fixed sized tabu list which is determined by a parameter and can be used to measure the effect of the tabu list size on solution quality. The same aspiration criteria is used as the standard TS, with a tabu job being accepted if the schedule produced has a cost lower than the best schedule so far.

6.4.6. Revised E/T Tabu Search

The Revised E/T Tabu Search is implemented in the same way as the standard E/T tabu search except that the size of the neighbourhood is reduced through the use of the revised tabu search extension (Reeves, 1993) which randomly selects a given number of jobs before choosing its next move. The “window” based tabu search could not be implemented since the neighbourhood used for the E/T scheme involves just one job, hence there are no “destination” jobs that need to be considered. As with the revised window tabu search, the revised E/T tabu search was also used to check the effect of different starting points by using good and bad random starting points as well as starting from the EDD sequence optimised by the Ow and Morton (1989) adjacency conditions.

6.4.7. Analyse Search

The Analyse Search is a method that implemented a tabu search on a slightly different neighbourhood scheme to the standard E/T neighbourhood scheme, although this neighbourhood scheme could be implemented with any of the other search methods if required.

The analyse neighbourhood scheme improves on the standard early/tardy search method by using the iterative approach outlined in Figure 6-3. This approach takes the sequence given by the early/tardy heuristic and re-inserts idle time into it to ensure that the sequence is optimally scheduled without the constraints of the early/tardy specification. The resulting schedule is then analysed and any necessary changes in the early/tardy specification are made. The process is repeated until an equilibrium is reached between the E/T job specification and the idle time insertion. In this way only schedules which are optimally scheduled are ever considered in the search. It is possible for this heuristic to perform better than the best possible solution of the E/T heuristic as it will also record the best solution found after idle time is inserted, which may not be obtainable by the heuristic.

Due to the large differences in the way in which the neighbourhood scheme operates, the tabu list also needed to be modified. To accommodate the many changes that were possible in one iteration, the tabu search used a time based tabu list. In this type of tabu list, instead of recording each job which is tabu, all jobs have recorded a time (the iteration number) of when they were last made tabu. A job that is tabu is one whose time is greater than the current iteration counter less the ‘size’ of the tabu list which, in this case, actually means the

number of iterations passed. This allows any number of jobs to become tabu in one iteration, and also allows the tabu status to be removed at the same time. Due to the dynamic nature of the neighbourhood, this technique is the slowest of all the search techniques. It was therefore important to try to reduce the size of the neighbourhood as much as possible. For this reason no aspiration criteria was used as a way of reducing the neighbourhood further and to speed up the search.

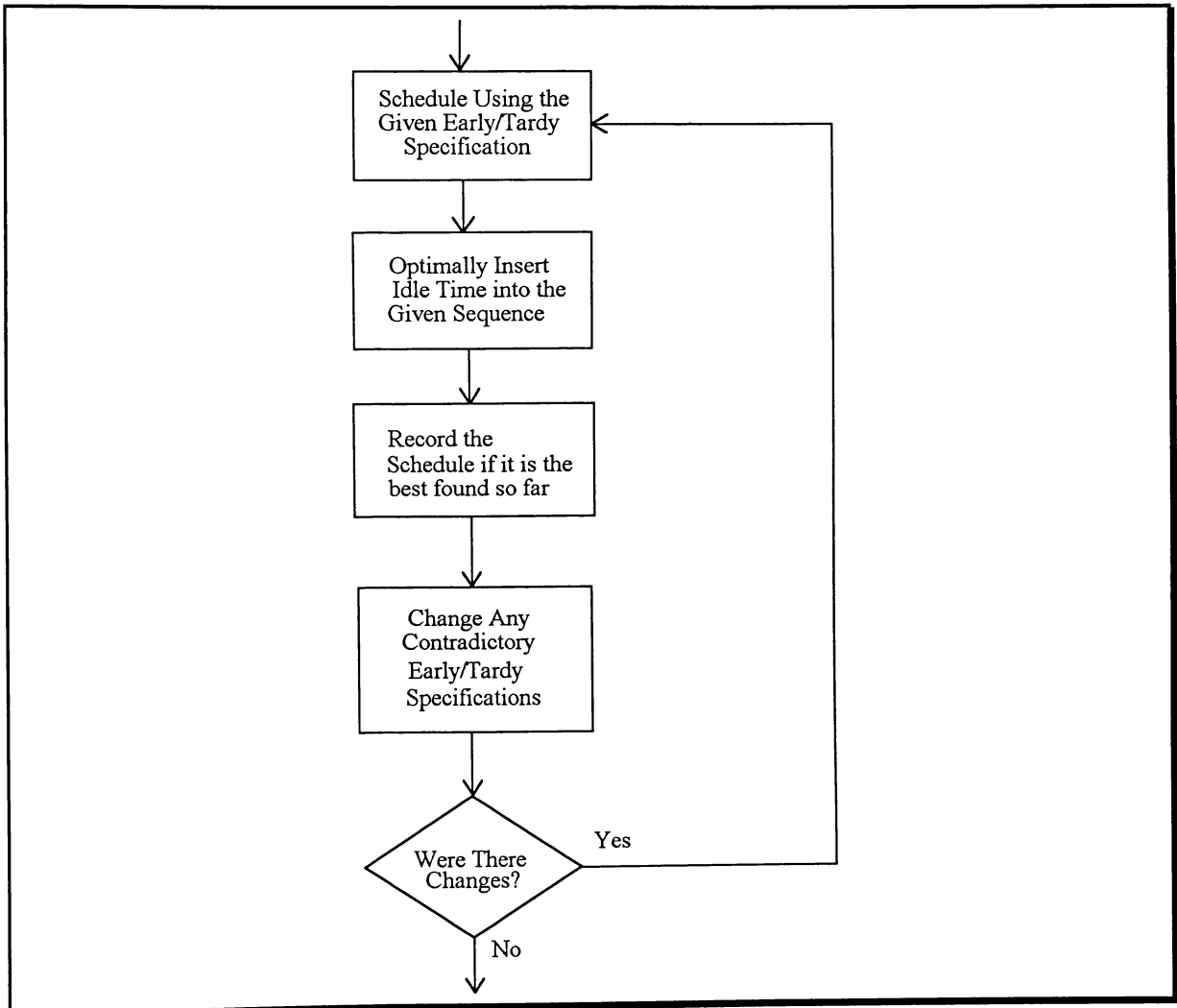


Figure 6-3. Analyse Algorithm

6.4.8. Revised Analyse Search

The Revised Analyse Search combines the Revised Early/Tardy Tabu Search and the Analyse Search neighbourhood. The neighbourhood scheme and the structure of the tabu list are taken from the analyse search and the selection criteria for jobs under consideration at each iteration are taken from the Revised Early/Tardy Tabu Search.

6.4.9. Revised “Smart” Early/Tardy Tabu Search

The Smart Early/Tardy Tabu Search uses the same ‘smart’ neighbourhood scheme used by the Smart Early/Tardy Simulated Annealing method which uses a neighbourhood scheme

that attempts to maintain a balance between the changes in early and tardy jobs within a group.

The smart neighbourhood scheme operates in a similar manner to a standard order-based revised tabu list rather than the random implementation that was used in the other revised tabu search variants. The order-based version was chosen because of the need to analyse the schedule and classify jobs into being either early or late. Since these jobs are in schedule order, this means that information carried out on the previous job can be reused for the next neighbour. This saves computational time and avoids the unnecessary duplication of work from having to reclassify early/tardy jobs which would be necessary if the random scheme was used.

The efficiency of the neighbourhood is increased by only allowing swaps with earlier jobs in the schedule, as this eliminates duplication in the neighbourhood. The tabu list has a fixed size, defined by a parameter. At each iteration the tabu list has either one or two entries added to it depending on whether one or two jobs had been being changed from early to late or vice versa.

6.4.10. Revised Window Tabu Search with Early/Tardy Extension

The Revised Window Tabu Search with Early/Tardy extension combines both the Revised Window Tabu Search technique and the Early/Tardy heuristic. The method uses the same neighbourhood as the Revised Standard Tabu Search but once it selects a move an extra diversification stage is added. The new sequence is then converted into its early/tardy job specification and solved using the early/tardy five point insertion heuristic. If the early/tardy schedule is better than the original schedule, then it is adopted as the current schedule.

The tabu list is managed in exactly the same way as the Revised Window Tabu Search. The starting point for this method is the better of the EDD sequence with idle time optimally inserted or the early/tardy approximation to this schedule. The reason for using this combination of techniques is that the standard techniques are far quicker in evaluating their neighbours than the E/T heuristic, hence the E/T heuristic is used to try to diversify the search at each iteration. A problem that standard tabu search has is that it can take a long time to move between local minima. This technique is designed to help alleviate this problem. As with the revised window tabu search, this tabu search was also used to check the effect of different starting points by using good and bad random starting points as well as using the Optimised Ow and Morton EDD starting point.

6.4.11. Revised Early/Tardy and Standard Hybrid Tabu Search

In this particular TS we again combine the Revised Tabu Search and the Revised E/T Tabu Search. This time, however, we start with the early/tardy tabu search to get some initial

gains from the heuristic then, after a given number of upward movements (defined by a parameter), the search switches over to the standard tabu search and continues from where the E/T search left off. The reasoning for this combination of techniques is that the early/tardy heuristic can make some large gains at the start of its search but does not perform as well as the standard tabu search in moving to new regions of the search space. This may be due to the narrower neighbourhood of the heuristic being unable to move into these new areas of the search space.

6.5. Common Due Date Tabu Search Methods

To determine whether a compressed solution space, based on the early/tardy job specifications, is beneficial to the solution of scheduling problems, several tabu search methods were developed to solve the common due date problem. As the common due date problem does not require the use of a heuristic for it to be solved, this problem enables an objective assessment of the success of compressed solution spaces with search techniques, without interference from the errors produced by the heuristics.

6.5.1. CDD E/T Tabu Search

The common due date E/T Tabu Search is identical in its implementation to the E/T TS except that the five point insertion heuristic has been replaced by an algorithm specifically for CDD problems. This algorithm takes the E/T job specifications and moves all early jobs to the start of the schedule and orders them in non-increasing order of the ratio p_i/α_i while the tardy jobs are scheduled at the end of the schedule and ordered in non-decreasing order of the ratio p_i/β_i . This algorithm is considerably faster than the five point insertion heuristic hence the performance from this search is greatly improved over the E/T tabu search. The initial schedule, since EDD ordering is no longer possible, sequences jobs in non-increasing order of ratio p_i/α_i and then inserts optimal idle time. All jobs scheduled before the common due date are given an early job specification, while all jobs scheduled after the common due date are given a tardy job specification.

6.5.2. CDD Revised E/T Tabu Search

The common due date Revised E/T Tabu Search is implemented in the same manner as the common due date E/T Tabu search except that it also implements the revised tabu extension of Reeves (1993).

6.5.3. CDD Revised E/T Tabu Swap Search

The common due date Revised E/T Tabu Swap Search is implemented in the same manner as the common due date revised tabu search except that it uses an extended neighbourhood scheme. Instead of just changing one job from early to late or vice versa, it also allows moves which change one job from early to late while another job is also changed from late to early. This type of search accommodates the situation where late jobs can not be

scheduled early due to the time constraints placed on the schedule by the common due date.

6.5.4. CDD Revised E/T Tabu Date Swap Search

The common due date Revised E/T Tabu Date Swap Search is implemented in the same manner as the common due date revised tabu search except that it uses a slightly different neighbourhood scheme. To avoid the problem of trying to schedule late jobs early when the common due date restricts the number of early jobs, this search automatically schedules the early jobs closest to the due date, late, so the late job being moved can be accommodated.

6.6. Summary

This section summarises the ways in which each search technique is implemented. Table 6-1 details all the SA methods, Table 6-2 all the GA methods and Table 6-3 all the standard TS methods. Table 6-4 outlines the search techniques which are based on modified TS methods by using a combination of solution spaces. Table 6-5 summaries the TS methods which are used to determine the effect of the starting point on the quality of the final solution. Table 6-6 summarises the TS methods designed specifically for the common due date problem which are later used to determine the effect of using a compressed solution space on the speed of the search.

Search Name	Solution Space	Neighbourhood Scheme	Starting Point
Std SA	Sequence of Jobs	Insert + Swap	EDD
Std SA ETS	Sequence of Jobs	Insert + Swap	Best of EDD & E/T Approx EDD
E/T SA	E/T Job Specification	Flip E/T Specification	E/T Approx. EDD

Table 6-1. Summary of SA Search Techniques

Search Name	Solution Space	Neighbourhood Scheme	Starting Point
Std GA	Sequence of Jobs	PMX, OX, CX, UOBX Order Based Mutation	Random + EDD Seed + E/T Approx EDD Seed
E/T GA	E/T Job Specification	Uniform Crossover, E/T Flip Mutation	Random + E/T EDD Approx.

Table 6-2. Summary of GA Search Techniques

Search Name	Solution Space	Neighbourhood Scheme	Starting Point
Std TS	Sequence of Jobs	Insert + Swap	EDD
Std TS ETS	Sequence of Jobs	Insert + Swap	Best of EDD & E/T Approx EDD
Std TS with Windows	Sequence of Jobs	Restricted Destination Insert + Swap	EDD
Rev TS	Sequence of Jobs	Restricted Source Insert + Swap	EDD
Rev TS OS	Sequence of Jobs	Restricted Source Insert + Swap	EDD Optimised by Ow and Morton
Rev Win TS	Sequence of Jobs	Restricted Source and Destination Insert + Swap	EDD
Rev Win TS ETS	Sequence of Jobs	Restricted Source and Destination Insert + Swap	Best of EDD & E/T Approx EDD
Rev Win TS OS	Sequence of Jobs	Restricted Source and Destination Insert + Swap	Best of EDD Optimised by Ow and Morton & E/T Approx to the same
E/T TS	E/T Job Specification	Flip E/T Specification	E/T Approx EDD
Rev E/T TS	E/T Job Specification	Restricted Flip E/T Specification	E/T Approx EDD
Rev E/T TS OS	E/T Job Specification	Restricted Flip E/T Specification	E/T Approx EDD Optimised by Ow and Morton

Table 6-3. Summary of TS Search Techniques

Search Name	Solution Space	Neighbourhood Scheme	Starting Point
Analyse	E/T Job Specification	Flip E/T Specification	E/T Approx EDD
Rev Analysed	E/T Job Specification	Restricted Flip E/T Specification	E/T Approx EDD
Rev Smart TS	E/T Job Specification	Single and Twin Flip E/T Specification	E/T Approx EDD
Rev Win TS ET Ext	Sequence of Jobs	Restricted Insert + Swap	Best of EDD & E/T Approx to EDD
Rev Win TS ET Ext	Sequence of Jobs	Restricted Insert + Swap	Best of EDD Optimised by Ow and Morton & E/T Approx to the same
Rev E/T Std Hybrid	Phase 1: E/T Job Specification Phase 2: Sequence of Jobs	Phase 1: Restricted Flip E/T Specification Phase 2: Restricted Insert + Swap	Phase 1: E/T Approx EDD Phase 2: E/T Solution

Table 6-4. Summary of Modified TS Search Techniques

Search Name	Solution Space	Neighbourhood Scheme	Starting Point
Rev Win TS Ran Good Start	Sequence of Jobs	Restricted Source and Destination Insert + Swap	Random Good Start
Rev Win TS Ran Bad Start	Sequence of Jobs	Restricted Source and Destination Insert + Swap	Random Bad Start
Rev E/T TS Ran Goods Start	E/T Job Specification	Restricted Flip E/T Specification	Random Good Start
Rev E/T TS Ran Bad Start	E/T Job Specification	Restricted Flip E/T Specification	Random Bad Start
Rev Win TS ET Ext Ran Good Start	Sequence of Jobs	Restricted Insert + Swap	Random Good Start
Rev Win TS ET Ext Ran Bad Start	Sequence of Jobs	Restricted Insert + Swap	Random Bad Start

Table 6-5. Summary of Starting Point TS Search Techniques

Search Name	Solution Space	Neighbourhood Scheme	Starting Point
E/T CDD Rev Win TS	Sequence of Jobs	Restricted Insert + Swap	Non-increasing Alpha Ratio
E/T CDD ET Rev Tabu	E/T Job Specification	Restricted Flip E/T Specification	Non-increasing Alpha Ratio
E/T CDD Win Rev ET Ext	Sequence of Jobs	Restricted Insert + Swap	Non-increasing Alpha Ratio

Table 6-6. Summary of Common Due Date TS Search Techniques

The computational experiments, carried out in Chapter 7, will also compare the best search from each of the search techniques. This will enable an assessment of which type of search technique is the most effective to solve the generalised E/T scheduling problem.

7. Computational Experiments

7.1. Introduction

In this chapter, methods for solving the distinct due date single machine E/T scheduling problem are examined. Section 7.2. explains how the random test problems for the computational experiments were generated. Section 7.3. outlines an experiment which determines which idle time insertion algorithm, either the Fry Armstrong *et al.* (1987) algorithm or the algorithm proposed in Chapter 4, is the best. Section 7.4. solves Common Due Date (CDD) problems to determine whether it is advantageous to use an E/T solution space rather than the sequence of jobs solution space. Section 7.5. compares the two E/T heuristics by using each of them to enumerate the entire solution space of small sized problems. The results from each heuristic are then compared in terms of time and solution quality. Section 7.6. provides details on experiments which involve search techniques and small sized problems, while Section 7.7. outlines experiments which involve large sized problems. The conclusions that can be drawn from these experiments are then summarised in Section 7.8.

All computational experiments were solved on a VAX 6000-620/630 mainframe computer system at the University of Waikato. All programs were written in GNU G++ 2.4.4, a variant of the C++ language.

7.2. Data Generation and Analysis

7.2.1. Data Generation

No standard E/T scheduling problem data has been reported in the literature, therefore problem data was generated using the method described by Baker and Martin (1974) who used their technique to generate data for the total tardiness problem. In this method two different parameters are used to describe the data:

1. The Tardiness Factor, which is the proportion of jobs which are expected to be tardy in a given sequence. Baker and Martin define the tardiness factor as:

$$t = 1 - \bar{d} / n\bar{p}$$

where

$\bar{d}$ = average due date, and

$\bar{p}$ = average processing time of the jobs being scheduled.

2. The Due Date Range, which is the expected range of the due dates as a proportion of the total processing time required for all jobs. Baker and Martin define the due date range as:

$$R = (b-a) / n\mu_p$$

where

a = minimum due date in the range,

b = maximum due date in the range, and

μ_p = the specified mean processing time of the jobs.

In effect, this tardiness factor (not to be confused with the tardiness penalty of each job) defines how close the due dates are to the current time (which in our tests is always time 0), while the due date range defines how spread out the due dates are relative to the average due date.

As suggested by Baker and Martin, the tardiness factors used in our tests were 0.2 and 0.6 and the due date ranges for the problem were 0.2 and 0.95. An extra due date range of 1.15 was also used to simulate the situation where idle time insertion would become more important. Two additional parameters were also included; the range of the earliness penalty, α , and the range of the lateness penalty, β . Three different sets of ranges, all drawn from the uniform distribution, were used to simulate different penalty scenarios. These situations were parameterised as $\alpha = [1,5]$ $\beta = [1,5]$, $\alpha = [1,5]$ $\beta = [6,10]$ and $\alpha = [6,10]$ $\beta = [1,5]$. From these parameters eighteen different cases were developed. In reality the most likely case is for the earliness penalty to be less than the tardiness penalty; hence only these cases were investigated with larger problems.

Processing times were drawn from a normal distribution with a mean of 100 and a standard deviation of 25, while due dates were drawn from a uniform distribution with a mean determined by the tardiness factor and a range determined by the due date range.

Table 7-1. outlines the parameters used in each of the problem classes. As the size of the problems vary, this information is provided in the explanation of each experiment.

Class Number	Class Name	Tardiness Factor	Due Date Range	Early Penalty Range	Tardy Penalty Range
1	A	0.2	0.2	[1,5]	[1,5]
2	B	0.2	0.95	[1,5]	[1,5]
3	C	0.2	1.15	[1,5]	[1,5]
4	D	0.6	0.2	[1,5]	[1,5]
5	E	0.6	0.95	[1,5]	[1,5]
6	F	0.6	1.15	[1,5]	[1,5]
7	G	0.2	0.2	[1,5]	[6,10]
8	H	0.2	0.95	[1,5]	[6,10]
9	I	0.2	1.15	[1,5]	[6,10]
10	J	0.6	0.2	[1,5]	[6,10]
11	K	0.6	0.95	[1,5]	[6,10]
12	L	0.6	1.15	[1,5]	[6,10]
13	M	0.2	0.2	[6,10]	[1,5]
14	N	0.2	0.95	[6,10]	[1,5]
15	O	0.2	1.15	[6,10]	[1,5]
16	P	0.6	0.2	[6,10]	[1,5]
17	Q	0.6	0.95	[6,10]	[1,5]
18	R	0.6	1.15	[6,10]	[1,5]

Table 7-1. Parameters for Each Class of Problem

7.2.2. Statistical Analysis

Many of the results presented in this chapter are for determining the best search techniques to use. A mean performance level for a search technique can be derived by averaging the percentage deviation from the smallest known minimum for each problem being considered. The different techniques can then be ranked by this mean to determine which performed the best. To test whether the results obtained from one technique are statistically significant from the results obtained from other techniques, multiple comparison statistics need to be used.

One of the most widely accepted multiple comparison statistical test is the Tukey Honestly Significant Difference (HSD) test. This test is the most conservative of the multiple comparison techniques as it exercises the greatest control over the rejection of the null hypothesis, which states that the two means are not significantly different (Klockars and Sax, 1986). This means that the Tukey HSD will be the least likely test to detect real differences between pairs of means (refer to Maxwell and Delaney (1990) for a full description of Tukey HSD analysis).

Due to the conservative nature of the Tukey HSD test, it has been used for all comparisons of the same type of search technique. Many of these results are presented in the appendices.

Like most other multiple comparison techniques, the Tukey HSD assumes homogeneity of variance, that is all samples are from populations which have the same variance. This assumption holds when we compare different parameters for the same search technique, but this is not so when comparing across different types of search techniques. To overcome this limitation a modification to the Tukey approach, proposed by Games and Howell (1976), was used. Details of this approach can be found in Klockars and Sax (1986), Howell (1992) and Games and Howell (1976). The implementation used in this thesis was based on Klockars and Sax (1986).

Statistics within this chapter show the Games and Howell Tukey results when comparing across different search techniques, however the appendices also provide the Tukey HSD results for comparison.

7.3. Idle Time Insertion

7.3.1. Introduction

Since the idle time insertion algorithm is important to the overall efficiency of search techniques (Yano and Kim, 1991), the first computational experiment explores which idle time insertion algorithm, either the Fry, Armstrong *et al.* (1987) algorithm or the algorithm developed in chapter 4 (referred to as the James Algorithm), is the most efficient.

To determine which algorithm is more efficient 90 problems, 5 problems from each of the 18 classes, were randomly generated using the procedure described in Section 7.2. Each of these problems was restricted to 8 jobs, which was small enough for the heuristics to fully enumerate yet still provide a non-trivial problem with a solution space size of 8! or 40,320 different solutions. For each problem both heuristics were used to fully enumerate the solution space, that is all possible sequences of jobs were calculated and idle time was inserted into each of those sequences. The time taken by each algorithm was recorded as a measure of its efficiency. Since both algorithms provide optimal solutions, solution quality is not an issue.

7.3.2. Results

The average times for each class of problem are shown Table 7-2.

Class Number	Class Name	James Algorithm CPU sec	Fry Algorithm CPU sec
1	A	181.742	22.928
2	B	149.264	18.204
3	C	125.112	18.408
4	D	126.104	11.718
5	E	123.728	12.540
6	F	101.902	11.912
7	G	158.510	13.560
8	H	124.556	13.046
9	I	122.808	13.140
10	J	114.168	10.486
11	K	103.668	10.510
12	L	100.834	10.508
13	M	172.114	28.526
14	N	140.636	23.614
15	O	139.228	24.298
16	P	125.692	15.492
17	Q	100.104	14.394
18	R	105.498	15.554
Mean		128.648	16.047
Standard Deviation		25.376	5.614

Table 7-2. Comparison of Idle Time Insertion Algorithm Times

From these results it is clear that the Fry Algorithm is superior to the James Algorithm in terms of computational efficiency as it is, on average, approximately 8 times faster. The reason for the James Algorithm being so much slower is that it requires all groups to be rescheduled at each iteration whereas the Fry algorithm works from the latest ‘groups’ to the earliest ‘groups’ therefore minimising the need to reschedule groups. The Fry, Armstrong *et al.* (1987) algorithm is therefore used for idle time insertion in all the remaining computational experiments carried out in this chapter.

7.4. Common Due Date Problems

7.4.1. Introduction

When considering the use of an E/T solution space, such as that proposed in Chapter 5, it is necessary to evaluate whether this type of solution space is going to provide a more efficient solution space to search than the sequence of jobs solution space. To evaluate this hypothesis we first consider the restricted common due date problem. For this problem, the best sequence of jobs can always be determined from a given E/T job specification, therefore the “heuristic” factor is removed from our analysis, eliminating any possible interference in our analysis. The only difference between this problem and the generalised E/T scheduling problem is that all jobs are due on the same date.

7.4.2. Computational Experiment

To test the efficiency of the different solution spaces, 6 different tabu search methods are used to solve 60 randomly generated, 250 job problems. The problems were generated according to the procedure outlined in Section 7.2., however the average of the due dates calculated was used as the common due date for the problems. The due date for these problems is therefore defined by the tardiness factor, as a common due date problem does not have a due date range. The problems that were considered tested only the case where earliness penalties are less than the tardiness penalties, as this is considered the most likely situation in reality. Thirty problems were generated for each tardiness factor 0.2 and 0.6. Each problem was given 7500 CPU seconds to be solved by each of the search techniques trialed.

The standard Revised TS and Revised Window TS starting at the same point as the E/T search were used to test the sequence of jobs solution space. To test the E/T solution space four different tabu search techniques were used: standard TS, Revised TS, Revised TS which included a Swap neighbourhood and a Revised Date Swap TS which included the extension of changing early jobs closest to the due date to being late if there was insufficient room to schedule a late job early (refer to Section 6.5.4). The parameters for the TS used in this experiment; shown (in Figure 7-1) in the order of tabu list size, source window size,

destination window size; were the best parameters found for the distinct due date experiment when solving 250 job problems (see Section 7.7.).

Figure 7-1 shows the average deviation from the known minimum of the problem for each of the different TS techniques at the average time that this minimum was found. The graph is an average of the 60 different problem results both in terms of time and deviation. The variance in both the deviation from the known minimum and the solution times is therefore not shown on this graph, however an ANOVA analysis is carried out in Appendix III to provide this information.

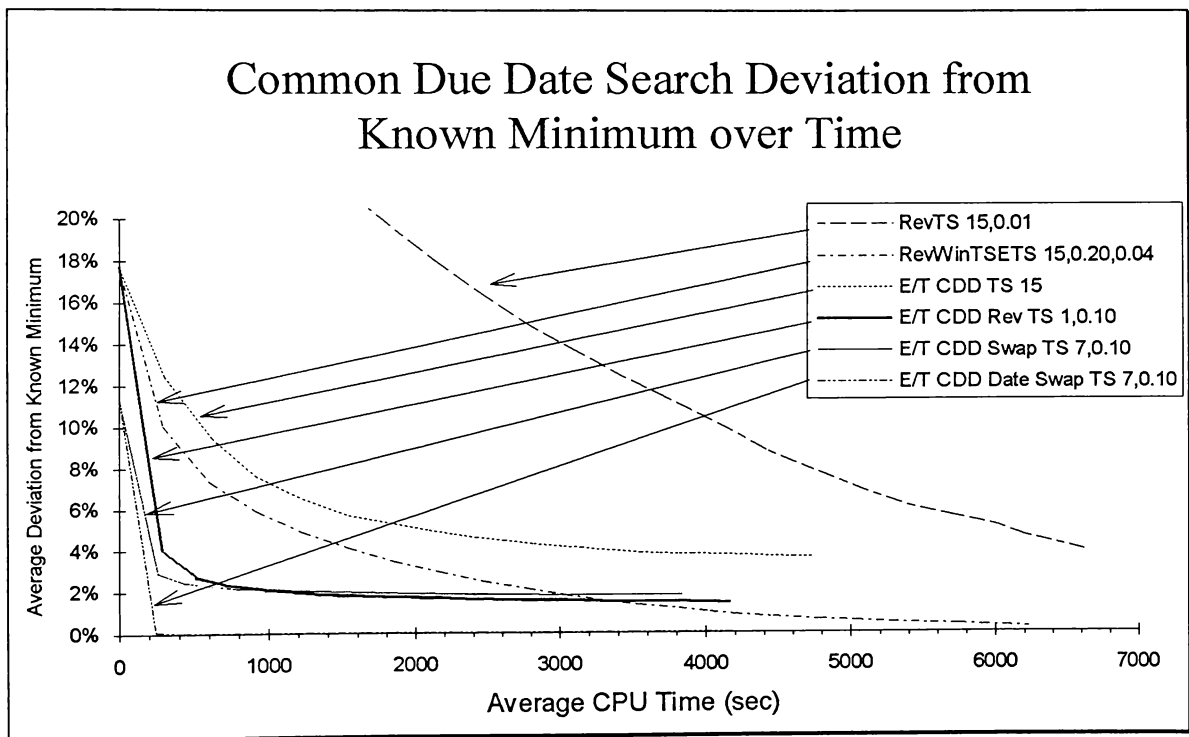


Figure 7-1.

Figure 7-1 emphasises the importance of using high quality starting points for these types of search techniques. The Revised TS takes over 2000 CPU seconds to obtain solutions of a similar quality to the starting point obtained from the E/T solution space. This in turn means lower quality solutions when time restraints are in place.

When the Revised Window TS is compared with the E/T CDD TS, it is clear that the E/T CDD TS is less efficient, however when we also include the Revised E/T CDD TS it proves that the inefficiency of the E/T CDD TS is due to the large size of the neighbourhood, rather than the solution space itself.

An interesting feature of the Revised E/T CDD TS is the way in which the search quickly finds solutions which are less than 4% deviation from the known minimum but after this

initial gain, very little progress is made. We also see that E/T CDD TS with the Swap neighbourhood extension also follows a similar pattern. While the swap extension finds better solutions initially, after 1000 CPU seconds the Revised CDD starts to provide better quality solutions. This is caused by the swap strategy intensifying the search in a particular area of the solution space, which is not always beneficial to the search, especially when the search actually needs to be diversified into new areas of the solution space. The Revised CDD search is therefore more likely to explore different regions of the solution space, hence slightly better long term progress is made.

The best search technique was clearly the E/T CDD Date Swap TS which, like the Revised E/T CDD TS, also improved significantly very quickly, however this time the search found solutions close to the known minima for each problem within an average time of 400 CPU seconds. This type of search has the same advantages of the swap TS, in that its performance is not affected by problems with little space for early jobs, due to a high tardiness factor, but it has the added advantage of resolving this problem in a manner which is simple, effective and more predictable than the E/T CDD Swap TS.

From the ANOVA analysis in Appendix III it can be seen that the statistical difference between these search techniques varies. The Revised TS is significantly worse than all the other TS methods. When the best standard TS method, Revised Window TS, and the best E/T TS, E/T CDD Date Swap TS, are compared it can be seen that these two methods are statistically different up until approximately 1000 CPU seconds at the 5% level of significance. This leads us to conclusion that by using an E/T solution space, either to generate a starting point or for an initial “quick search”, one can increase the solution quality significantly.

ANOVA analysis shows that there is no statistical significance between the Revised E/T CDD TS and the E/T CDD Date Swap TS other than at the starting point of the search. It also shows that these two techniques differ considerably in their performance on different problem classes. The Revised E/T CDD performs considerably worse on problems with a high tardiness factor, since the E/T CDD Date Swap scheme has been specially designed to cope with this type of problem as it makes particular early jobs become late jobs to increase the size of the valid neighbourhood. The ANOVA analysis also reveals that there is no statistical significance between the E/T CDD Date Swap TS and the E/T CDD Swap TS.

ANOVA analysis for the Revised Window TS reveals strong differences between the classes of problems. Figure 7-2 shows the effect of different tardiness factors on the solutions obtained by both the Revised Window TS and the E/T CDD Date Swap TS methods.

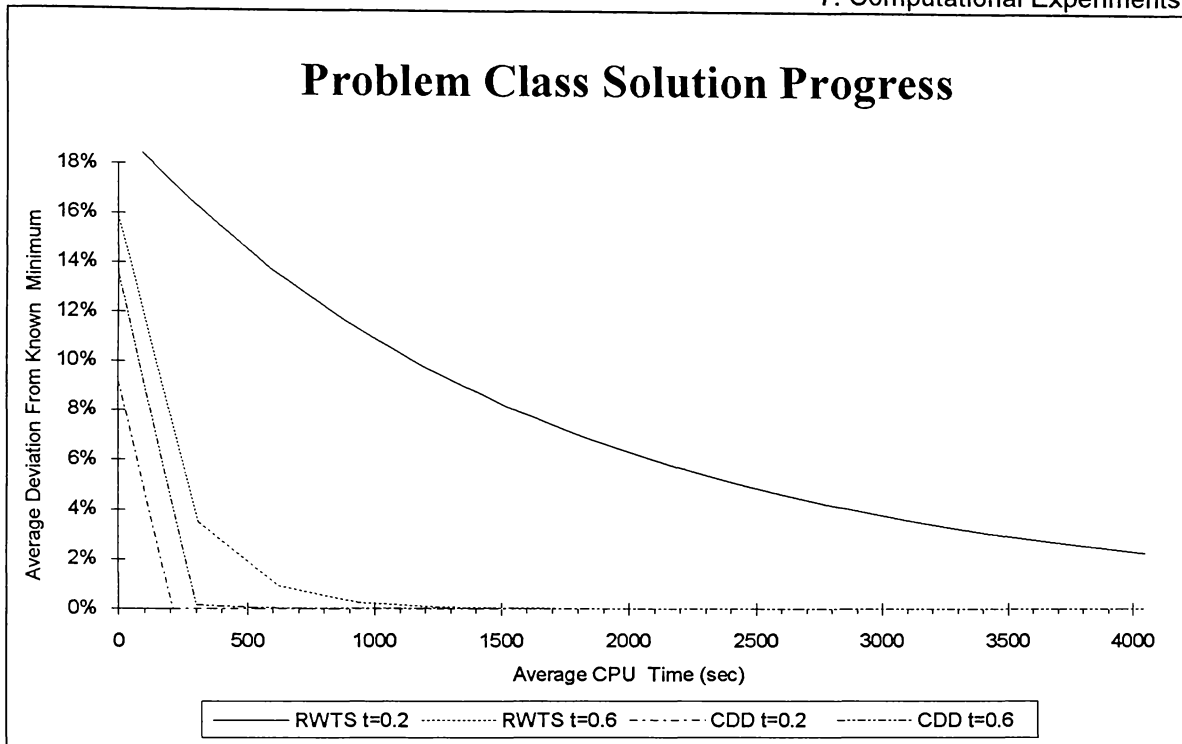


Figure 7-2.

Figure 7-2 shows that the Revised Window TS is slower at finding high quality solutions for problem classes with a low tardiness factor when compared to problems classes with a high tardiness factor. The reason for a low tardiness factor slowing down the search would be due to these classes of problems having less extreme costs in the neighbouring solutions, hence the solution space tends to be flatter and hence more difficult to search.

To determine how significant the tardiness factor is on the search, a t-test was performed on each method between the problems generated with the two different tardiness factors. The results from this analysis are presented in Table A3-2 in Appendix III. Each of the searches have results which differ significantly with the tardiness factor of the problem, except for the E/T CDD Date Swap TS. The E/T CDD Date Swap TS is insensitive to the tardiness factor except at its starting point, where classes with a low tardiness factor are solved significantly better than those with a high tardiness factor. From Figure 7-2 The E/T CDD Date Swap TS appears to solve the problem classes with a smaller tardiness factor slightly faster than those with a higher tardiness factor. The reason for this is that when the tardiness factor is low, the heuristic is less likely to reschedule early jobs when testing neighbours which change a late job to an early job, which is a more common occurrence when the tardiness factor is high.

Figure 7-3 gives another perspective on the quality of solutions obtained from the searches by showing the total number of solutions ranked either first or second from all the solutions obtained from the six different search methods used. It must be noted that this form of analysis eliminates the variance issues surrounding the solution quality, however comparing this graph to Figure 7-1 does give an indication of the variance in solution quality.

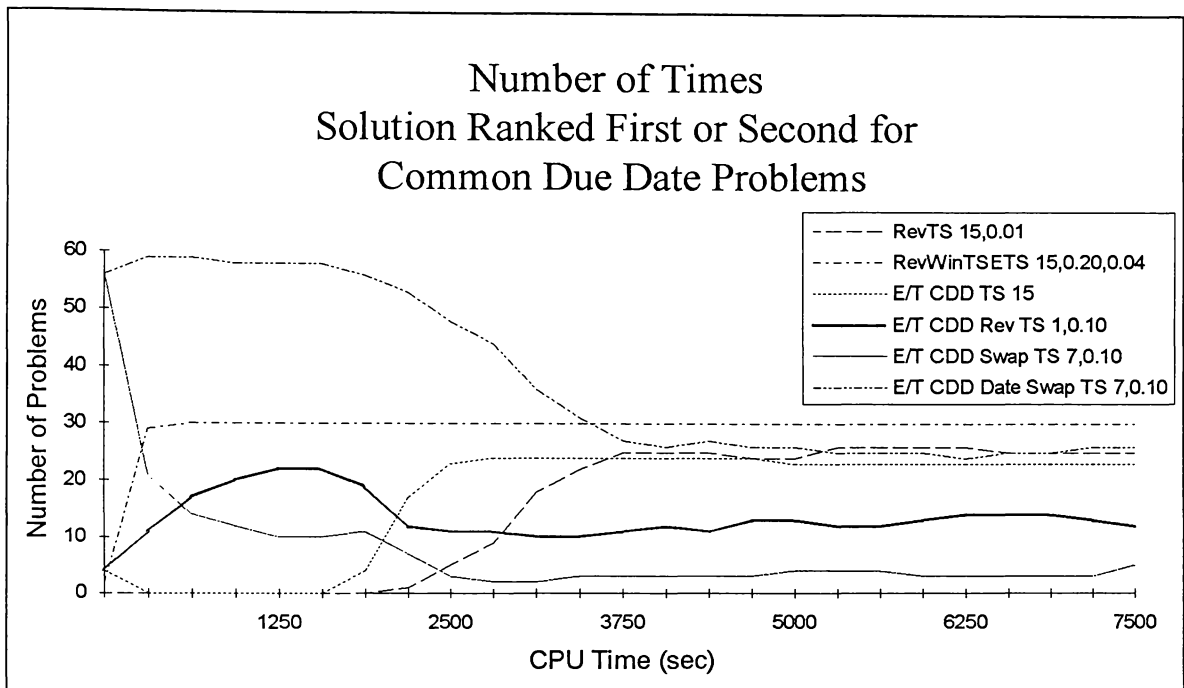


Figure 7-3.

Figure 7-3 demonstrates how the CDD Date Swap is consistently superior to all the other searches for approximately 3000 CPU seconds, then the Revised TS and the CDD TS techniques tend to ‘catch up’. This particular graph shows how these two techniques, although on average performing poorly, find good quality solutions for some problems if given enough time, however the variance in their solution quality is also high, hence their poor overall performance in Figure 7-1.

7.4.3. Conclusions

From these experiments we can conclude that the use of an E/T solution space does increase the performance of a search technique, such as TS, when compared to using a sequence of jobs solution space. This is especially true for problems that have a low tardiness factor. Overall the E/T Date Swap TS was found to give the best average performance of all the search techniques. The standard E/T CDD TS suffered when the tardiness factor was high as many of the moves which involved scheduling late jobs early became infeasible, making it difficult for the search to move to new areas of the solution space.

The experiments here show that by using an E/T solution space search initially, early gains can be made by a search technique. A standard search can utilise these initial gains by using the result as a starting point for its search, in effect creating a hybrid technique. The E/T solution space would be used to find the most suitable area to search while the standard TS used to ‘fine tune’ the solution.

7.5. Solution Space Computational Experiment

7.5.1. Introduction

In Section 7.4. it was concluded that the use of an E/T solution space with a search technique, such as tabu search, was beneficial in solving the CDD problem. CDD problems, however, can be optimally solved from the E/T job specification which is not possible for distinct due date problems, therefore two heuristics were devised to fulfil this function in Chapter 5. What we do not know is whether the time penalty and the inaccuracy of a heuristic in this role will cause a search to become ineffective.

To test the accuracy of these E/T heuristics, each heuristic was used to enumerate the entire E/T solution space with the objective of finding the best solution. The optimal solution for each problem was obtained by using the mixed integer formulation, developed by Keyser and Sarper (1991), which was solved using SAS/OR version 6.07. The size of the different problem classes vary as some of the problems could not be solved by SAS within the computer resource constraints of 6 hours CPU time and 100Mb disk space. The only option available was to reduce the size of the problems to allow SAS to solve the problem within these time and space constraints. Problems with 15 jobs were originally attempted for all classes, this was reduced until at least half of the problems could be solved at a particular size. Further jobs were then generated to ensure that each class had 10 solved problems in it.

7.5.2. Experiment Results

The results from solving these problems with mixed integer programming and the two E/T heuristics are summarised by problem class in Table 7-3.

Class No.	Problem Size (No. Jobs)	Five Point Insertion Average Deviation from Optimal	Five Point Insertion Average CPU Time	Dynamic Loop Average Deviation from Optimal	Dynamic Loop Average CPU Time	SAS-MIP CPU Time
1	9	0.746%	5.4	0.029%	7.3	1003.2
2	15	2.224%	846.9	1.646%	1368.2	722.3
3	15	1.300%	747.8	1.368%	1064.7	798.4
4	9	0.168%	3.0	1.217%	2.6	2130.6
5	10	0.326%	6.4	0.411%	8.1	862.6
6	12	0.372%	23.8	0.242%	36.3	1883.1
7	9	0.563%	5.4	0.139%	8.0	1517.9
8	15	1.416%	724.6	3.558%	1663.3	494.9
9	15	0.181%	678.9	0.538%	1104.6	533.7
10	8	0.000%	1.3	0.056%	1.7	598.7
11	10	0.111%	7.0	1.090%	10.1	940.1
12	10	0.388%	3.5	0.727%	4.6	699.1
13	9	0.087%	5.4	0.087%	8.0	1651.2
14	15	0.378%	843.4	0.413%	1243.3	1942.7
15	15	0.147%	745.7	0.313%	925.8	240.4
16	9	0.476%	3.4	0.591%	3.6	2809.5
17	12	1.257%	25.8	1.230%	30.7	1376.5
18	12	0.541%	16.6	0.514%	19.3	922.5
Overall Mean		0.593%	260.8	0.787%	417.2	1173.8
Standard Deviation		1.37%	359.4	1.731%	614.6	1538.6
Maximum		8.26%	1012.3	9.010%	2260.07	13119.4
Minimum		0.000%	0.94	0.000%	1.06	9.62

Table 7-3. Comparison of the Two E/T Heuristics and Mixed Integer Programming

The results, outlined in Table 7-3, show that both E/T heuristics provide good quality solutions, with both heuristics having a deviation from optimal of less than 1%, however the Five Point Insertion Heuristic has a lower overall mean deviation. An ANOVA test on the deviation from optimal of the two E/T heuristics gave a p-value of 0.2410, indicating that there is no statistical difference in the solution quality of the two E/T heuristics. An ANOVA test was used to determine if there was significant difference in the time taken between the different E/T heuristics. This test showed that difference between the times of the two heuristics was statistically significant (p-value of 0.000). These results point to the Five Point Insertion scheme being superior to the Dynamic Loop heuristic, as it is significantly faster and appears to have a slightly lower mean deviation.

Figure 7-4 shows the cumulative frequency graph of the results for the two E/T heuristics. This graph shows the proportion of problems solved with less than a given percentage

deviation from optimal. This graph provides some guidelines as to the probability of a solution to a problem being above a given percentage from optimal. For example, if a 2% deviation was acceptable then from Figure 7-4 we can see that over 90% of the Five Point Insertion heuristic's solutions were under 2%, therefore there would be a 10% chance that a solution used from this heuristic will be outside the acceptable range.

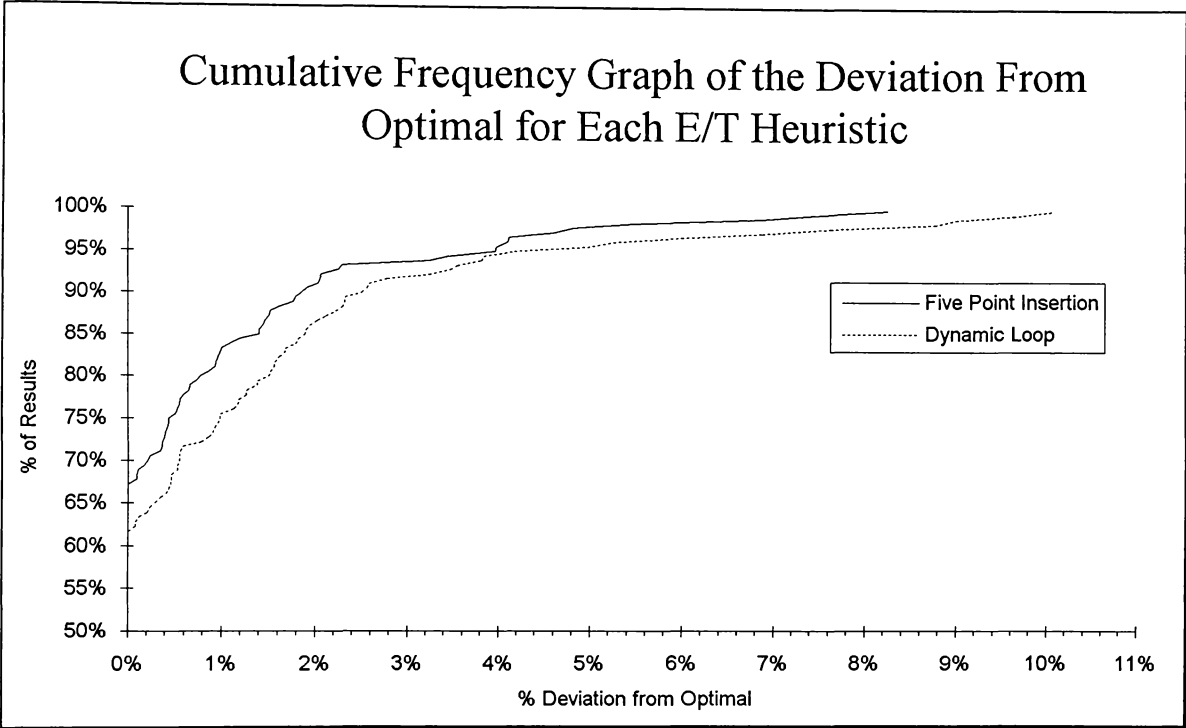


Figure 7-4.

Another ANOVA test was used determine how well the different classes of problems were solved using the E/T heuristics. Figure 7-5 shows the results of a Tukey HSD test with groups of jobs significant at the 5% level of significance, ranked from best to worst. Problem classes that are on a common line are not significantly different from each other. The class of problem has a major impact on the solution quality, with a p-value of 0.000. A two way ANOVA shows that the interaction between heuristic type and the problem is not significant on the solution quality with a p-value of 0.454.

Class #	10	13	15	6	7	9	5	1	14	18	16	12	11	4	17	3	2	8
---------	----	----	----	---	---	---	---	---	----	----	----	----	----	---	----	---	---	---

Figure 7-5 Tukey HSD Test by Problem Class

7.5.3. Conclusions

From this experiment it can be concluded that the Five Point Insertion heuristic is superior to the Dynamic Loop heuristic due to two distinct factors:

- 1. It has a lower mean deviation from optimum, although this is not statistically different to the mean of the Dynamic Loop heuristic, and

2. It has a far lower computational time, which is statistically significant from the Dynamic Loop heuristic.

The Dynamic Loop heuristic was also found to solve larger problems slower than the Five Point Insertion heuristic. This can be seen by looking at the ratio of (Total Time for Enumerating Using Dynamic Loop) : (Total Time for Enumerating using the Five Point Insertion) for the 9 job problems and the 15 job problems. For the 9 job problems the Dynamic Loop is 30.3% slower than the Five Point Insertion, while for the 15 job problems the Dynamic Loop is 60.6% slower. Although some of this difference may be due to the different problem classes, further tests on large sized problems indicated that the performance of the Dynamic Loop deteriorated faster than that of the Five Point Insertion heuristic.

Due to these two factors the Five Point Insertion heuristic was used as the E/T heuristic in the remaining computational experiments. The Dynamic Loop heuristic was not developed further as, even if its accuracy was improved through better cost estimating heuristics, it was clearly too slow. Further research is therefore required into how the speed of conflict resolution heuristics, such as the Dynamic Loop heuristic, could be increased.

The development of better E/T heuristics which are faster and/or more accurate is an area requiring further research. As the E/T heuristic determines the overall performance of the search technique, both in terms of speed and quality of solution, research on improving this heuristic will, potentially, produce large gains in the quality of the search being performed.

7.6. Small Problems

7.6.1. Introduction

We know that the E/T solution space can provide significant benefits for large CDD problems, and that the Five Point Insertion heuristic provides a means of obtaining high quality solutions for the distinct due date problem.

In this section we explore the use of different standard heuristics and the use of different search heuristics on the small problems that were developed in Section 7.5. The computational results of each search using the same class of search heuristic are compared to determine better search parameters and modifications. These better searches are then compared across search heuristics to give a picture of which type of search heuristic is the most successful.

For each of the search heuristics, the search was terminated after a given number of seconds. To maintain comparability between problem classes of different sizes, this time was set at 0.95 CPU seconds per job in the problem being solved as this approximately equal to

30 seconds on a 33Mhz 386 PC. This amount of time was considered to be very generous, especially when existing heuristics found solutions to these problems in under 0.1 CPU seconds, hence a longer amount of time would have little effect on the results.

7.6.2. Existing Heuristic Results

The first experiment on these small sized problems was to solve each problem using the techniques outlined in the literature for distinct due date E/T scheduling problems. The heuristic developed by Fry, Armstrong *et al.* (1987) was trialed, however this heuristic was designed specifically for problems where all the jobs had the same E/T penalties, so it was not expected to perform well on this more general problem. The heuristic of Keyser and Sarper (1991) was trialed in two forms; the original form, which did not contain an algorithm for optimal idle time insertion, and the modified form, which included the idle time insertion algorithm of Fry, Armstrong *et al.* (1987). This modification was expected to increase the quality of solutions significantly. Two other dispatching heuristics were also trialed, the earliest due date (EDD) and the Smallest Slack (SLK) dispatching rules, which were used by both the Fry, Armstrong *et al.* (1987) and Keyser and Sarper (1991) as starting points for their heuristics. Both of these heuristics had idle time optimally inserted using the Fry, Armstrong *et al.* (1987) algorithm. The E/T approximations to each of these dispatching rules using the Five Point Insertion heuristic were also used to show the difference the E/T scheme can make to initial starting points. The results from this experiment are outlined in Table 7-4.

Heuristic	Average Difference from Optimal	Standard Deviation of Difference from Optimal	Minimum Difference from Optimal	Maximum Difference from Optimal	Average CPU Time (Sec)
Fry, Armstrong <i>et al.</i>	105.31%	84.29%	0.00%	640.96%	0.001
Keyser and Sarper	48.14%	80.63%	0.00%	406.35%	0.054
Keyser and Sarper with Fry Idle time insertion	5.09%	8.51%	0.00%	49.63%	0.082
Earliest Due Date (EDD) with Idle Time Inserted.	45.12%	30.50%	0.00%	171.57%	0.035
Smallest Slack (SLK) with Idle Time Inserted	46.60%	29.73%	0.00%	183.54%	0.019
E/T EDD	16.36%	18.15%	0.00%	85.46%	0.037
E/T SLK	18.78%	21.81%	0.00%	180.43%	0.022

Table 7-4. Results from Previous Heuristics on Small Problems

From the results outlined in Table 7-4, as predicted, the Fry, Armstrong (1987) performed poorly. The original Keyser and Sarper (1991) heuristic performed worse than the EDD and SLK algorithms with idle time optimally inserted, however the Keyser and Sarper (1991) heuristic with optimal idle time did provide far better results, thus demonstrating the importance of idle time insertion on the quality of solutions obtained.

Both E/T approximations demonstrate that large gains can be made by using the E/T solution space, with both producing average results which are less than half that of the standard heuristic results. It is also clear from these results that the EDD rule is slightly better than the SLK dispatching rule, as it has a slightly lower mean deviation from optimal. The E/T approximations, however, accentuate this, as the E/T EDD provides a mean which is over 2% better and also has a lower variance.

Overall the results obtained from these heuristics are relatively poor. The best heuristic, the Keyser and Sarper heuristic with the Fry idle time insertion algorithm, also takes the longest time to calculate - hence a time/quality tradeoff is evident here.

7.6.3. Simulated Annealing Results

The three SA techniques outlined in Chapter 6 were tested using the small problems generated in Section 7.5.

The initial temperature, $temp_0$, was tested at 0.5, 0.75, 1.00 and 1.25. All temperatures over 1.00 mean that every neighbour was accepted, hence at an initial temperature of 1.25 the search was initially conducting a 'random walk'. The delta value's, Δ , decrement rate, x , was tested at 0, 0.005, 0.01, 0.05, 0.1, 0.25, 0.5, 1, 1.5, 2, 2.5 and 3. The larger the value of x , the faster the temperature cooled during the search.

The results from these experiments are outlined in Tables 7-5 and 7-6.

Type of Simulated Annealing	Lowest Mean Deviation from Optimal	Best Initial Temperature	Best Delta Parameter	Significance of Initial Temperature (p-value)	Significance of Delta Parameter (p-value)
Standard	0.0446%	1.25	0.05	0.994	0.000
Early/Tardy	0.8229%	0.50	1.50	0.889	0.000
ET Smart	1.11812%	0.75	1.50	0.827	0.002

Table 7-5. Simulated Annealing Small Problem Results

SA Technique	Groups from Best to Worst (Significance at 5%)											
Standard												
Initial Temperature	<u>0.75</u>	<u>0.50</u>	<u>1.00</u>	<u>1.25</u>								
Delta Parameter	<u>0.05</u>	<u>0.1</u>	<u>0.25</u>	<u>0.5</u>	<u>1.0</u>	<u>0.01</u>	<u>1.5</u>	<u>2.0</u>	<u>3.0</u>	<u>2.5</u>	<u>0.005</u>	<u>0.00</u>
Early/Tardy												
Initial Temperature	<u>0.50</u>	<u>1.25</u>	<u>1.00</u>	<u>0.75</u>								
Delta Parameter	<u>0.50</u>	<u>1.0</u>	<u>0.25</u>	<u>1.5</u>	<u>2.0</u>	<u>2.5</u>	<u>3.0</u>	<u>0.10</u>				
ET Smart												
Initial Temperature	<u>1.00</u>	<u>0.75</u>	<u>0.50</u>	<u>1.25</u>								
Delta Parameter	<u>1.50</u>	<u>0.5</u>	<u>1.0</u>	<u>0.25</u>	<u>2.0</u>	<u>2.5</u>	<u>0.10</u>					

Table 7-6. Simulated Annealing Tukey HSD Analysis

The results show that the value of the delta parameter is statistically significant for all the SA techniques, while the value of the initial temperature does not make a significant difference.

The Tukey HSD analysis on the standard SA approach shows that the delta parameters between 0.05 and 0.1 provide the best results. Delta parameter values higher than 1.5 appear to ‘freeze’ the system too quickly by getting the search stuck in a local minima, while values less than 0.05 tend to allow too many poor moves for too long.

The Tukey HSD analysis for the early/tardy SA approach shows that all the delta parameter values except for 0.10 are not statistically different. Delta parameter values of less than 0.25 and greater than 1.5 will therefore produce substantially poorer results when using this solution space. This indicates that the E/T solution space requires a faster cooling rate than the standard solution space. This is due to the way that large changes can be made to the resulting schedule by small changes in the E/T job specifications. To discourage these large changes the temperature needs to be decreased more quickly.

The smart early/tardy simulated annealing has results that are very similar to the standard E/T SA technique in that the delta parameters values are not statistically different from each other except for 0.10. Generally delta parameter values between 0.5 and 1.5 appear to be the best.

The three SA techniques were then compared using the results obtained from the parameters which gave the lowest average deviation from optimal. The results of this analysis are shown in Table 7-7.

	Standard	Early/Tardy	E/T Smart
Mean Deviation from Optimal	<u>0.0004</u>	<u>0.0082</u>	<u>0.0118</u>

Table 7-7. Games and Howell Tukey Analysis between Simulated Annealing Techniques

Table 7-7 clearly shows that the use of SA with a standard solution space provides superior results to using SA with the E/T solution space for small sized problems. The E/T SA also has a lower mean than the E/T Smart SA, however this is not statistically different.

7.6.4. Genetic Algorithms Results

The two GAs outlined in Chapter 6 were used to test GA searches on the same small problems used in the previous section.

The population sizes that were trialed were 10, 20, 30, 40, 50, 60, 70 and 100. The different mutation rates trialed were 0, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9 and 1.0.

The results of this experiment are outlined in Tables 7-8 and 7-9

Type of Genetic Algorithm	Lowest Mean Deviation from Optimal	Population	Mutation	Significance of Population (p-value)	Significance of Mutation (p-value)
Standard	0.0930%	20	0.4	0.647	0.000
Early/Tardy	0.9470%	60	0.4	0.000	0.000

Table 7-8. Genetic Algorithms Small Problem Results

GA Technique	Groups from Best to Worst (Significance at 5%)											
Standard												
Population	<u>50</u>	<u>60</u>	<u>30</u>	<u>40</u>	<u>10</u>	<u>20</u>						
Mutation	<u>0.3</u>	<u>0.5</u>	<u>0.1</u>	<u>0.05</u>	<u>0.4</u>	<u>0.2</u>	<u>0.0</u>					
Early/Tardy												
Population	<u>70</u>	<u>60</u>	<u>50</u>	<u>80</u>	<u>40</u>	<u>30</u>	<u>20</u>	<u>10</u>				
Mutation	<u>1.0</u>	<u>0.9</u>	<u>0.5</u>	<u>0.7</u>	<u>0.4</u>	<u>0.8</u>	<u>0.6</u>	<u>0.3</u>	<u>0.2</u>	<u>0.1</u>	<u>0.05</u>	<u>0.0</u>

Table 7-9. Genetic Algorithm Tukey HSD Analysis

The results for standard genetic algorithms show that the size of the population does not have a significant effect on the quality of the solution. The value of the mutation rate is also

not statistically significant for values greater than 0.0, indicating that some mutation is important in obtaining good quality solutions. There is, however, a strong interaction between the population size and the mutation rate (p-value of 0.009). The results from these experiments highlighted this relationship, with smaller populations performing better with higher mutation rates.

When using the early/tardy GA, the population size was found to be a significant factor in solution quality. No significant difference was found for populations greater than or equal to 50 although a population size of 70 appeared to give the best results. As with the standard GA there was a very strong interaction between the population size and the mutation rate (p-value of 0.000). All population sizes performed better at higher mutation rates, although the increase in performance was more significant with smaller population sizes than with the larger ones.

Clearly from the means of the two techniques, the standard technique performs better than the E/T GA technique. A one way ANOVA test on these two techniques confirms that these two techniques are significantly different from each other (p-value 0.000).

7.6.5. Tabu Search Results

Tabu Search was the most extensively tested type of search technique with 8 different tabu search based techniques being evaluated. This extensive testing was due, in part, to the positive results obtained from initial experimentation and also to the literature which suggests that TS is superior to both SA and GAs, for example Reeves (1993) .

For these experiments, various tabu list sizes were used ranging from 1 to 8. List sizes greater than 8 were not trialed since the smallest problem had just 8 jobs in it, and therefore potentially the search could have got stuck if all the jobs became tabu. Two different windows sizes were also used, 0.25 and 0.5. Smaller window sizes were not used as a window size of 0.25 provided only 2 jobs for an 8 job problem and any smaller than this would have been meaningless. The results from these experiments are outlined in Tables 7-10 and 7-11.

Type of Tabu Search	Lowest Mean Deviation From Optimal	Tabu List Size	Window Size	Significance of Tabu List Size (p-value)	Significance of Window Size (p-value)
Standard	0.3995%	6	N/A	0.090	N/A
Window	1.9120%	8	0.5	0.312	0.000
Revised	0.0219%	4-5 6-8	0.25 0.5	0.873	0.990
Early/Tardy	1.766%	3	N/A	0.640	N/A
Revised Early/Tardy	1.974%	1	0.5	0.000	0.000
Analyse	1.056%	3	N/A	0.143	N/A
Revised Analyse	1.029%	6	0.25	0.181	0.042
ET Smart	0.886%	4	1.00	0.879	0.000

Table 7-10. Tabu Search Small Problem Results

TS Technique	Groups from Best to Worst (Significance at 5%)							
Standard								
Tabu List Size	<u>8</u>	<u>6</u>	<u>7</u>	<u>5</u>	<u>4</u>	<u>3</u>		
Window Size	<u>1.0</u>	<u>0.5</u>	<u>0.25</u>					
Revised								
Tabu List Size	<u>4</u>	<u>6</u>	<u>5</u>	<u>7</u>	<u>8</u>	<u>3</u>		
Window Size	<u>0.5</u>	<u>0.25</u>						
Early/Tardy (Revised)								
Tabu List Size	<u>1</u>	<u>2</u>	<u>3</u>	<u>4</u>	<u>5</u>	<u>6</u>	<u>7</u>	<u>8</u>
Window Size	<u>1.0</u>	<u>0.5</u>	<u>0.25</u>					
Analyse (Revised)								
Tabu List Size	<u>6</u>	<u>5</u>	<u>7</u>	<u>2</u>	<u>8</u>	<u>4</u>	<u>3</u>	<u>1</u>
Window Size	<u>0.5</u>	<u>1.0</u>	<u>0.25</u>					
E/T Smart								
Tabu List Size	<u>3</u>	<u>4</u>	<u>6</u>	<u>5</u>	<u>2</u>	<u>8</u>	<u>7</u>	<u>1</u>
Window Size	<u>1.0</u>	<u>0.5</u>	<u>0.25</u>					

Table 7-11. Tabu Search Tukey HSD Analysis

Table 7-10 shows that the size of the tabu list makes no statistical difference to the results except when using the Revised E/T TS. Table 7-11 shows that for the Revised E/T TS, there is no statistical difference between tabu list sizes between 1 and 7, however there is a statistical difference between tabu list sizes between 1 and 4 and a tabu list size of 8. The Tukey HSD analysis also shows that for the Revised E/T TS, the larger the tabu list size the poorer the search performed. This indicates that, for this sort of solution space, the use of the Revised TS extension is sufficient to stop the TS from cycling.

The best list sizes for the Standard TS were all within the 7 ± 2 proposed by Glover and Greenberg (1989) while for the Early/Tardy TS the best list size was just 3 which is outside this bound. This indicates that when using an E/T solution space it is more important to have as many neighbours as possible available for selection than with normal solution spaces. The reason for this is that when a job is changed from being early to late or vice versa, large changes are possible to the resulting physical schedule, producing a completely new set of neighbouring solutions, totally unrelated to the previous set of neighbouring solutions. There is therefore no need to make moves tabu for more than a few further moves. As a consequence of this, cycling is not as great an issue with this type of solution space.

The best tabu list size for the Revised TS was 4 while the best list size for the Revised Early/Tardy TS was just 1. This reduction in the best size of the tabu list is not surprising as the implementation of the revised tabu list structure effectively implements a variable length tabu list with a maximum size of $2/(\text{window size})$ and a minimum length of 0. This effect can be illustrated in that a job may not have a chance to be involved in a move until it appears in the random ordering of jobs again, which may be immediate, if the random ordering list gets regenerated, or it could be up to $2/(\text{window size})$ if it appeared first in the random ordering the first time and last the next.

The best tabu list size for the Analyse neighbourhood scheme was a tabu list size of 6, while the best tabu list size for the E/T Smart neighbourhood scheme was 3.

Each window size was found to be significantly different for the Standard, Early/Tardy and E/T Smart TS techniques. In each of these cases the larger the window size, the better the results obtained. For the Revised TS and the Analyse TS no window size was significantly different from any other. This can be explained in that the Revised TS appears to find better quality neighbouring solutions than the standard window based TS. These higher quality neighbours allow the search to converge more quickly to high quality solutions, regardless of the window size.

It was found that, in general, the larger the window size then the larger the tabu list required to achieve the same results. This may suggest that larger tabu lists are required for larger neighbourhoods although there is no statistical interaction between the tabu list length and window size (p-value of 23.2%)

The best of the TS methods were then compared using Games and Howell Tukey HSD. The results are shown in Table 7-12.

	Revised	Standard	E/T Smart	Analyse	Early/Tardy
Mean Deviation from Optimal	<u>0.02%</u>	<u>0.40%</u>	<u>0.89%</u>	<u>1.06%</u>	<u>1.77%</u>

Table 7-12. Games and Howell Tukey Analysis on Tabu Search Methods

Table 7-12. shows that the Revised TS method is superior to the TS techniques based on an E/T solution space. The standard Early/Tardy scheme is the worst of all the schemes examined here, however its results are not statistically worse than the E/T Smart TS or the Analyse TS. It is interesting to note that the results obtained from the two standard techniques are better than those possible from the E/T based TS, as their deviation from optimal is lower than the best results found by the Five Point Insertion Heuristic.

7.6.6. Search Technique Effects

To test the effect of the different search techniques, the best standard and early/tardy search techniques were compared using Games and Howell Tukey HSD. The results are shown in Table 7-13.

Mean	Technique	Parameters
0.02%	Revised TS	(4, 0.25)
0.04%	Standard SA	(1.25, 0.05)
0.09%	Standard GA	(20,0.4)
0.40%	Standard TS	(6,1.0)
0.82%	E/T SA	(0.5,1.5)
0.89%	E/T Smart TS	(4,1.0)
0.95%	E/T GA	(60,0.4)
1.03%	Analyse	(6,0.25)
1.18%	E/T Smart SA	(0.75,1.5)
1.77%	E/T TS	(3,1.0)

Table 7-13 Analysis by Search Method

Table 7-13 again shows how dominant the standard solution space is over the E/T solution space. The Revised TS dominates the search techniques based on the standard solution space, although statistically there is little difference between any of the standard search techniques that have been considered. SA, however, appears to dominate the E/T solution space search techniques, but again there is no significant difference between these except for E/T SA and E/T TS.

7.6.7. Discussion and Conclusions

From our analysis of small sized problems, the standard sequence of jobs solution space has clearly dominated those search techniques based on the E/T solution space. Some of the possible reasons for this are:

1. For problems with a limited number of jobs the solution space is small enough for good search techniques to find the optimal solution quickly. Therefore the advantages of the solution space compression are not large enough to warrant the extra penalty in solution quality incurred through the use of an E/T heuristic.
2. The comparison of solution quality may have been different if a smaller amount of time was allowed for solving each of the problems. The search technique based on the E/T solution space may perform better short-term but long term their solution quality may not be as good. This would reflect the findings in Section 7.4. with the CDD problems.
3. A neighbourhood scheme may need to be developed which accommodates problems which are ‘tight’ in a similar manner to that used for the E/T CDD Date Swap Tabu Search investigated in Section 7.4. This, however, would not be a simple process as there is no clear method of resolving which early jobs should be made late when multiple due dates are involved.

To determine whether this reasoning is correct, these heuristics must be trialed on larger problems which are more realistically sized. This is carried out in the next section.

A question that must still be asked when considering these results is how much deviation from optimal is acceptable? The answer will be case dependant, however all problems have limits on the amount of time available to find a solution. Generally the longer the amount of time spent on a problem, the higher the solution quality. The objective is therefore to find which methods can find the best solution in a given amount of time, rather than the amount of error contained in the solution.

In our experiments with small problems, no one search method was superior to any other method, except that the methods based on a sequence-of-jobs solution space were superior to those based on the E/T solution space.

In conclusion, it is clear from this experiment that for small sized problems, using a sequence of jobs solution space with any of the search techniques available will produce high quality results.

7.7. Large Problems

7.7.1. Introduction

This section examines the use of different search techniques to solve large E/T scheduling problems. Initially 4 randomly generated 250 job problems from each of the problem classes where the tardiness penalty is greater than the earliness penalty were created as outlined in Section 7.2. A further 6 problems from each class were then generated, however these extra problems were only solved by the more promising techniques.

The problems were first solved by each of the different search techniques. The first analysis compares the results which use the same search technique to determine which modifications and/or parameters give the best results. The best searches from each of the different techniques are then compared to determine which search technique is the best for this type of problem. The effect that the starting point of the search has on the quality of the final solution is then explored. Comparisons between the solutions found by these search techniques and other heuristics proposed in the literature are made to determine if these techniques are significantly better than what has already been proposed. A proposal is then given for modifying the Revised E/T TS in a similar manner to the E/T CDD Date Swap TS which was found to be the best TS for the CDD problem. The effectiveness of this modification is then analysed. Finally conclusions from these experiments are then made.

For all of the search techniques tested within this section, each problem was allowed 30 CPU seconds per job on the VAX 6000 mainframe computer, giving a total of 7500 CPU seconds for each problem.

7.7.2. Search Technique Experiment Results

All the search techniques were initially tested using four test problems from each of the six classes of problems where the earliness penalty is less than the tardiness penalty. The results were compared to the best known schedule of these problems to obtain a percentage deviation from the known minimum. These percentages were averaged to give an overall measure of the quality of the solutions obtained from the techniques. The times to the best solution were also averaged to give average solution quality versus average time graphs as seen in Figures 7-6, 7-7 and 7-8. ANOVA analysis was also carried out to determine which search techniques and search technique parameters provided solutions that were statistically different at the 5% level of significance. The details of this analysis are provided in Appendix IV.

Four different types of SA were tested, Standard SA, Early/Tardy SA, Early/Tardy Smart SA and Standard SA starting at the E/T approximation of the EDD schedule. The parameters chosen were taken from the best parameters found for the small problems, however the initial temperature for the standard techniques was lowered as the initial temperature had little significance on the solution quality and it was expected that a high initial temperature for these larger problems may cause the search to wander off into less desirable area of the solution space. As with all “intelligent” search heuristics, the tuning of the parameters is both a science and an art. There is a large subjective element in determining these parameters as there are no right or wrong answers - just ones that may be better under different circumstances. The statistical analysis helps in these decisions and is weighed up with other information about the problem, in this case the larger problem size, to determine the most appropriate value.

A graph of the average time versus the average deviation from the least known minimum for this experiment is shown in Figure 7-6. In this graph only the averages of the best solution found so far are shown. Where the lines end indicates the point where no better solutions could be found by the search.

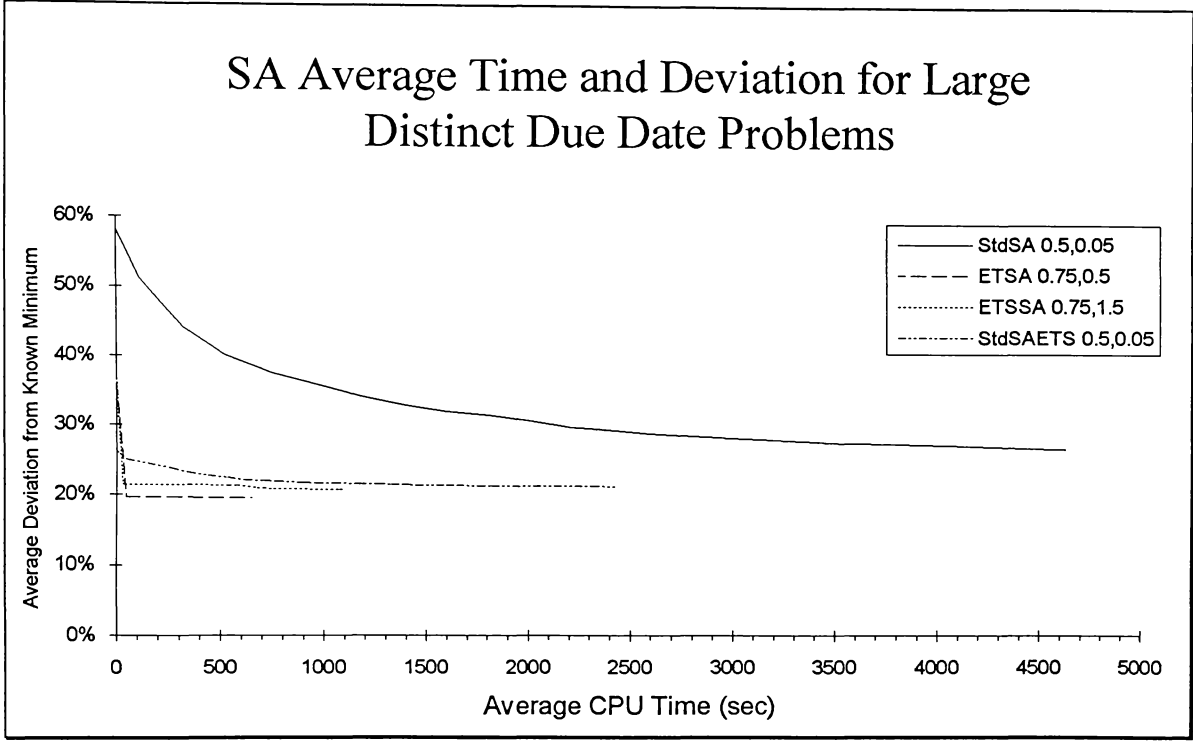


Figure 7-6.

Figure 7-6 demonstrates how quickly the E/T solution space based techniques fall into a local minima, making little progress from this point on. The standard schemes appear to make more gradual progress although they too appear to settle at a point on average over 20% from the smallest known minima. At the 5% level of significance, there is no statistical difference between the Early/Tardy SA, E/T Smart SA or the Standard SA with an E/T EDD starting point. There is, however a significant difference between these techniques and the Standard SA. It appears that by using the E/T heuristic, even just once to determine the starting point in the search can make a significant difference to the overall quality of solutions obtained.

Four different GAs were trialed, two with completely random populations while the other two both had the E/T approximation to the EDD schedule included in their population. The Standard GA of the latter group, also had the EDD schedule included in its population. Each type was trialed with the standard sequence of jobs chromosomes and with the E/T job specification chromosomes. The best parameters of the small problems were used as the basis for the parameters to solve these large problems. The size of the population was set at

60 for the standard problems as the larger population sizes were found to provide better results, although these were not statistically significant. The population size for the E/T based GAs was again set at 100 as it was found in the experiments with small problems that the larger the population the better. A population size of 100 was considered the maximum that could be used so that a comparable number of generations could be made in the time given. A mutation rate of 0.3 was used for the standard techniques as this was found to be the best average mutation rate in the smaller problems, while a mutation rate of 0.4 was used for the E/T based GAs. The mutation rate is slightly lower for these larger problems as a larger solution space is being used and therefore it is less likely to have duplicate members within the population. The average time versus average deviation from least known minimum graph is shown in Figure 7-7.

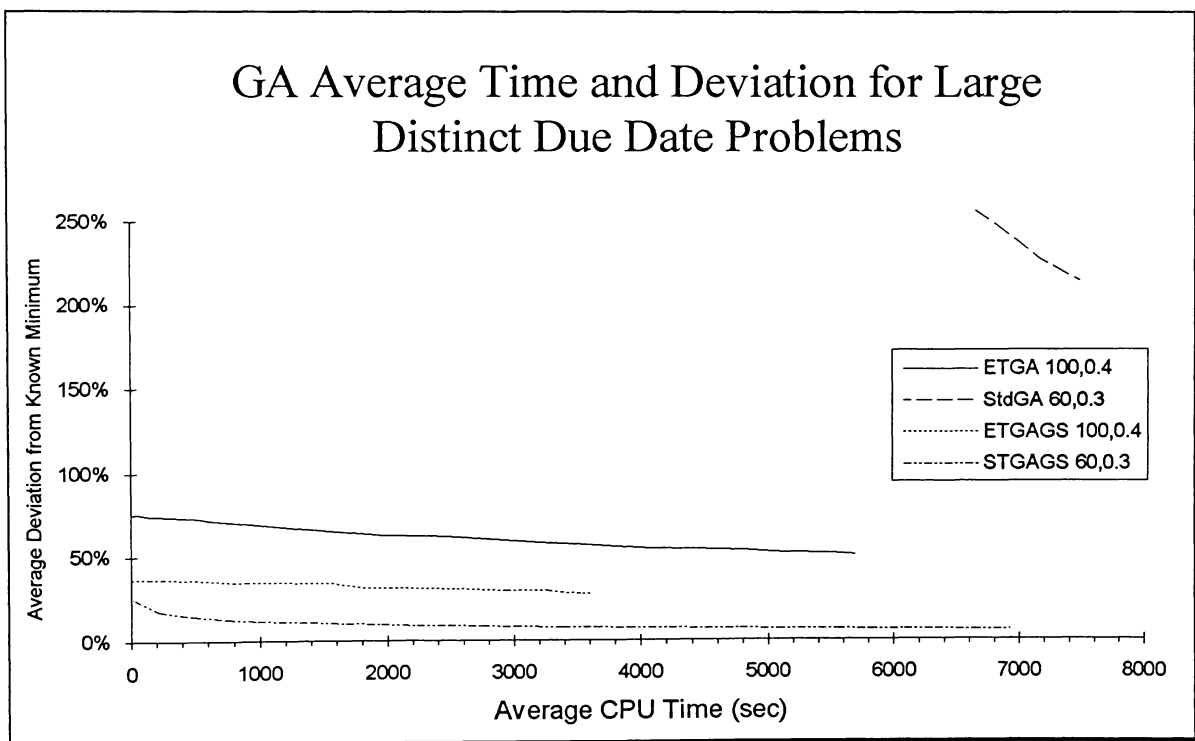


Figure 7-7.

Figure 7-7 shows how the Standard GA with its completely random population is very inefficient and hardly manages to get onto the graph in the time allowed. Of the other GA techniques which were tested, the Standard GA with the EDD schedules as members, provides the best results, however these results are not statistically different, at the 5% level of significance, from the other two E/T based GAs. Of the E/T schemes, the E/T GA with the good starting members produced the best results.

To obtain a more comprehensive analysis of the difference between searches based on sequence of jobs solution spaces and searches based on E/T solution spaces, a more in depth analysis was carried out using TS. TS was chosen for this analysis as the Revised TS

gave the best results of all the searches for the small problems and, even though these were not statistically different from the other search techniques, TS was regarded by the author as the search technique that was the most likely to produce high quality solutions in a small amount of time for large sized problems. Extra work was therefore done in parameter tuning the searches to try to optimise solution quality. Once good sets of parameters were found, these TS methods were then used to solve extra problems in each class to provide stronger statistical validity to the analysis.

Due to the size of these problems the standard TS could not be tested as it took too long to evaluate the entire neighbourhood before moving. Therefore the first technique tested was the Revised TS. For this TS, the tabu list size was set at 15. As the tabu list size was found to be unimportant in the small sized problems and since some research has indicated that tabu list size is related to problem size (Laguna and González Velarde, 1991) a larger tabu list size was used. The window sizes that were trialed were 0.06 (15 jobs), 0.04 (10 jobs), 0.02 (5 jobs), 0.012 (3 jobs) and 0.008 (2 jobs). From our analysis of this experiment, detailed in full in Table A4-3 in Appendix IV, it was clear that the smaller the window size, the better the search performed. Although initially there was no interaction between the size of the window and the type of problem on the solution quality, after approximately 2800 CPU seconds, the interaction between these two factors became significant. This interaction would be caused by the amount of time required by the idle time insertion algorithm for each class of problem. Those problems which required more idle time to be inserted would not perform as many iterations as those problems that did not need idle time, due to the extra time spent calculating and inserting idle time in each neighbouring sequence. The ANOVA analysis shows that those classes of problems with a low tardiness factor, that is those problems requiring more idle time to be inserted, the TS did not perform as well as those classes of problems with a high tardiness factor.

The second technique tested was the Revised E/T TS. For the small problems, it was found that the best tabu list size was 1. To test whether this was still true a test was done with one tabu list size set at 7, the other at 1. The best results were consistently obtained when the tabu list size was set at 1, hence the remaining tests had their tabu list size set at 1. When considering what window size to use, a tradeoff was required. With the small problems, the larger the window, the better the trial solution quality. For large size problems, however, the larger the window size the slower the search will progress and therefore the less effective it will be. The window sizes tested were 0.14 (35 jobs), 0.12 (30 jobs), 0.10 (25 jobs), 0.08 (20 jobs), 0.06 (15 jobs), 0.04 (10 jobs) and 0.02 (5 jobs). Although the results from this experiment, shown in Table A4-4 in Appendix IV, are not conclusive it appears that a window size of 0.08 was the best window size the majority of the time. There is a clear deterioration in the solution quality as the window sizes become smaller than 0.08, however this is not the case for window sizes larger than 0.08 as most of

the time the window sizes 0.14, 0.12 and 0.10 are not in ascending or descending order when ranked in order of solution quality. For the Revised E/T TS the interaction between the window size and the problem was not as strong as the interaction which occurred with the Revised TS, as it only became significant at the 5% level after 4375 CPU seconds. This interaction is not as large as the Revised TS as the speed of the E/T heuristic is not as sensitive as the idle time insertion algorithm to the class of problem being solved. The ANOVA results also show that this type of search is best suited to problems with a high tardiness factor and a large due date range where idle time is more likely.

The next type of TS trialed was the Revised Window TS, starting at the E/T approximation to the EDD schedule. Again a test was carried out to determine the most appropriate tabu list size using the values 7, 15 and 30. From these tests it was found that the tabu list size of 15 produced the best results. Recall that the Revised Window TS has two different window sizes, a source window size and a destination window size. The combinations of these two factors, in order of (source window size, destination window size), trialed were (0.008, 0.4), (0.02, 0.2), (0.02, 0.4), (0.04, 0.1), (0.04, 0.2), (0.12, 0.12), (0.2, 0.04), (0.2, 0.2). From the results of these experiments, outlined in Table A4-5 in Appendix IV, it was found that for the majority of the time a source window of 0.2 and a destination window of 0.04 provided the best results, however the combinations (0.04, 0.2) and (0.12, 0.12) also provided a similar level of solution quality. The common element between these different sets of parameters is that each considers a total of 60 jobs at each iteration. Those techniques with a larger or smaller total number of jobs did not perform as well. Despite this there was no significant difference between the searches using these parameters up to 6250 CPU seconds. After 6250 CPU seconds the top three searches were significantly different from the search using the parameters (0.01,0.4). As with the Revised TS, and for the same reasons, there was a high level of interaction between the problem and the window sizes used as the search seems to perform better on problems with a high tardiness factor.

The Revised Window TS with the E/T Extension was not trialed with the small problems, however from our trials of the Revised Window TS the window sizes were expected to be similar for both of these searches as they use the same neighbourhood scheme. Trials on the tabu list size were carried out with list sizes of 1, 3, 7 and 15. The results from these tests were inconclusive, with different tabu list sizes performing well after different amounts of time. A tabu list size of 7 was used to fine tune the window size, as this was a mid-range value and also because of its popularity in the literature (Glover and Greenberg, 1989), even though it has been discredited as the best tabu list size for all problems (Barnes and Laguna, 1991a). The source window size and destination window size combinations trialed were (0.01, 0.4), (0.02, 0.2), (0.04, 0.1), (0.04, 0.2), (0.12,0.12), (0.2, 0.04). As with the Revised Window TS, the best window sizes were (0.12, 0.12) and (0.2, 0.04). Although (0.04, 0.1) did provide good results initially, it failed to maintain this level of quality through

to the end. It should be noted, however, that no significant difference was found between any of these TS techniques for over 5625 CPU seconds. After this time the TS with the parameters (7,0.01,0.4) became significantly worse than the other searches. The ANOVA results on the solution of different problem classes solved by this type of TS show that there is little difference between this type of search and the Revised Window with the E/T EDD starting point. This is not surprising as the neighbourhood and the search is identical, except for the E/T heuristic check after each iteration movement.

The next TS method trialed was the Hybrid TS, where the search initially searches the E/T solution space using a Revised E/T TS then, after a given number of non-improving moves, it switches to the Revised Window TS. The number of non-improving moves required before switching searches is another variable for this type of search. The Tabu list size for the Revised E/T TS was fixed at 1, as this was shown to be the best tabu list size during our previous experiments, while the tabu list size for the Revised Window TS was determined by a parameter. The tabu list sizes tested were 1, 7, 15 and 20. From these trials, it was found that 20 provided the best results and was therefore adopted for the remaining trials. The source and destination window sizes tested were (0.01, 0.4), (0.02, 0.2), (0.02, 0.4), (0.04, 0.1), (0.04, 0.2), (0.12, 0.12), (0.20, 0.04), (0.2, 0.2). Of these, the window sizes that gave the best results were (0.12, 0.12), (0.04, 0.2) and (0.2, 0.04). The number of non-improving moves was also trialed at 1 and 3. Although different TS parameters were better at different stages of the search, for long term results, the parameters (1,20,0.12,0.12), in the order of (number of non-improving moves, tabu list size, source window size, destination window size), were found to be the best while (3,20,0.04, 0.2) gave just as high quality results from the start. It should be noted, however, that there was no statistical difference, at the 5% level, between most of these techniques. The two techniques that were consistently poor were (3,1,0.04,0.2) and (3,20,0.01,0.4). From the ANOVA results it appears that this type of TS solves problems with a high tardiness factor easily as well as problems with a low tardiness factor and a small due date range. The most difficult class of problem for this TS to solve are those with a low tardiness factor and a large due date range.

The Revised Analyse Search was tested using a tabu list size of 6 and a window size of 0.04. This technique did not, however, provide results which were competitive with the other techniques as the technique tended to be slow, making only a few iterations throughout the entire time allowed. Due to these problems the Revised Analyse Search was not tested further.

The Smart TS was also very slow, due to the amount of schedule analysis required. As a consequence, when it was solving problems with a high tardiness factor, it was not able to

complete even one iteration in the 7500 CPU seconds allowed. This technique was therefore not included in the analysis of large sized problems.

The best TS techniques, except for the Revised Analyse search, were then used to solve a further 6 problems from each class. With the original 4 problems solved and these further 6 problems gave a total of 10 problems from each class from which comparisons could be made. A graph of the best average deviation versus average time is shown in Figure 7-8. ANOVA analysis of these results, summarised in Table A4-8 in Appendix IV, shows that initially only the Revised TS was significantly different from the other search techniques. After 3150 CPU seconds, however, the Revised ET TS is also significantly different from the remaining search techniques. Throughout the majority of the search the Revised Window TS with the E/T EDD start performs the best, however at the very end of the search, the E/T Hybrid has the lowest average deviation, followed by the Revised Window E/T Extension, followed by the Revised Window with the E/T EDD start. There is, however, very little difference between these three techniques. The common elements in all of these searches are that they all use the E/T heuristic at least once during the search and that they all use the standard neighbourhood scheme at some stage during the search. The mix of these two features differentiates these three techniques, however the mix does not appear to make a large impact on the results.

To determine the impact of the starting point on the search, the best Revised TS and the Revised Window TS were also started at the EDD sequence which was optimised using the Ow and Morton adjacency conditions, while the best Revised ET Tabu was started at an E/T approximation to this point. As can be seen from Figure 7-8, in each case these searches started at far better points in the solution space than their E/T approximations to the EDD sequence. However only the Revised TS benefited overall from the use of this superior starting point as this search started from the EDD schedule, not the E/T approximation to the EDD. The Revised Window TS and the Revised ET TS produced worse results, on average, than their E/T EDD starting point counterparts. From these results, it may be deduced that the use of this type of starting point may in fact force the search into a deeper local minima, which in turn takes the search more time to climb out, hence the poorer performance.

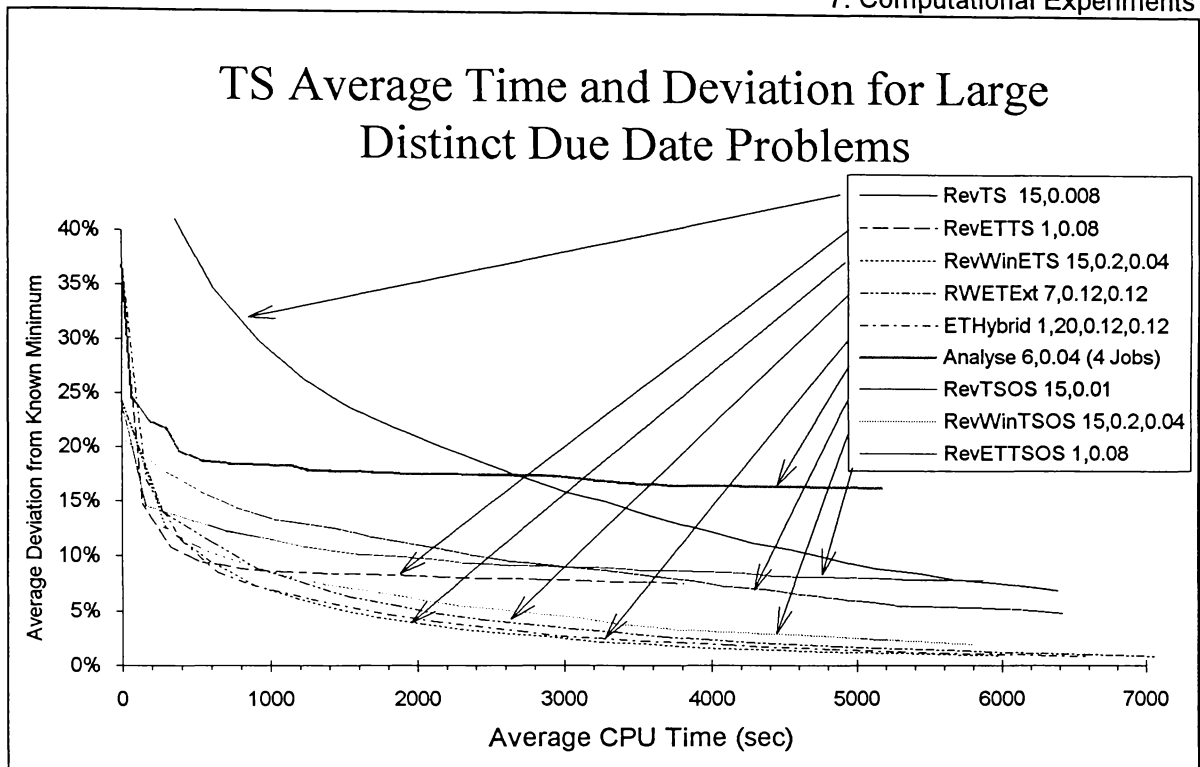


Figure 7-8.

7.7.3. Search Technique Effects

The best results from each of the search technique tests carried out in Section 7.7.2. were analysed to determine the effect that the type of search and the type of solution space used had on the overall performance of the search. The techniques included in this comparison were: Standard SA using an E/T EDD starting point, Early/Tardy SA, Standard GA with EDD and E/T EDD chromosomes, E/T GA with E/T EDD chromosome, Revised TS, Revised E/T TS, Revised Window TS with E/T EDD Starting point, Revised Window TS with the E/T Extension and the E/T Hybrid TS.

Figure 7-9 shows the average deviation versus average time graph for all of the main search techniques considered.

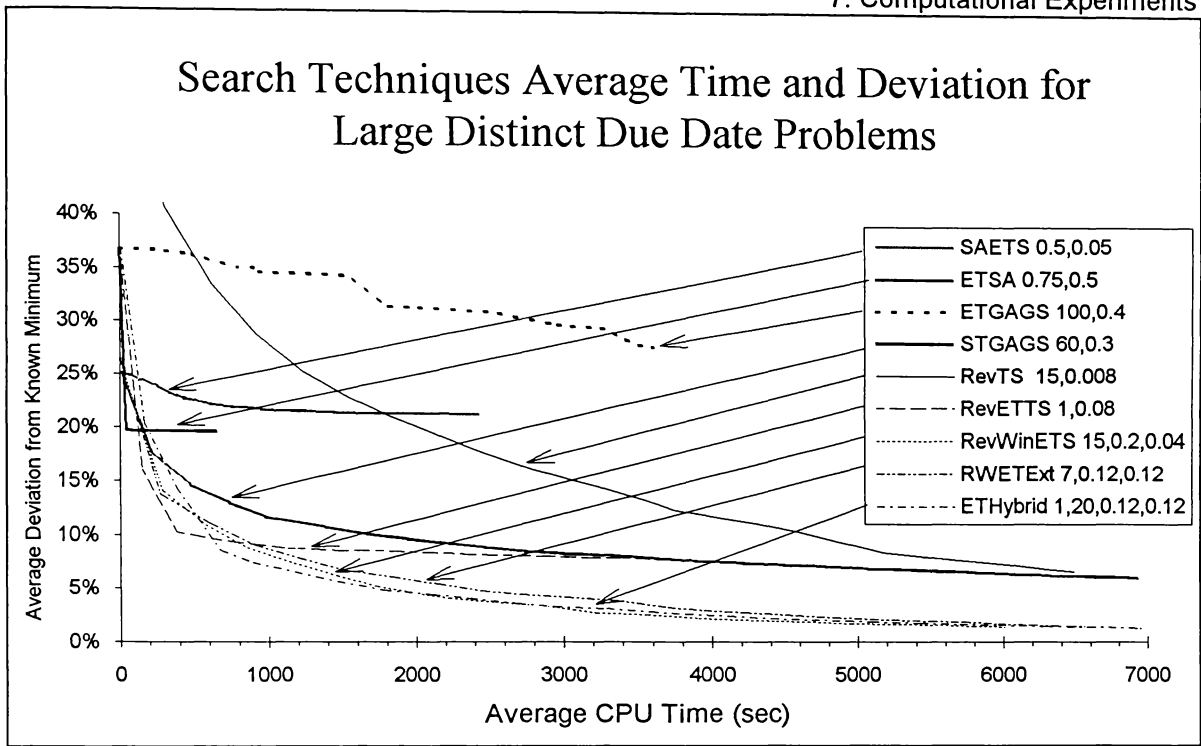


Figure 7-9.

A Games and Howell Tukey HSD analysis was used to determine whether the results obtained from each of the search techniques were statistically different. These results are presented in Table 7-14. The best group of search techniques were those that used a combination of both the E/T solution space and standard solution space. The worst techniques were the E/T GA, Standard SA with E/T EDD starting point and the E/T SA. The Revised TS, although initially a very poor performer, progressively finds better solutions so that by the end of the time allowed, it produced results that are on average under 7% deviation from the best known minimum.

Mean	Technique	Parameters
1.41%	Revised E/T Extension TS	(7, 0.12, 0.12)
1.43%	Hybrid TS	(1,20,0.12,0.12)
1.46%	Revised Window TS E/T EDD Start	(15,0.2,0.04)
6.16%	Standard GA	(60, 0.3)
6.86%	Revised TS	(15,0.008)
7.73%	Revised E/T TS	(1,0.08)
19.69%	E/T SA	(0.75,0.5)
21.29%	Standard SA E/T EDD Start	(0.5,0.05)
27.84%	E/T GA	(100,0.4)

Table 7-14. Games and Howell Tukey Analysis Between Techniques at the End of the Search

From these results it is apparent that TS provides the most 'powerful' search technique, in that even when it starts at poor quality solution it is still able to find solutions which are

better than those found by SA and GAs which used higher quality starting points. The examination of the effect of starting points on the quality of the final solution quality is carried out in Section 7.7.4.

7.7.4. Effect of Using Different Starting Points

Barnes and Laguna (1991a) note that “a procedure for obtaining one or more ‘good’ starting solutions, as opposed to a random start, is beneficial to the eventual success of the TS method”. Grefenstette (1987), however, discovered that GAs, which had initial population members generated by a heuristic, found good solutions quickly but fully random populations produced better results in the long run. In Section 7.7.2. it was discovered that the use of a better starting point with TS did not necessarily mean a better overall result. In this section we examine whether it is beneficial to use a good starting point, based on the EDD sequence of jobs, or if by using a random starting point the quality of the final solution will be the same.

The hypothesis for this experiment is that ‘the starting point of the search does not have any effect on the quality of the final solution’. To test this hypothesis each of the randomly generated large problems are tested using the search techniques with the same parameters but beginning from different starting points. The end results were then compared to see whether there was any statistical significance between the final solution quality and different starting points. Another important issue examined in this section is to explore the effect that the solution space has on the importance of the starting point for a search.

To compare the effect of starting points when using SA, the standard SA technique starting at the EDD schedule, was compared to the standard SA starting at the best of the EDD schedule or the E/T approximation to the EDD schedule. The graph of these two searches is shown in Figure 7-6. From this graph and the ANOVA results in Appendix IV, it can be seen that while at the beginning of the search these two techniques are statistically different, after approximately 900 CPU seconds this significance is lost. It must be noted, however, that although the statistical difference is not significant, the SA starting at the better starting point always produces better average results than the one that did not.

The tests involving GAs compared the standard GA and a E/T GA with a completely random population with a standard GA which had as one of its members the EDD sequence with idle time optimally inserted and another member at the E/T approximation to this EDD schedule and an E/T GA which has as one of its members the E/T approximation to the EDD schedule. The results from this experiment, detailed in Table A4-2 in Appendix IV, showed that the GAs which started with the EDD population members always had a lower mean deviation from optimal, with the standard GA having a lower mean than the E/T GA, however the difference between the standard GA and E/T GA with a good initial population

and the E/T GA using random population is not statistically significant, yet all of these GAs are statistically different to the standard GA using a random start.

For testing the sensitivity of TS to the starting point used, three different starting points were used. These were random starting points which were good, random starting points which were bad and starting points which were generated by heuristics. Three different types of TS were used for this experiment. These were Revised Window TS, Revised E/T TS and Revised E/T Extension TS. Each of the tabu searches used the same set of parameters, which were a tabu list size of 15, a source window size of 0.04 and a destination window size of 0.2. The good and bad starting points were created by randomly generating 100 different schedules and selecting either the best or the worst schedule from those generated respectively. The two heuristic starting points were generated by using an EDD schedule and/or E/T approximation to the EDD schedule. A graph of the results, except for the Revised Window TS random starting point searches whose performance was too poor to appear, is shown in Figure 7-10.

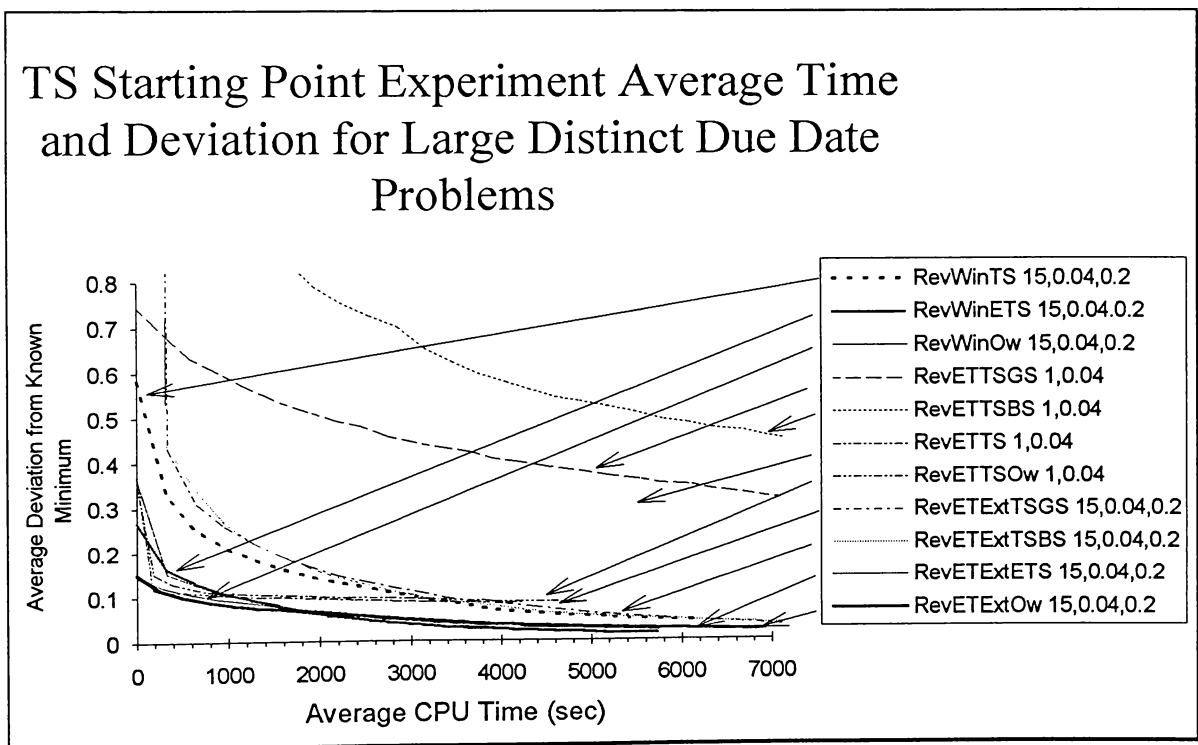


Figure 7-10.

A Games and Howell Tukey HSD between the different methods was used to determine which methods gave final results that were significantly different at the 5% level. These results are presented in Table 7-15. The Tukey HSD results at each of the time intervals are summarised in Table A5-1 in Appendix V.

Mean	Technique
1.515%	Revised Window TS E/T EDD Start
1.985%	Revised Window E/T Extension E/T EDD Start
2.299%	Revised Window TS Ow EDD Start
2.362%	Revised Window E/T Extension Ow EDD Start
3.530%	Revised Window E/T Extension Good Start
3.652%	Revised Window E/T Extension Bad Start
4.462%	Revised Window EDD
8.347%	Revised E/T EDD Start
8.499%	Revised E/T Ow EDD Start
32.159%	Revised E/T Good Start
45.580%	Revised E/T Bad Start
158.353%	Revised Window Bad Start
162.073%	Revised Window Good Start

Table 7-15. Games and Howell Tukey HSD between Starting Points

From the results in Table 7-15, it can be seen that the Revised E/T Extension TS suffered the least from starting at poor quality solutions. Starting this search at either a good or a bad random starting point gave results that were worse than starting the search at the EDD schedule, but this difference was not statistically different. From the graph, it can be seen that starting the search by using the E/T EDD approximation did enable the search to find high quality solutions more quickly, however in the long run the difference was minimal, as the E/T element within the E/T Extension TS brought the schedule into line with the EDD starting point relatively quickly, acting as a diversification strategy for the search. As with the Revised Window TS in Section 7.7.2. the Revised E/T Extension TS did not benefit from starting at the optimised Ow and Morton Starting point with the average results being worse than starting at the E/T EDD schedule.

The Revised Window TS, on the other hand, is sensitive to the starting point given to the search. Although there is no statistical difference between the search using the good random start and the search using the bad random start, there is a large statistical difference between the use of a random starting point and a starting point generated by the EDD heuristic. The TS which started at the E/T EDD approximation and the Ow and Morton EDD also produces results that are statistically better than the search which started at the EDD schedule.

The Revised E/T TS also appeared to be affected slightly by the use of good and bad starting points. The Revised E/T techniques which used a random starting point were not, however, statistically different from the Revised E/T which started at the E/T approximation of the EDD schedule. The graph does illustrate that the use of good quality starting points, such as the E/T EDD and the Ow and Morton EDD, does give better overall results than the use of random starting points.

From our results it can be concluded that the starting point that is used for a search technique can influence the quality of the final solution. In all but one case, the quality of the final solution was directly correlated to the quality of the starting point of the search, however it was also found that some search techniques are more sensitive to starting points than others. In all the searches trialed, the use of a highly optimised starting point, as in the Ow and Morton EDD starting point, in the search produced results that were worse than those when the E/T EDD starting point was used. Although this difference was never statistically significant it does show that caution is necessary when using search techniques not to over optimise the starting point as this may slow down the search. Search techniques where the search incorporated a diversification strategy which enabled the search to find areas in the solution space with high quality solutions, like the Revised E/T Extension TS, were the least sensitive to starting point differences. The searches based on the standard sequence of jobs neighbourhood were especially susceptible to poor starting points, as the search often makes only small incremental progress to a high quality solution, which is beneficial when the search is close to the minima, but inefficient when far from the minima. By using heuristics, such as the E/T approximation to the EDD schedule, the performance of these searches was improved considerably, as this provides a good area of the solution space in which to search. An important aspect to note is that a good starting point becomes more important as the size of the problem is increased. In large problems good starting points and any subsequent gains which can be made quickly become vital as there is unlikely to be sufficient time to explore the solution space thoroughly.

7.7.5. Comparison of ‘Intelligent’ Search Heuristics with Other Heuristics

The question that must now be asked is how much better are the results from intelligent search heuristic, such as SA, GA or TS, compared to the results from other heuristics - such as those proposed by Fry, Armstrong *et al.* (1987), Keyser and Sarper (1991) or simply dispatching rules with idle time optimally inserted.

To answer this question the eight standard heuristics were used to solve the four large problems from each class, and these results were then compared to the E/T Heuristic and the search-based heuristics. The heuristics used were the:

- Fry Armstrong *et al.* (1987) heuristic,
- Keyser and Sarper (1991) heuristic,
- Keyser and Sarper (1991) heuristic with the Fry, Armstrong *et al.* (1987) idle time insertion algorithm,
- EDD with idle time optimally inserted,
- EDD with idle time optimally inserted optimised using the Ow and Morton adjacency conditions,
- Smallest Slack sequence (SLK) with idle time optimally inserted,

- E/T approximation to the EDD schedule,
- E/T approximation to the SLK schedule,
- Revised TS starting at the EDD schedule, and
- Revised Window TS starting at the E/T approximation of the EDD schedule or EDD schedule, whichever is better.

Games and Howell Tukey analysis was then carried out to determine which methods produced results that were significantly different from each other at the 5% level of significance. These results are summarised in Table 7-16 and the differences are graphed in Figure 7-11.

Mean	Technique	Parameters	Ave Time Required (CPU sec)
1.47%	Revised Window TS E/T Start	(15, 0.2, 0.04)	6012
6.86%	Revised TS	(15,0.008)	6488
16.18%	Keyser with Fry Idle Time Insertion		3283
17.99%	EDD Opt. with Ow and Morton		16
36.69%	E/T EDD		16
42.56%	E/T SLK		9
58.17%	EDD		15
59.07%	SLK		8
106.35%	Fry		3
151.43%	Keyser		878

Table 7-16. Games and Howell Tukey Analysis between Heuristics

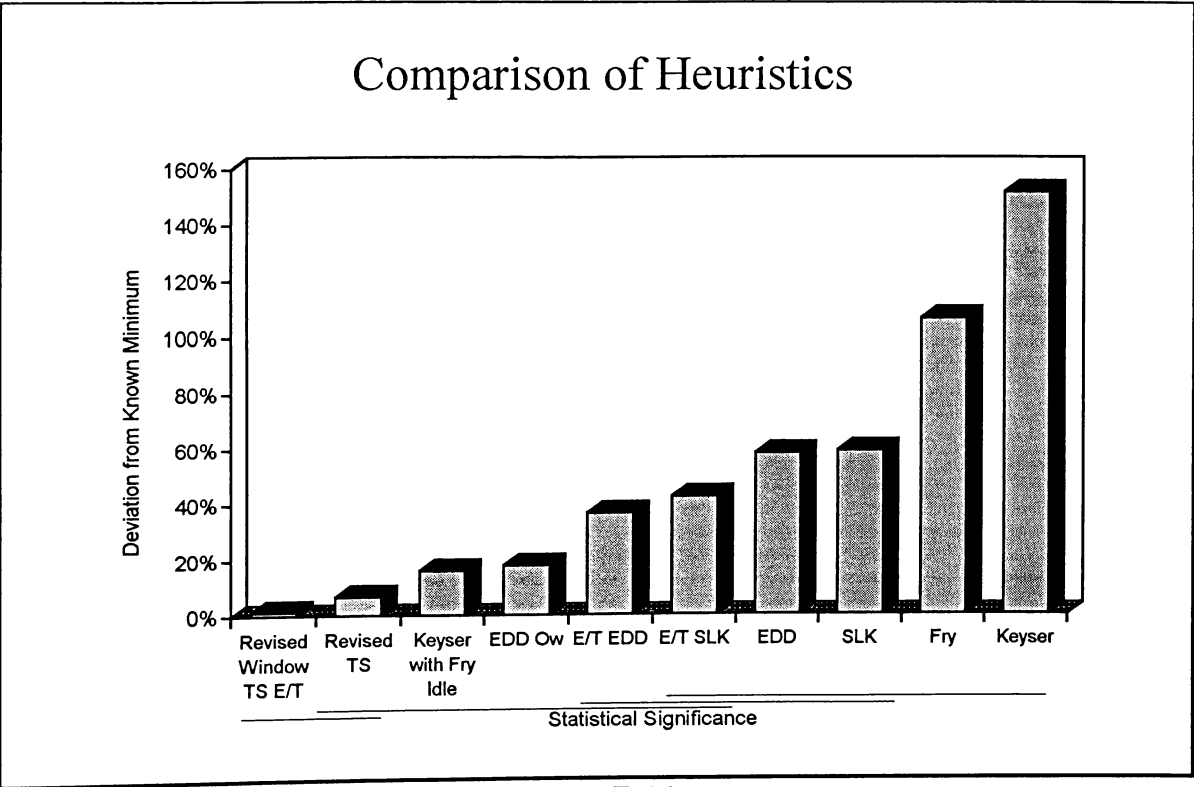


Figure 7-11.

Table 7-16 and Figure 7-11 reveal several important facts:

1. With the exception of the EDD sequence modified by the Ow and Morton adjacency conditions, the standard heuristics which deal with these problems do not perform well with mean results which are up to 1½ times higher than the least known minimum.
2. Idle time insertion is critical for this type of problem. This is demonstrated by the large increase in performance by the Keyser heuristic when it includes an optimal idle time insertion algorithm rather than the approximate techniques originally suggested. When an optimal insertion algorithm was used, however, the average time increased markedly with the Keyser algorithm, in some cases, taking over 8000 CPU seconds to run - longer than the time allowed for the TS search.
3. Although the E/T approximation of the EDD schedule is better than the EDD schedule, this difference is not significant at the 5% level.
4. The implementation of an intelligent search technique, such as TS, produces significantly better results.

From the points outlined above we can conclude that standard heuristics can provide a quick solution to the problem, however these are not of a very high quality. The best all round standard heuristic was the EDD sequence modified by the Ow and Morton adjacency conditions. The Keyser algorithm with optimal idle time produced better results but was considerably slower. Search techniques provide a means of improving the solution quality significantly, however this is at the expense of computational time. The Revised Window E/T Start TS takes approximately 200 seconds to achieve the same deviation from the best known solution as the Ow and Morton technique, and a further 660 seconds to halve this deviation.

To explore the underlying reasons for the performance differences between these heuristics, one 30 job problem from each of the classes G to L, that is those classes where the earliness penalty is less than the tardiness penalty, was generated and analysed. Graphs of the costs and the schedules produced by each of these techniques are presented in Appendix VI. The final costs of the schedules found by each of the techniques are shown in Table 7-17 where the entries that are in bold are the best schedules found by the nine heuristics.

Technique	G	H	I	J	K	L
Revised Window TS E/T Start	57112	7582	10358	117805	26350	42405
Revised TS	57112	7582	10358	118460	26350	42405
Keyser with Fry Idle Time Insertion	60701	8191	12622	123340	26350	43369
EDD Ow	64934	8191	11986	122267	26350	43369
E/T EDD	62319	8276	11578	122171	27356	50705
E/T SLK	62319	8276	10701	122145	27356	50705
EDD	80514	9900	17676	142911	35158	52854
SLK	80730	10474	17985	142536	36394	54712
Fry	87407	24192	20251	140671	45138	56436
Keyser	61175	43636	38594	123340	26350	43369

Table 7-17. Schedule Costs of 30 Job Problems

The impact of idle time can be easily seen by examining the schedules and cost graphs in Appendix VI and Table 7-17. The Keyser heuristic, which does not employ an optimal idle time insertion heuristic, is severely penalised when solving problems when the tardiness factor is low and the due date range is loose, that is classes H and I. For classes J, K and L where the tardiness factor is 0.6 the schedules produced by the standard Keyser heuristic are identical to those produced if idle time was inserted. The performance of this heuristic is therefore dependent on the tardiness factor of the problem being solved. This explains the high quality results obtained by Keyser and Sarper (1991) but the relatively poor performance of the results here, as the type of problem used in your analysis has a large influence on the results obtained. It may be appropriate, therefore, to use different idle time algorithms which are dependant on the problem being solved. If it is a problem with a high tardiness factor then no idle time insertion may be appropriate. If it has a low tardiness factor and a low due date range, then idle time insertion just at the start of the schedule may be more appropriate. If it has a low tardiness factor and a high due date range then full idle time insertion will probably be necessary. Further research is needed to determine the levels for the tardiness factor and the due date range where these rules apply. This finding makes us conclude that the solution method used to solve a problem should be dependent on the problem's tardiness factors and due date range. The solution method could therefore exploit the particular features of the problem being solved, therefore increasing its efficiency. This is another area for further research.

Table 7-17 also shows that for this size problem, there is little difference between the Revised Window TS with an E/T starting point and the Revised TS.

By examining the schedules that have been produced in Appendix VI, the value of the E/T heuristic is demonstrated. To measure how "close" a schedule is to the best known schedule, it is possible to count the number of insert moves to get from the current schedule to the best known schedule. This was done to compare the E/T EDD schedule and the

EDD schedules and to determine how far away each was from the least known minimum. These results are presented in Table 7-18.

Problem Class	E/T EDD	EDD
G	4	24
H	6	5
I	1	8
J	8	18
K	2	6
L	10	9

Table 7-18. Insert Moves from Optimal

From these results it is clear than in all but two cases, the E/T heuristic produced a schedule that is closer to the final solution, in terms of insert moves, than the EDD schedule. In some cases, especially problem class G, this reduces the number of moves required substantially.

Another means of assessing the quality of the schedule is to look at how it performs under some other criteria which, although it is not the main objective being optimised, may be minor or informal objectives that are held by the scheduler for what ever reason. Table 7-19 shows the number of tardy jobs in the schedules produced by each of the heuristics.

Technique	G	H	I	J	K	L
Revised Window TS E/T Start	7	6	5	21	19	22
Revised TS	7	6	5	21	19	22
Keyser with Fry Idle Time Insertion	7	6	5	21	19	26
EDD Ow	8	6	5	21	19	26
E/T EDD	8	3	4	20	18	24
E/T SLK	8	3	4	20	18	24
EDD	8	6	5	20	20	27
SLK	8	6	5	20	20	27
Fry	7	6	5	21	20	26
Keyser	7	2	4	21	19	26

Table 7-19. The Total Number of Tardy Jobs in the Final Schedule

The results in Table 7-19 are inconclusive. All the techniques obtained at least one minimum, however the better techniques were the E/T EDD and the standard Keyser heuristic with three minimums. It is not surprising that the standard Keyser heuristic performed so well with this objective as it does not insert idle time and therefore it has much more slack to avoid a job becoming late. From these results it could be suggested that the number of tardy jobs is similar across all of the heuristics.

Table 7-20 shows the schedules being evaluated using the criteria of the maximum unweighted tardy distance.

Technique	G	H	I	J	K	L
Revised Window TS E/T Start	715	115	214	2106	1030	1444
Revised TS	715	115	214	2006	1030	1444
Keyser with Fry Idle Time Insertion	597	130	213	2006	1030	1444
EDD Ow	822	130	214	2006	1030	1444
E/T EDD	822	115	214	2006	1061	1472
E/T SLK	822	115	214	2006	1061	1472
EDD	749	130	135	1613	457	461
SLK	753	180	135	1613	457	461
Fry	605	1268	342	1613	940	878
Keyser	573	47	209	2006	1030	1444

Table 7-20. The Maximum Tardy Distance in the Final Schedule

The results in Table 7-20 are also not surprising, as the heuristics customised to solving the generalised E/T scheduling problem do not perform as well as the simple dispatching heuristics, such as the EDD rule. This is due to the optimising heuristics moving, usually a low penalty, job away from its due date, sacrificing this job so that a larger number of, usually more important, jobs can get closer to their due date.

This experiment again confirms the superiority of search techniques when compared to standard heuristics, however it does show that if the objective is not accurately represented by minimising the sum of the weighted early and tardy distances then a search heuristic may not provide the best result unless the objective function can be fully defined and the solution space searched according to this true objective function.

It must also be remembered that this experiment involved just one problem from each class to enable us to understand these different heuristics better. These results are therefore not generalisable to all problems in these classes but may indicate areas of further research, as discussed in Chapter 9.

7.7.6. Date Based E/T Neighbourhoods

The graph of the Revised E/T TS in Figure 7-9, closely resembles the graph of the CDD Revised TS in Figure 7-1 as it drops quickly to a certain level then makes little progress. In the CDD case, it was found that by using a ‘date swap’ neighbourhood the solution quality of the search increased dramatically when compared to standard search techniques. Recall that when using a ‘date swap’ neighbourhood scheme, if a job is changed from being late to being early and, due to there being an excessive number of early jobs, this is not possible then the jobs closest to the due date automatically become late, hence eliminating the majority of invalid areas within the solution space.

The question that needs to be asked is whether it is possible to provide a similar date swap procedure for the distinct due date problem. There are several problems in developing such an algorithm, due to the fact that there is no longer a strictly correct WLPT/WSPT ordering of jobs in our schedule. With the CDD problem the jobs with the highest earliness penalty were changed to being late and the number of jobs required to make enough room was easily calculated. With the distinct due date problem, the issues become more complex. These include:

1. Which due date is the one to start swapping jobs from?
2. Which jobs should be flipped - the ones closest to the due date or the jobs with the highest earliness ratio? These are one and the same for the CDD problem but not for the distinct due date problem.
3. How should the absorption of idle time be handled in the calculation of how many jobs are required to become late?

A simple date swap distinct due date TS was built which used the due date of the late job being made early and by flipping the necessary jobs earlier than this date that have an E/T job specification of early to being late. The idle time directly after the group of jobs where the late job's due date lies and all idle time encountered as early jobs are being swapped is also taken into consideration when determining the number of jobs required to become late.

The resulting search was then compared to the standard Revised E/T TS both in terms of overall solution quality, as shown in Figure 7-12, and in terms of solution quality by class of problem shown in Figure 7-13.

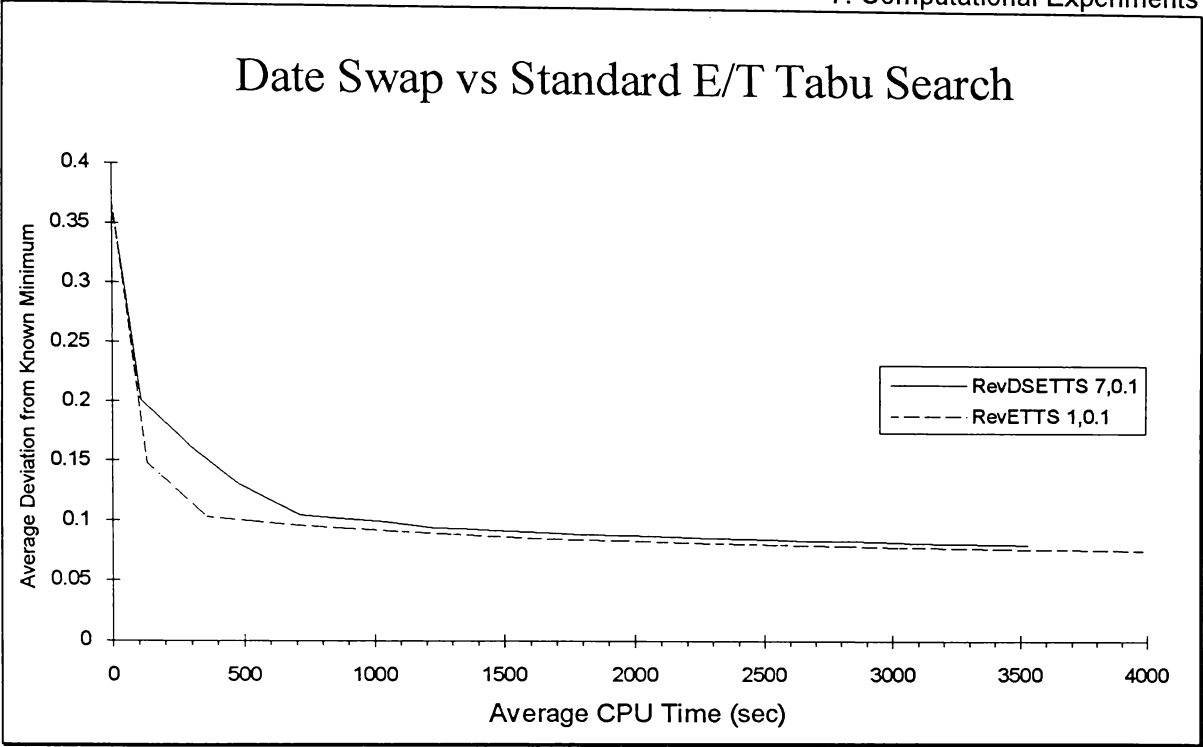


Figure 7-12

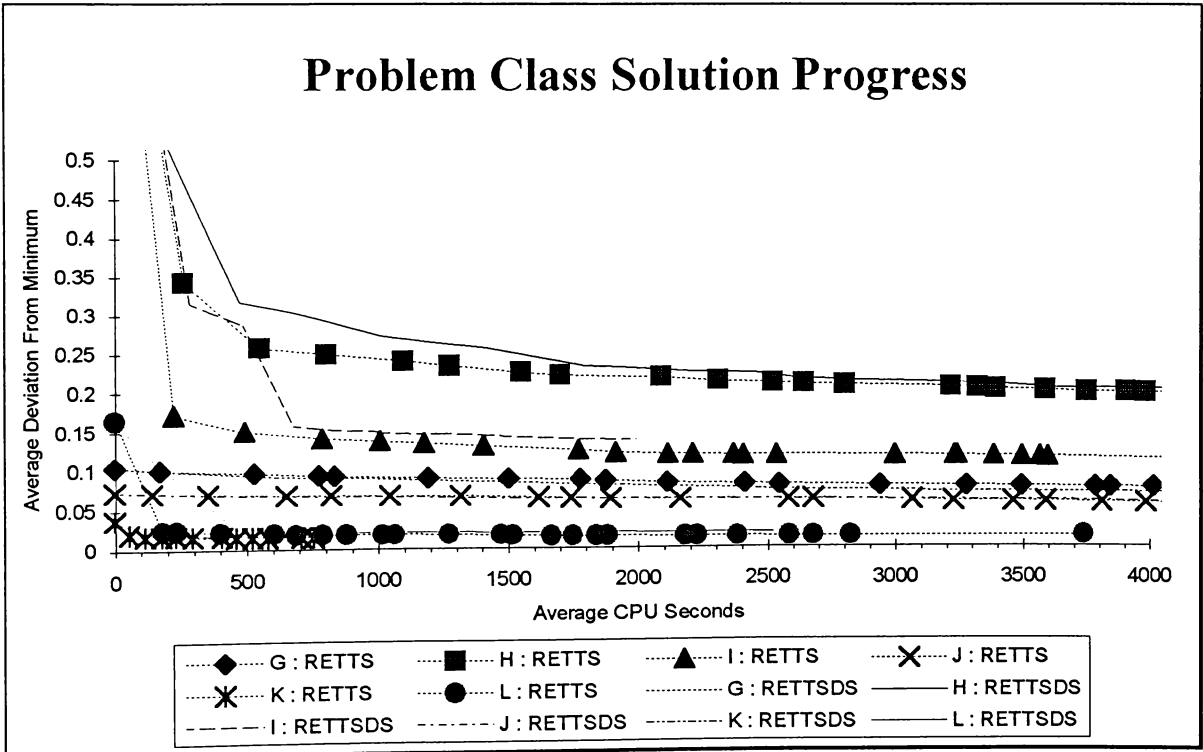


Figure 7-13

As can be seen in both Figure 7-12 and Figure 7-13 this form of Date Swap TS actually performs worse than the standard Revised E/T TS approach.

The reason for this is that the Date Swap TS is substantially slower than the Revised E/T TS on problems with a high tardiness factor. Looking at the number of iterations performed

by each of these searches, the Revised E/T TS performed between 1130-1187 iterations in the 7500 CPU seconds for problems with a high tardiness factor, whereas the Date Swap algorithm performs between 7 and 18 with one problem completing 613. The time penalty involved in calculating the jobs to be swapped was too great to offset the improvements in the solution space. However by implementing the three questions outlined above differently, a better, faster technique may be able to be formulated. Further research is therefore required to find out which of these elements is the most important and to build a search method to accommodate this element.

7.7.7. Conclusions

In this section a set of randomly generated 250 job problems were solved using various search techniques and standard heuristics. From these experiments it was found that search techniques that were based on the E/T solution space found good areas of the solution space in which to search and quickly dropped into a local minima, however often these searches had difficulty in progressing beyond this point. On the other hand, searches which used the sequence of jobs solution space, made consistent progress through out the search, however it suffered greatly from not being able to find good areas of the solution space quickly. Techniques which used the E/T solution space, in some way, initially then subsequently used the sequence of jobs solution space provided the best results. This result was demonstrated extensively when the effect of different starting points was explored.

Of the different types of search techniques it was found that searches based on the Revised Window TS are the most efficient and robust search techniques for large problems. Each of the SA methods performed poorly in these tests, while GAs using a sequence of jobs chromosome structure performed well, however the E/T GA gave the worst results of all the searches. Extensions to these base techniques may increase their performance, however, TS continually provided good quality results efficiently. When the search heuristics were compared to the other E/T scheduling heuristics proposed in the literature, it was clear that the search based techniques were superior.

A problem with the E/T solution space is in solving problems that have a high tardiness ratio. This tends to produce a 'bottle neck' for jobs trying to be scheduled early and, as a result, causes large areas of the solution space to become invalid. This invalid area subsequently makes it difficult for search techniques to progress effectively. An attempt to remedy this problem by using a modified date swap procedure, did not resolve this problem, hence further research is required in this area.

7.8. Conclusions

In this chapter many different experiments have been carried out to determine the most appropriate method of solving the generalised distinct due date E/T scheduling problem.

Tests of idle time insertion algorithms showed that the algorithm developed by Fry, Armstrong *et al.* (1987) produced optimal results more quickly than the algorithm proposed in Chapter 4. It was therefore adopted throughout the remaining experiments whenever idle time insertion was required.

The benefits of using an E/T solution space were then explored by using it to solve the CDD problem, as this eliminated the 'heuristic factor' from the analysis. Results from this found that the search based on the E/T solution space dropped quickly to a local minima but made little progress from there. The reason for this was traced to classes of problems with a low tardiness factor and was resolved by using a 'date swap' search which automatically rescheduled the early jobs which were close to the due date to being late as the need for 'early job' space arose.

With the credibility of using an E/T solution space established in the CDD experiments, this type of solution space was then tested by using the two E/T heuristics developed in Chapter 5. From these experiments it was found that the Five Point Insertion heuristic produced higher quality results and was faster than the Dynamic Loop heuristic, hence the Five Point Insertion heuristic was used in the remaining experiments.

Tests on small problems revealed that the search techniques based on the sequence of jobs solution space outperformed the search techniques based on the E/T solution space. When the same trials were conducted on large problems, however, the E/T solution space provided a means of finding a good area of the solution space to search, however searches using the E/T solution space by itself, as in the CDD case, provided some initial gains but little long term improvement. The best results were obtained by using a hybrid of the two solution spaces. These experiments also revealed that TS was the most robust of the search techniques, with it consistently providing the best results. Starting points were also found to be critical to the eventual success of the search, with a direct correlation between the quality of the starting point and the ending point of the search. However it was also discovered that the use of highly optimised starting points may in fact slow the search down due to the deeper local minima that the search has to get out of before making progress. It was also discovered that searches that incorporated diversification strategies, in this case the use of the E/T heuristic, were far less sensitive to the starting point used. Generally searches based on the E/T solution space were less susceptible to the starting point than those based on the sequence of jobs solution space.

The tardiness factor was found to be a major influence in the success of some of the search techniques, especially the E/T based searches. This was due, primarily to the problems with a high tardiness factor making much of the E/T solution space invalid and therefore trapping

the search in by infeasible solutions. The solution used to solve this problem for the CDD case, that is the Date Swap neighbourhood, did not work for the distinct due date case because the implementation of it had a high time penalty and subsequently the search did not progress very fast at all. The development of faster searches which do address this problem is a promising area for future research.

Experiments with existing heuristics showed that the results produced by the search techniques were significantly better than those results produced by the heuristics of Fry, Armstrong *et. al* (1987) and Keyser and Sarper (1991). Another fact underlined in these experiments was the importance of having idle time optimally inserted into the schedule. This was shown with the original Keyser and Sarper (1991) heuristic having an average deviation from optimal of 151%, while the same heuristic with idle time inserted optimally had an average deviation from optimal of just 16%. Experiments with 30 job problems revealed that search techniques were superior in terms of E/T costs, however when the objective criteria was changed, the results from the simpler dispatch heuristics were slightly better, hence providing a warning about the need to accurately define the objective function for search techniques as they may be more prone to error in objective function definition.

In conclusion the E/T heuristic and the corresponding E/T solution space can be used effectively in solving the distinct due date E/T scheduling problem. This technique is especially useful as a diversification tool for a search, allowing the search to find good areas of the solution space quickly.

8. Determining the Early/Tardy Ratio

8.1. Introduction

In their review of scheduling theory, McKay *et al.* (1988) highlight that many scheduling models do not reflect the reality of the job shop. E/T scheduling has attempted to make scheduling theory more relevant for manufacturers which are in a “Just-In-Time” environment however, even in this more general model, there are still many assumptions made which restrict the usefulness of the theory when solving practical problems.

One real world question which may be asked of E/T Scheduling is, “How are the early and tardy penalties determined?”. To date it has always been assumed that these figures are freely available and known with certainty. Often, however, this is not the case. With the generalised E/T scheduling model it is not the values of the individual penalties that need to be correct but the ratio between the early and tardy penalties.

In this chapter we first examine some of the possible components that make up the earliness cost and the tardiness cost of a particular job. We then determine how much error can be sustained in the ratio between these two costs before accuracy is significantly reduced.

8.2. Early/Tardy Cost Components

From informal discussions with various people in industry - including managers, schedulers and an industry consultant - an initial assessment of the components that make up the early and tardiness penalties was made.

The earliness penalty is often easier to determine as its costs are more easily identifiable. Some of the costs which can be included are:

- Cost of Capital investment in the finished goods inventory,
- Cost of warehousing/storage,
- Cost of deterioration in the product.

Determining the tardiness penalty is far more difficult and requires more care. However this does not diminish the fact that these cost do exist, that they are non-zero and that they are important. The major cost of a late delivery to a customer is the loss of goodwill the supplier has with the customer. The extent of this loss will depend on many different factors, some of which are:

- The perceived urgency of the product for the customer.
- The availability of other suppliers of the product. If there is more than one supplier, then it is more important to deliver on time, so that customers will not switch to other suppliers.

8. Determining the Early/Tardy Ratio

- The type of product. If it is a subcomponent of a product which is not available elsewhere, then the final customers are more likely to wait. Hence there may be less demand from the manufacturer.
- The perceived value of the customer to the company. This can be measured in terms of the proportion of turnover the customer buys, or the customer could potentially buy.

It is clear, however, that determining the tardiness penalty is very subjective and the value it is given will vary among different people in an organisation. For example, those in the marketing/sales department often make the tardiness penalty much higher than those in production as they are held responsible if orders are not delivered. It also falls to marketing personnel to try to convince the customers to continue buying from the company. In production, as they often are wanting to minimise inventories, a higher earliness penalty and a lower tardiness penalty will be proposed. This conflict is common in order-based production companies, and is ultimately handled by an intermediary - the production scheduler.

The determination of a single early cost and a single tardy cost, as proposed in some of the research on E/T scheduling, such as Fry, Armstrong *et al.* (1987), can be shown to be unrealistic in practice. While the earliness factor will vary with the cost, size, and type of product; the tardiness factor will vary with the customer, the market, and urgency of the product. The earliness penalty is therefore determined more by product-based variables, while the tardiness factor is determined by more market-based, or external variables.

How can these penalties be determined? The costing of early penalties is relatively straight forward. Information such as: the internal rate of return and the standard cost used for warehousing space can be used to calculate the total dollar cost per day of producing a job early. The tardiness penalty is not easily converted into dollar terms. It is therefore easier to rank each job, in terms of its priority, and assign values to each to determine the exact level of importance on an arbitrary scale taking into account the elements mentioned above. Once this is completed, the organisation can then determine the ratio between the early penalties, based on dollars, and the tardiness penalty, based on our arbitrary scale. This therefore becomes the one critical, and highly subjective, decision that needs to be made. Once this ratio is determined, the penalties of one of the costs can be converted into the same units as the other, hence fully defining the early and tardy ratios for each job.

8.3. Effect of Error in the Early/Tardy Ratio

In Section 8.2. a method for determining the earliness penalty α , and the tardiness penalty β , was given. The key to this method, however, is in determining the ratio between the early penalty and the tardy penalty or the α/β ratio.

In this Section, we examine the effect an error has on this ratio and determine just how much error can be sustained before the solutions given become unreliable. To determine this we randomly generate four nine job problems for the problem classes where the earliness penalty is less than the tardiness penalty, in the manner described in Section 7.2. Each of the problems was then solved optimally by using a complete enumeration on the solution space. Although this took far more computational time than mixed integer programming, it did however enable us to solve problems with nine jobs, rather than the eight jobs which was the maximum previously for class J problems. The solution space of each of the problems was then enumerated another 38 times with each job's early/tardy factor multiplied by an error factor which ranged from -0.95 to 0.95 with 0.05 intervals. This means that the tests accommodate deviations in the α/β ratio by up to 95% above and below their true value. The deviation from optimal for each problem class was then calculated. The results by problem class are presented in Figure 8-1.

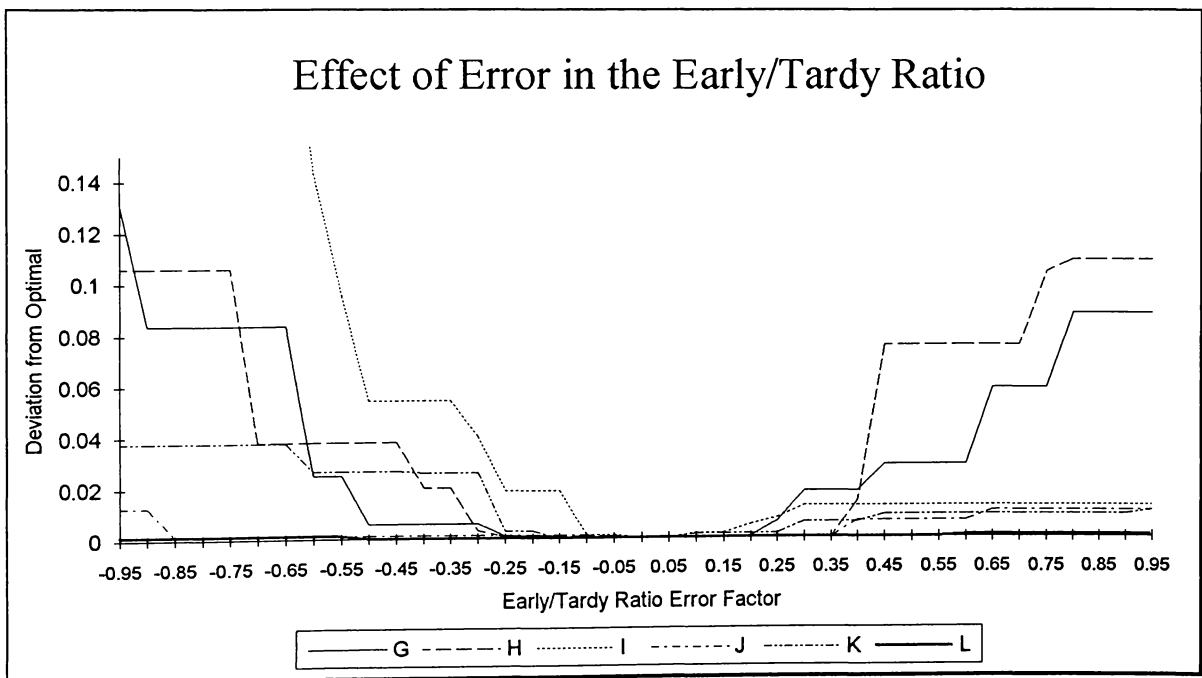


Figure 8-1.

Figure 8-1 shows several interesting points regarding the different problem classes and the effect of error on them.

All of the problem classes with a low tardiness factor, classes G, H and I, were more susceptible to over-estimation of the earliness penalty and/or under-estimation of the

8. Determining the Early/Tardy Ratio

tardiness penalty than those with a higher tardiness factor. Classes I, H and K are also very susceptible to the under-estimation of the earliness penalty and/or the over estimation of the tardiness penalty. In contrast to this class L, which has a tardiness factor of 0.6 and a due date range of 1.15, is not affected at all by errors of up to 50% either way. Class J, which has a tardiness factor of 0.6 and a due date range of 0.2, is also fairly insensitive to the error in the early/tardy ratio.

From these results we conclude that, with the exception of under-estimating the α/β ratio of class K, that problems with a high tardiness factor tend to be less susceptible to errors in the early/tardy ratio than those with a low tardiness factor. An aggregate of this data is shown in Figure 8-2.

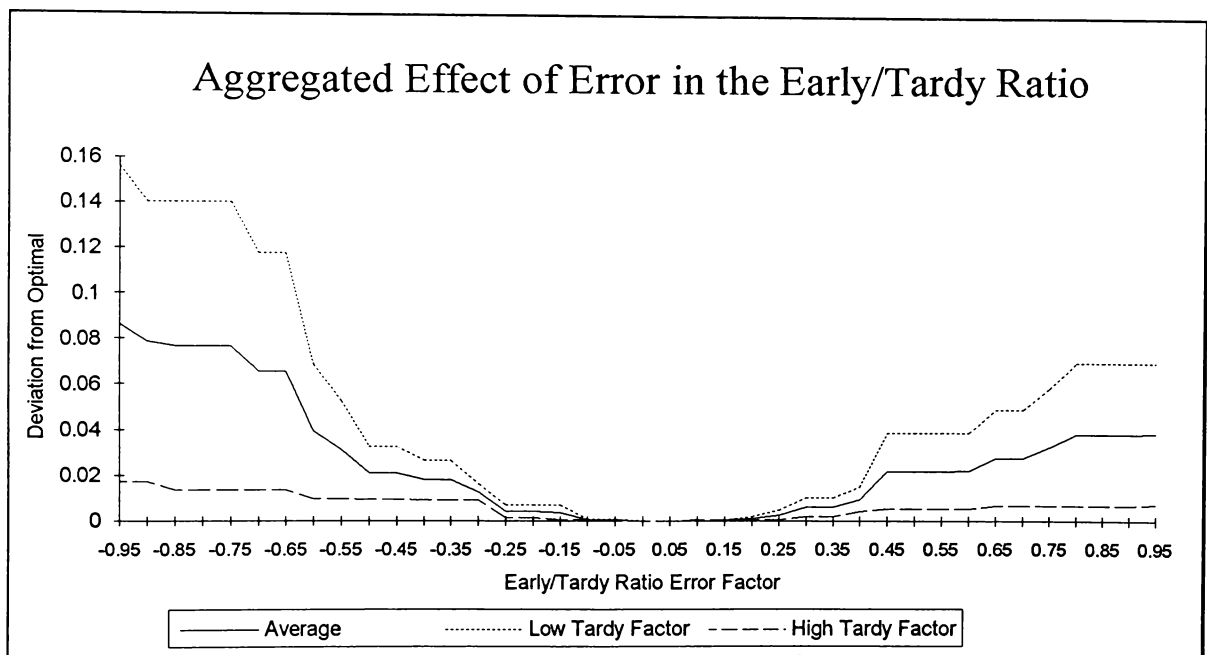


Figure 8-2.

Regression analysis was undertaken to replicate the aggregate graphs of high and low tardy factor problems by using both a quartic and a quadratic function. The resulting quartic equation for the low tardy factor jobs was $-0.010695x + 0.1979489x^2 - 0.045252x^3 - 0.074376x^4$, where x is the amount of error. The value of r^2 for this equation was 0.957 which is statistically significant. The quadratic function is $-0.036466x + 0.1475677x^2$, which has an r^2 of 0.921 which is again statistically significant. The graph of these equations is shown in Figure 8-3.

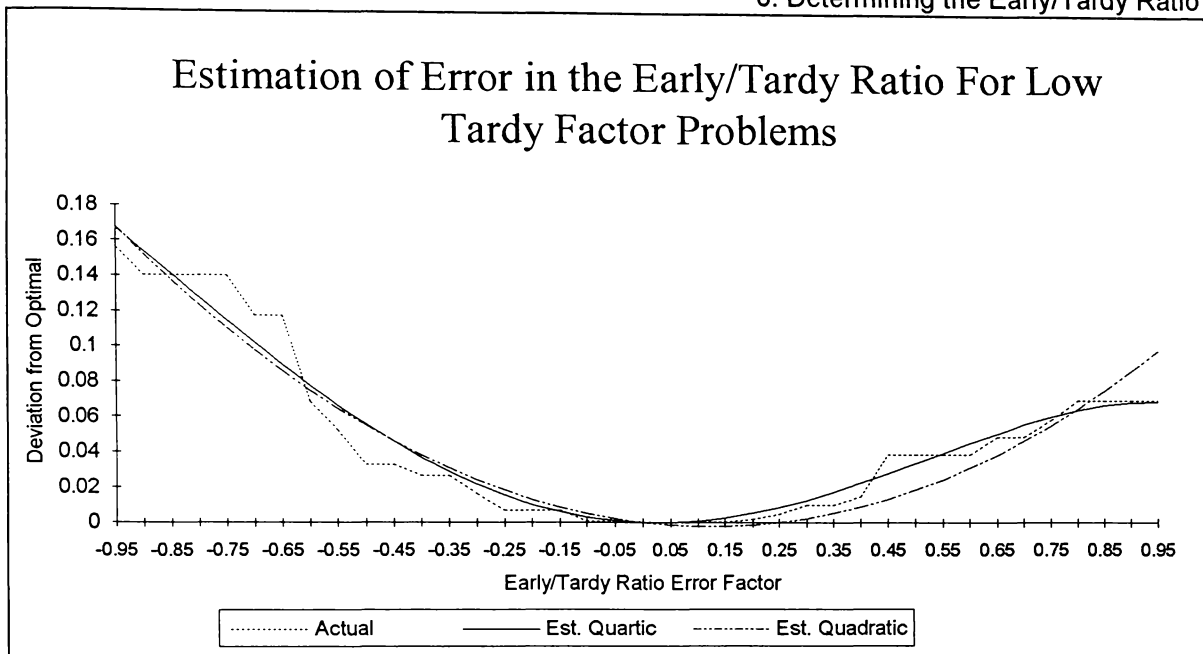


Figure 8-3

From Figure 8-3 we can see that the quartic equation approximates the data far better than the quadratic equation. The quadratic equation appears to underestimate the error with positive error factors less than 0.8. With negative error factors, both equations give approximately the same results.

The resulting quartic equation for high tardiness factor jobs is $-0.004941x + 0.0324146x^2 + 0.00001618x^3 - 0.022949x^4$. For this regression line the r^2 was 0.915, which is statistically significant. The quadratic function is $-0.004932x + 0.0168696x^2$, which has an r^2 of 0.762, which is again statistically significant. These two equations are graphed in Figure 8-4.

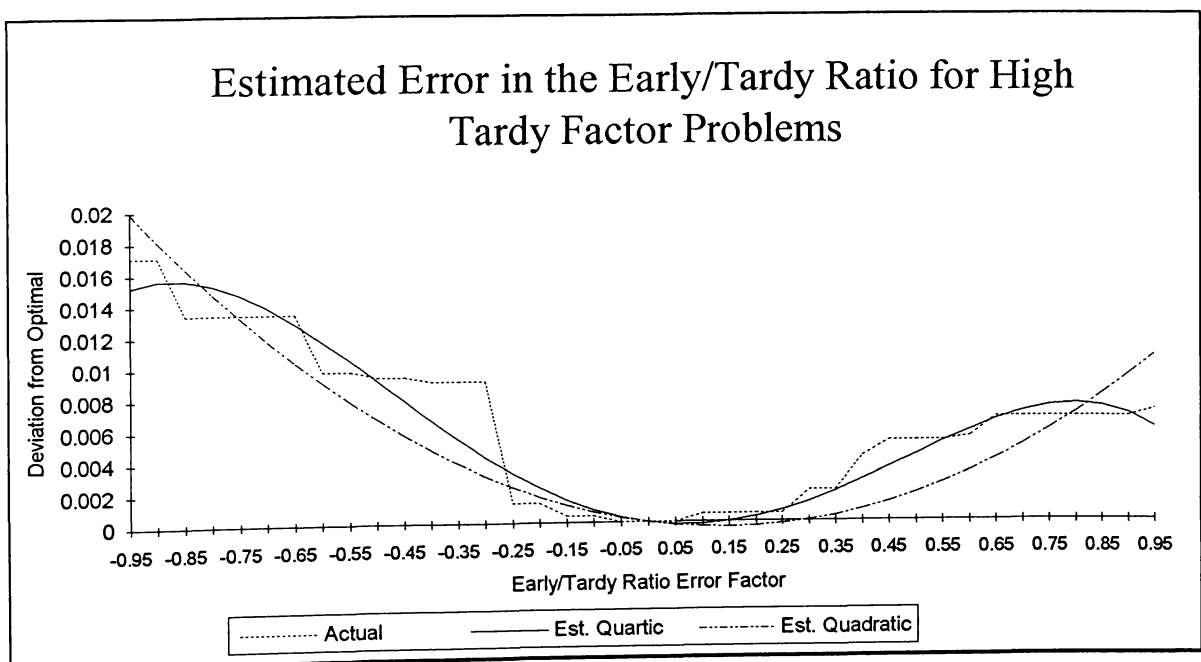


Figure 8-4

From the graph of Figure 8-4, it would appear that the quartic equation would not accurately extrapolate the graph, as it clearly has turning points at each end of the graph which are not consistent with our expectations of the larger the error, the greater the deviation from the optimal. Between these two turning point, however, the graph is clearly more accurate than the quadratic function, which for both positive and negative error factors tends to underestimate the error. A compromise function may be to use the quartic equation for error factors between its end turning points and use the quadratic equation for error factors beyond these two points.

An ANOVA analysis was carried out to determine:

1. whether the graphs of each problem class were significantly different from each other, and
2. whether the graphs of each problem class were significantly different from optimal.

The results from this analysis are presented in Table 8-1, where the Tukey results are presented in the same format as those in Appendix III.

Error Factor	Significance Between Problem Classes (p-value)	Between Problem Classes Tukey ANOVA at 5%	Significance from Optimal (p-value)
-0.95	0.009	[645[21]3]	0.001
-0.90	0.014	[645[12]3]	0.002
-0.85	0.011	[645[12]3]	0.003
-0.80	0.011	[645[12]3]	0.003
-0.75	0.002	[645[12]3]	0.003
-0.70	0.002	[6452[1]3]	0.005
-0.65	0.002	[6452[1]3]	0.005
-0.60	0.109	[64152]3	0.005
-0.55	0.089	641523	0.008
-0.50	0.089	641523	0.005
-0.45	0.089	641523	0.005
-0.40	0.102	641253	0.011
-0.35	0.102	641253	0.011
-0.30	0.238	642153	0.039
-0.25	0.420	621453	0.179
-0.20	0.420	621453	0.179
-0.15	0.389	562143	0.231
-0.10	0.744	561234	0.068
-0.05	0.615	456123	0.125
0.00	-	-	-
0.05	-	-	-
0.10	0.563	124653	0.163
0.15	0.563	124653	0.163
0.20	0.151	246153	0.085
0.25	0.179	246513	0.038
0.30	0.150	246531	0.024
0.35	0.150	246531	0.024
0.40	0.694	654321	0.009
0.45	0.332	645312	0.049
0.50	0.332	645312	0.049
0.55	0.332	645312	0.049
0.60	0.336	645312	0.048
0.65	0.203	654312	0.016
0.70	0.203	654312	0.016
0.75	0.026	654312	0.008
0.80	0.011	[6[5431]2]	0.005
0.85	0.011	[6[5431]2]	0.005
0.90	0.011	[6[5431]2]	0.005
0.95	0.011	[6[4531]2]	0.005

Table 8-1. ANOVA using Different Error Factors

Table 8-1 shows that there appears to be less statistical difference between the various problem classes for positive error factors, however there is slightly more significance for negative error factors. The deviation from optimal is more significant earlier for negative error factors than positive error factors. It also shows that an acceptable margin of error for the early/tardy ratio is no less than -25% and no greater than 20%, as these are within a 5% level of significance. Using this analysis it appears that it is better to over-estimate the earliness penalty and/or under estimate the lateness penalty rather than vice versa. This,

however, appears to be true only up to a point, as beyond this “critical point” the deviations are significant.

8.4. Conclusions

This chapter has introduced the issue of early/tardy cost determination and proposed some guidelines for calculating them. It is proposed that the early cost is primarily comprised of product based costs, such as product cost and size. The tardiness cost, on the other hand, is primarily comprised of market based costs, such as the amount of competition in the market and the urgency of the product for the customer.

For the generalised distinct due date E/T scheduling problem, the values of the early and tardy penalties are not important, however the value of the ratio between these two values is. As long as the ratio between the early penalty and the tardy penalty is correct then an accurate schedule will be produced. Tests determined how much error could be sustained by the early/tardy ratio before producing schedules with a significantly different cost. These tests showed that, as long as the error was between -25% and 20%, there was no significant difference in the cost of the resulting schedule.

The amount of error that can be sustained when solving larger problems has not been investigated, and is an area for further research. It would be expected that the amount of error may increase with larger problems, however the rate of increase is an unknown quantity.

In conclusion, the determination of the early/tardy penalties is an issue that needs to be dealt with when using E/T scheduling models. This is not a trivial exercise for any organisation as many subjective, and often political, issues must be raised and discussed - especially when tardiness factors and the relative importance of different customers are considered. Each of these penalties has different associated costs, however there is a reasonable margin of error in which to work before the errors become significant. Further research is needed to determine the impact that some of the factors, highlighted in this chapter, actually have on the real earliness penalty and tardiness penalty which occurs in industry.

9. Summary and Conclusions

9.1. Summary

The adoption of Just-In-Time (JIT) manufacturing in industry has increased markedly due to the large cost savings that can be obtained and the need to maintain a competitive advantage. The Early/Tardy (E/T) scheduling model reflects the objectives of JIT as it discourages producing jobs both before and after they are required by customers. Research to date, however, has concentrated on simplified models of this problem. These simplified models have assumed that either: all jobs are due on the same date, or there are restrictions on the earliness and tardiness penalties of each job. Baker and Scudder (1990) note that the generalised distinct due date Early/Tardy (E/T) scheduling problem had not been well researched and that the use of “sophisticated heuristic methods” had not yet been used as a basis for solving this problem. Of the few papers that have focused on the generalised problem none used so-called “intelligent” search techniques such as Simulated Annealing (SA), Genetic Algorithms (GAs) or Tabu Search (TS). These search techniques have, however, been used to solve a variety of scheduling problems including the common due date E/T scheduling problem.

The importance of the neighbourhood scheme/chromosomal representation and the impact this has on the quality of solutions obtained is well documented in the literature. Neighbourhood schemes which are customised to a particular type of problem were found to be more successful than general neighbourhood schemes.

In this thesis, search techniques were used to solve the distinct due date E/T scheduling problem by employing the two phase approach proposed by Baker and Scudder (1990). This approach uses a sequencing phase to order the jobs and a scheduling phase to insert idle time into the sequence. Swap and insert neighbourhood schemes were used to generate new sequences for the sequencing phase. A new idle time insertion algorithm was developed along with several theorems about optimal schedules. Computational experiments, however, showed that the new algorithm was not as efficient as those previously published. The algorithm developed by Fry, Armstrong *et al.* (1987) was therefore used.

The use of different solution spaces, based on the use of a heuristic and the modification of either the problem data or the operation of the heuristic, were discussed as an alternative means of solving scheduling problems.

The properties of the common due date problem enable it to define a schedule by specifying whether each job is to be produced early or late, called the E/T job specification. Schedules which can be defined by an E/T job specification can employ a search that uses E/T job

specification solution space, of size 2^n , instead of the sequence of jobs solution space, of size $n!$. This reduction in solution space size can increase the efficiency of the search.

Computational experiments were carried out to measure the efficiency of using E/T job specification solution space over the sequence-of-jobs solution space using the restricted Common Due Date (CDD) problem. These experiments showed that the E/T neighbourhood scheme initially made large gains in solution quality, with little subsequent progress. This was traced to problems with a high tardiness factor, that is those problems where the probability of the job being scheduled late is over 50%. A neighbourhood scheme, called the Date Swap TS, overcame this problem successfully by rescheduling the early jobs closest to the due date to being late when there was insufficient room to schedule a late job early. This stopped large areas of the solution space from becoming infeasible and therefore enabled the search to find the highest quality solutions in the least amount of time. Even in its basic form however, the E/T solution space did provide high quality solutions in a relatively small amount of time.

The properties of the restricted common due date problem were examined and generalised, with the aid of some already published results, to be applicable to the distinct due date problem. This allowed the development of two different heuristics which enabled a schedule to be defined from E/T job specifications. The first heuristic scheduled all the early jobs then inserted each late job in turn into the schedule, from the most important job to the least important. The second heuristic was based on the concept of conflict resolution. This means that all the early jobs are scheduled without any regard for the late jobs and the late jobs are scheduled without any regard for the early jobs. The resulting schedule is therefore infeasible due to conflicts between the early and late jobs. These conflicts are progressively resolved to make the schedule feasible. Experiments found that the insertion-based heuristic provided higher quality solutions in a shorter amount of time and was therefore adopted for the remaining experiments.

Tests on small distinct due date problems revealed that previously published heuristics performed very poorly, while search techniques performed well. In these tests the search techniques based on the standard solution space found higher quality solutions than those search techniques based on the E/T solution space. One reason for this was that the error in the E/T heuristic was greater than the error in the results found by these standard search techniques, making it impossible for the E/T heuristics to perform this well. When the search techniques were compared, the Revised TS was identified as the best search technique.

Experiments with large sized problems found that the searches based on the E/T solution space quickly found a local minima and made little progress from this point, in a similar

manner to the CDD experiments. The standard techniques, on the other hand, started at points which were far worst, but made far better progress overall when given the time. E/T SA was the best SA-based search technique, while the Standard GA with some heuristically generated population members was the best GA-based technique. With TS-based techniques there was little difference between the Revised Window TS starting at the E/T approximation to the EDD schedule, the Hybrid TS and the Revised Window TS with the E/T Extension. When a comparison was made between the search techniques, it was again found that the TS methods dominated all the other search techniques.

Experiments on the effect of starting points on the final solution quality showed that the starting point did have a major effect for most types of searches. Searches which used a random starting point performed significantly worse than those that used a starting point generated by a heuristic. This was evident for both the sequence-of-jobs solution space searches and the E/T job specification solution space searches. The Revised Window E/T Extension TS, however, was relatively insensitive to the starting point used. This indicates that the use of a search which incorporates a diversification strategy, such as this, is able to eliminate the sensitivity of the search to its starting point. The use of highly optimised starting points, as in the EDD schedule optimised by the Ow and Morton (1989) adjacency conditions, was found to be less effective than using the E/T approximation to the EDD schedule. This is due to the search starting in a deeper local minima, hence it takes longer for the search to climb out of this minima and move into other areas of the search space.

Comparisons between the search techniques and previously published heuristics on the large problems also show that large differences in solution quality exist between these two forms of heuristics. The heuristics published in the literature obtained a mean deviation from the known minimum of up to 150% more than the search techniques.

The effect of error in the earliness and tardiness penalties on the quality of the final solution was found to be dependent on the class of problem being solved. Problems with a low tardiness factor were more affected by error than those with a high tardiness factor. If the search technique is accurate enough, then as long as the error of the earliness penalty : tardiness penalty is between -25% and 20%, there is unlikely to be a significant difference in the solutions obtained.

9.2. Conclusions

What conclusions can be drawn from our experiment to the research questions proposed in Chapter 1?

1. Can a search technique be developed which can exploit the nature of the E/T scheduling problem?

Clearly, with the development of theory for the distinct due date E/T scheduling problem and the subsequent development of E/T scheduling heuristics which are used to create the E/T solution space, a search technique can be developed which exploits the nature of the E/T scheduling problem. However, it is also apparent that hybrid search techniques are superior to search techniques which are based on a single type of solution space. The E/T heuristics provide a good method of finding areas in the solution space in which to search, while the standard solution space provides a better method of fine tuning the schedule.

2. Which search technique is the most effective at solving the E/T Scheduling problem?

From our experiments in Sections 7.6.6 and 7.7.3, it was demonstrated that Tabu Search-based techniques were superior to those based on Simulated Annealing or Genetic Algorithms. For small problems, Revised Tabu Search provided the best results, while for large problems the Revised Window TS with an E/T approximation to the EDD schedule starting point, Revised Window TS with the E/T Extension and the Hybrid TS all provided similar results.

3. What is the effect of having errors in the earliness penalty and the tardiness penalty on the quality of the final solution?

From the experiment in Section 8.3, it can be concluded that the sensitivity of the solution to this error will depend, to a large extent, on the type of problem being solved. From our analysis it was found that problems with a low tardiness factor were more likely to be susceptible to error in the penalties than problems which had a high tardiness factor as this interrelationship between the earliness penalty and the tardiness penalty is greater for low tardiness factor problems. Generally if the early penalty : tardy penalty ratio has an error between -25% and 20% then there should be no significant difference between the results and the optimal solution, however it must be remembered that the use of search techniques may increase the effect of this error.

Other important conclusions from this research are:

1. The date swap neighbourhood scheme is an effective form of neighbourhood scheme for solving the CDD problem. It was the only search that quickly found solutions to all classes of problems and it had the lowest deviation from the known minimum of all the searches.

2. Infeasible areas within a solution space can make a large impact on the success or failure of a search. This was illustrated in solving the CDD problems with the standard E/T neighbourhood scheme. After some large initial gains the search ‘froze’ because it was unable to easily find feasible solutions to the problems which had a high tardiness factor.
3. The results obtained from search techniques are superior to the results obtained from previous standard heuristics. This was illustrated with both small and large problems with the results from the standard heuristics being of a very low quality.
4. The importance of idle time is illustrated graphically through the comparison of the standard Keyser and Sarper heuristic and the Keyser and Sarper heuristic with idle time optimally inserted. For small problems, the former had an average deviation from optimal of 105% while the latter had a deviation from optimal of just 5.09%. A similar result was obtained for large problems. However it became apparent through the course of this research that the importance of idle time was very problem dependent. A low tardiness factor was found to make problems require idle time at the start of the schedule. Schedules with a low tardiness factor and a high due date range also required idle time within the schedule. Schedules with a high tardiness factor seldom required idle time to be inserted. This leads us to conclude that the best method to solve a particular problem may best be selected depending on the type of problem being solved.
5. The way a search is implemented can affect its sensitivity to the starting points. In general, searches are extremely susceptible to the starting point used when the time for the search is limited. However this impact can be minimised by using a search which incorporates a diversification strategy, such as that used in the Revised TS with E/T Extension, as this allows the search to move quickly to better areas of the solution space.
6. In solving larger problems the hybrid approaches, which used both the E/T heuristic and the sequence-of-jobs solution space, consistently provided better results than any one technique used by itself.

9.3. Contributions

The following are contributions that this research has made to the areas of E/T scheduling and search technique implementation:

1. The explicit generalisation of the four common due date properties so they apply to the distinct due date problem. In their original form, Baker and Scudder (1990) state that two of the properties do not hold for the distinct due date problem, while the other two properties are not mentioned at all. The adjacency conditions developed by Ow and Morton (1989) for the distinct due date problem provide some necessary conditions for a schedule which can be applied when converting an E/T job specification to a physical schedule. However, the allocation of jobs to groups and the precise ordering of jobs within the central part of the group still requires a heuristic.

2. The development of two heuristics which can find a solution to the generalised distinct due date E/T scheduling problem from an E/T job specification. This builds on the theory developed above, by addressing the allocation and scheduling issues with the heuristics, each of which is based on a different paradigm. The first technique uses a construction heuristic paradigm, starting with a feasible schedule of early jobs, it inserts each late job in turn into the schedule ensuring that, at each stage, the schedule is ordered in the best possible way so as to minimise the objective function. There is no looking forward to see what jobs are coming up however the most important jobs are always scheduled first. The second technique uses a conflict resolution paradigm, whereby all the jobs are scheduled concurrently starting with an infeasible schedule, with no cost, and working toward a feasible schedule. The first feasible schedule found is the solution. The first contribution made by both of these heuristics is the fact that they have been able to schedule small E/T scheduling problems with a deviation from optimal of less than 0.8%. The second contribution is in the use of these heuristics to map the solution space from an abstract E/T job specification solution space to a physical solution space, not arbitrarily as in the case of Storer *et al.* (1993), but in a way that utilises the properties of the problem being solved. The third contribution of these heuristics is that they are the first heuristics which do not use a two phase approach of sequencing the jobs then inserting idle time, as both of these phases are done simultaneously in these heuristics.
3. The use of the three main search techniques: SA, GA and TS; to solve the generalised distinct due date E/T scheduling does not appear to have been investigated in the open literature. The stumbling block for search techniques in this area has been the need to use idle time insertion algorithms and the time penalty for evaluating neighbours that is subsequently incurred. Through the use of modifications which speed up the search, such as the Revised TS, this time penalty has been reduced, making the implementation of these searches feasible
4. The importance of starting points and their relationship to the solution space and neighbourhood scheme was investigated. Although starting points are a well researched area for search techniques, little work has been done on testing the effect of starting points with different solution spaces using the same search technique. Results showed that the Revised E/T Extension TS was insensitive to the starting point. This illustrates how diversification strategies can be used to minimise the effect of the starting point on the final solution, making these techniques more reliable.
5. The investigation of search techniques for the common due date search scheme led to the development of a new type of neighbourhood, called the E/T CDD Date Swap, which found high quality solutions very quickly when compared to the standard E/T

neighbourhood scheme. This search technique was superior to the standard search technique based on E/T job specifications in that it was able to solve problems with a high tardiness factor more efficiently by eliminating much of the invalid solution space present for the standard technique. There does not appear to be any research which has addressed the issue of excessive numbers of early jobs when solving problems with 'tight' due dates and its effect on a search technique. The E/T CDD Swap provides a simple yet effective means of improving search performance in situations such as this.

6. The effect of errors in the earliness and tardiness penalties does not appear to have been researched in the open literature. This research has shown that the effect of error in the penalties varies according to the type of problem being solved. Statistical analysis did, however, provide a range of results which were not statistically different from optimal and hence providing some broad guidelines for the accuracy requirements of these penalties.

9.4. Further Research

There are many areas in distinct due date E/T scheduling and search techniques which require further research and are beyond the scope of this thesis.

One such area of research is in removing the remaining assumptions from the generalised, single machine, distinct due date E/T scheduling model. Primarily this would involve including ready-time constraints and sequence dependent setup times and costs. The inclusion of ready times would be relatively straight forward, as currently all the heuristics have an implied ready time of 0. This needs to be generalised for each job. This may slow down the heuristics, because of the extra checking, but ultimately the underlying principles will remain unaltered. The incorporation of sequence dependent setup times, however, would have a major effect, especially on the E/T heuristics, as there is no provision at all for these costs in the current theory of E/T scheduling. The easiest method of implementing this modification is to use the Revised Window TS with the E/T start - initially ignoring setup costs, but incorporating them later into the Revised Window search. The E/T heuristic would therefore be used primarily for finding a starting solution for the search.

A logical progression from the single machine E/T scheduling problem is to look at the multi-machine problem. To date little, if anything, has been done in the area of multi-machine E/T scheduling. The reason for this is that it is highly intractable and will require new tools and techniques to be developed before it can be solved efficiently. Progress to date with single machine E/T scheduling may indicate, however, that the development of new techniques for the multi-machine problem, especially the generalised case, is a long way off.

The development of faster and more accurate E/T heuristics is another area requiring further research. One type of E/T heuristic that could be developed is based on using a search technique as part of the actual E/T heuristic phase. This search would be used to determine which group a job is to belong to and also the precise order of jobs within each group. This search heuristic would be required to run very quickly so that it could be executed many times during the search over the E/T solution space.

Better date swap-based heuristics are also required for distinct due date problems. The results of the current heuristic showed that it was too slow and, as a consequence, performed poorly. The CDD experiments, however, show that there are substantial gains to be made in solution quality by using these neighbourhood schemes.

The development of a decision support system which incorporates the E/T heuristic is another area worthy of investigation. This would allow production schedulers to do their own searching rather than using a search technique such as TS. Once a schedule is determined on the E/T level, the scheduler could then drop down to the physical level and modify the schedule from there - in a similar manner to the Hybrid TS.

One key area of further research is in finding methods to determine the early and tardy penalties for each job. In this research we have discussed some of the elements which make up these penalties and the effect that error has on the results. It is desirable that a more formal method of determining these values is developed and tested within industry.

The most challenging field of future research is in taking the techniques developed here to solve the E/T scheduling problem and applying them to other problems. Clearly, the use of a compressed solution space, such as the E/T job specification solution space, will not be possible for many problems. However, the key motivation which aided the design of the E/T heuristics was to try to use the major stumbling blocks of the problem and turn them into stepping stones for the problem's solution. Clearly some problems, like the standard *Travelling Salesman Problem*, do not have a structure that allows this to happen, however other problems may do - therein lies the challenge.

APPENDIX I

SEARCH TECHNIQUE EXAMPLES

The objective of this appendix is to provide some examples of how each of the different search techniques operate.

The problem used in these examples has the following information:

Job No.	Processing Time	Due Date	Earliness Penalty	Lateness Penalty
0	119	780	5	7
1	89	682	5	9
2	85	778	5	7
3	90	744	3	9
4	114	776	5	8
5	121	640	1	10
6	80	743	1	9
7	87	665	2	8
8	103	751	4	6

Table A1-1 Example Problem Data

Simulated Annealing

A SA search techniques based on the swap neighbourhood scheme was used in Table A1-2 to illustrate the way in which SA works. The parameters used for this search were an initial temperature of 0.2 of the standard initial temperature and a decrement rate parameter of 0.1.

Iteration #	Sequence	Cost	Temp	Random #	Prob. Accept	Swap
1	5-7-1-6-3-8-4-2-0	5788				
2	5-7-1-4-3-8-6-2-0	6839	4104	0.3086	0.7741	4 & 6
3	5-7-1-4-6-8-3-2-0	6516	4024	-	-	6 & 5
4	8-7-1-4-6-5-3-2-0	8339	3946	0.4948	0.6300	8 & 5
5	8-7-6-4-1-5-3-2-0	7617	3872	-	-	1 & 6
6	8-7-6-4-1-2-3-5-0	9686	3800	0.36807	0.5802	2 & 5
7	8-7-6-4-1-2-3-5-0	(10763)	3731	0.7657	0.7493	Rejected
8	8-4-6-7-1-2-3-5-0	9613	3664	-	-	7 & 4

Table A1-2. Simulated Annealing Search Iterations

Note that at each stage when the solution obtained has a cost higher than the current cost a probability is calculated from the temperature and the increase in cost to obtain a probability of accepting the solution. A random number is then used to determine whether the solution is to be accepted. If the random number is less than the acceptance probability, then the

move is accepted (as in iteration 2, 4 and 6), however if the random number is greater than the probability then the solution is rejected (as in iteration 7).

Genetic Algorithms

A GA implemented in the same way as the Standard GA outlined in Chapter 6 was implemented and the results from the first four generations are presented in Table A1-3. The parameters used were a population size of 5 and a mutation rate of 0.5.

Generation #		Sequence	Cost	Move
1	A	6-8-7-1-5-4-3-2-0	6863	
	B	4-5-1-7-2-0-8-3-6	9964	
	C	2-6-4-8-3-1-0-7-5	12221	
	D	6-4-5-1-7-2-0-8-3	7884	
	E	3-2-7-4-1-6-8-0-5	11043	
2	A	4-3-7-1-5-6-8-0-2	8219	PMX with A & E Sequences 4-3-2/6-8-0
	B	4-5-1-7-2-0-8-3-6	9964	
	C	6-8-7-1-5-4-3-2-0	6863	
	D	6-4-5-1-7-2-0-8-3	7884	
	E	8-0-7-6-1-4-3-2-5	10481	
3	A	6-4-5-1-7-2-0-8-3	7884	Mutation Insert job 5 at position 6 CX with A & C, eliminate result of 4-8-7-1-5-6-3-2-0 as it has the worst cost of the chromosomes involved.
	B	4-1-7-2-0-5-8-3-6	11713	
	C	6-3-7-1-5-4-8-0-2	6759	
	D	4-3-7-1-5-6-8-0-2	8219	
	E	6-8-7-1-5-4-3-2-0	6863	
4	A	6-4-5-1-7-2-0-8-3	7884	OX C & E, Sequence 8/3
	B	6-3-7-1-5-4-8-0-2	6759	
	C	4-3-7-1-5-6-8-0-2	8219	
	D	6-8-7-5-4-1-3-2-0	6222	
	E	3-2-0-6-7-1-8-5-4	10786	

Table A1-3. Genetic Algorithm Generations

At each generation two chromosomes are picked for a crossover. The type of crossover is randomly chosen and the resulting four chromosomes - the two parents and the two children - are examined and the best two placed back into the population. If the remaining two chromosomes have a lower cost than the worst two chromosomes in the population, then these replace the worst chromosomes as appropriate. After the crossover, a mutation operation is performed with the probability of 0.5. This will either swap two jobs or insert one job in another place within the schedule. The mutation operation therefore provides some diversity to the population that may be lost.

The best solution for the example above is obtained in generation number 4, with a cost of 6222.

Tabu Search

A TS with a swap neighbourhood scheme was used to illustrate how TS operates. The results of the search iterations are shown in Table A1-4. A tabu list of three was used in this example. The first job chosen out of each pair is made tabu so that the search does not cycle.

Iteration #	Sequence	Cost	Tabu List	Swap
1	5-7-1-6-3-8-4-2-0	5788		
2	5-7-8-6-3-1-4-2-0	5403	1	1 & 8
3	5-7-6-8-3-1-4-2-0	5186	1 6	6 & 8
4	5-6-7-8-3-1-4-2-0	5113	1 6 6	6 & 7 (Aspiration Criteria met)
5	5-6-7-3-8-1-4-2-0	5062	6 6 3	3 & 8
6	5-6-7-3-8-1-2-4-0	5196	6 3 2	2 & 4
7	5-6-7-3-1-8-2-4-0	5289	3 2 1	1 & 8
8	6-5-7-3-1-8-2-4-0	5330	2 1 6	6 & 5
9	6-5-7-8-1-3-2-4-0	5405	1 6 3	3 & 8
10	6-5-7-8-1-3-4-2-0	5337	6 3 4	4 & 2

Table A1-4. Tabu Search Iterations

The TS starts by examining all the neighbours to the current solution. For example in the first iteration it will consider all neighbouring schedules to the sequence 5-7-1-6-3-8-4-2-0 with two of the jobs swapped. The neighbouring sequences which are examined are:

7-5-1-6-3-8-4-2-0	1-7-5-6-3-8-4-2-0	6-7-1-5-3-8-4-2-0	3-7-1-6-5-8-4-2-0
8-7-1-6-3-5-4-2-0	4-7-1-6-3-8-5-2-0	2-7-1-6-3-8-4-5-0	0-7-1-6-3-8-4-2-5
5-1-7-6-3-8-4-2-0	5-6-1-7-3-8-4-2-0	5-3-1-6-7-8-4-2-0	5-8-1-6-3-7-4-2-0
5-4-1-6-3-8-7-2-0	5-2-1-6-3-8-4-7-0	5-0-1-6-3-8-4-2-7	5-7-6-1-3-8-4-2-0
5-7-3-6-1-8-4-2-0	5-7-8-6-3-1-4-2-0	5-7-4-6-3-8-1-2-0	5-7-2-6-3-8-4-1-0
5-7-0-6-3-8-4-2-1	5-7-1-3-6-8-4-2-0	5-7-1-8-3-6-4-2-0	5-7-1-4-3-8-6-2-0
5-7-1-2-3-8-4-6-0	5-7-1-0-3-8-4-2-6	5-7-1-6-8-3-4-2-0	5-7-1-6-4-8-3-2-0
5-7-1-6-2-8-4-3-0	5-7-1-6-0-8-4-2-3	5-7-1-6-3-4-8-2-0	5-7-1-6-3-2-4-8-0
5-7-1-6-3-0-4-2-8	5-7-1-6-3-8-2-4-0	5-7-1-6-3-8-0-2-4	5-7-1-6-3-8-4-0-2

Note that there are 36 sequences examined although there are actually 72 different neighbours to this sequence, as swapping x and y is the same as swapping y and x . Of these it was found that 5-7-8-6-3-1-4-2-0 had the lowest cost and hence it was selected and became the new current solution, and 1 was added to the tabu list to prevent a reversal of the swap of jobs 1 and 8 which formed the new sequence. At iteration 4 we can see that the search, despite having the job 6 on the tabu list, still selected a move which involved the job 6 because this new move had a cost which was lower than the best cost found so far, that is the tabu status of the move was overridden by the aspiration criteria. By iteration 5 a minima had been reached and iteration 6 shows the search climbing out of this minima. The tabu list ensures that the search does not fall back into the minima once it has moved out from it.

APPENDIX II

E/T HEURISTIC EXAMPLES

The schedule graphs that follow illustrate the way in which the two different E/T heuristics work on a small 15 job, class I problem which has a tardiness factor of 0.2 and a due date range on 1.15. The E/T job specification for this problem is that jobs 0, 1, 2, 3, 4, 6, 7, 9, 10, 12, 13 and 14 are to be scheduled early while jobs 5, 8 and 11 are to scheduled late. This E/T specification produces the optimal schedule with a cost of 2217.

The Data for this problem is in the following table.

Job No.	Processing Time	Due Date	Earliness Penalty	Lateness Penalty	E/T Job Specification
0	36	1865	3	9	0
1	75	440	1	9	0
2	69	957	4	8	0
3	82	652	3	7	0
4	100	711	5	9	0
5	112	755	4	10	1
6	89	1242	3	7	0
7	117	351	5	9	0
8	129	410	3	9	1
9	92	1796	5	9	0
10	75	1091	2	6	0
11	93	759	5	6	1
12	88	545	2	7	0
13	133	2009	2	6	0
14	117	1575	4	10	0

Table A2-1. Example Problem Data

The following narration will explain each heuristic in turn iteration by iteration and are to read in conjunction with Figures A2-1 and A2-2.

Five Point Insertion

- Iteration 1: All the early jobs are scheduled as close as possible to their due date. All early jobs are scheduled on their due dates except for jobs 12 and 3.
- Iterations 2-3: The group of early jobs is scanned, that is job 7, and job most important late job which is applicable to this group, job 8, is inserted. As job 8 has a very high tardiness penalty, the two early jobs, 7 and 1, are moved back to allow job 8 to be completed on time. In moving these two jobs back the order of jobs 7 and 1 is swapped as job 1 is less important.
- Iteration 4: The next group of jobs is examined, and two jobs could be inserted at this stage, Job 5 and Job 11, however because job 5 has a lower beta ratio, it is inserted first. The insertion point with the least cost is at the end of the group, hence it is scheduled at this point. If we examine this group at this stage, the total earliness penalty from jobs 12, 3 and 4 is 10 and the total tardiness penalty from job 5 is 10 also. Hence there is no cost benefit (or extra cost) by scheduling the jobs earlier than what they are at the moment.
- Iterations 5-7: The final late job is inserted, that is job 11. It can not be added to the end of the group, as it then collides with job 2, however now, for this group, we have a total earliness penalty of 10 but a tardiness factor of 16, hence there is a cost advantage of moving the group back. The group is then moved back to absorb the idle time between jobs 8 and 12. The group can not be moved any further back as job 8 is a late job already scheduled on its due date, hence to move it back any earlier would mean that job 12 would be scheduled before job 8.

Dynamic Loop

- Iteration 1: Early jobs are scheduled optimally, as with the five point insertion heuristic, as well as the late jobs. There are several areas of conflict to be resolved. Jobs 8 and 7, Jobs 8 and 1 and Jobs 5 and 4.
- Iteration 2: Conflicts start being resolved. The first conflict between Jobs 7 and 8 is addressed first. Job 8 is moved forward to the due date of Job 1, as from this date job 1 can be considered in the conflict resolution as well.
- Iteration 3: The earliness penalty of jobs 7 and 1 is less than the lateness penalty of job 8, hence job 1 is moved back to absorb the idle time between it and job 7.
- Iteration 4: In moving job 1 back, it passes a critical point which swaps the ordering of jobs 7 and 1.
- Iterations 5-6: Jobs 1 and 7 are then moved back even more to resolve the conflict with job 8.
- Iteration 7: The conflict between jobs 5 and 4 is addressed. First the late jobs 5 and 11 are move out as fas as the starting time of job 2.
- Iterations 8-9: The tardiness penalty is found to be greater than the earliness penalty, therefore the group of early jobs is pushed back and the late jobs are rescheduled to the due date of job 3, since job 4 is the conflicting job the removal of job 4 would make this date possible. Also job 8 is rescheduled earlier along with jobs 1 and 7.
- Iterations 10-11: The early jobs 12, 3 and 4 are then pushed back further to absorb the idle time between jobs 8 and 12.
- Iteration 12: No further pushing back of the early jobs is cost beneficial, hence the jobs 5 and 11 are scheduled later at the end of the group and the conflict is resolved.

Figure A2-1 Problem IA With 5 Point Insertion

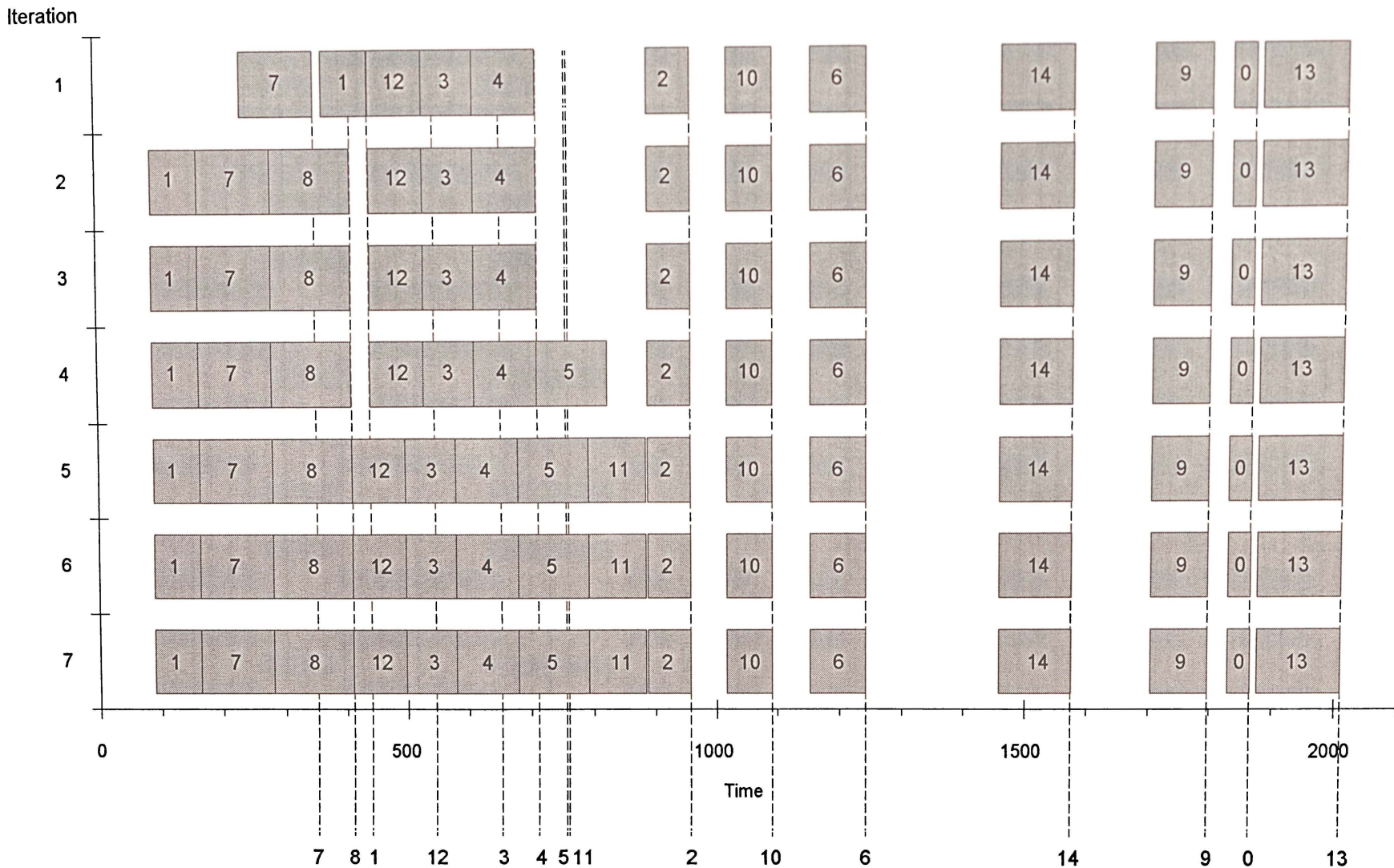
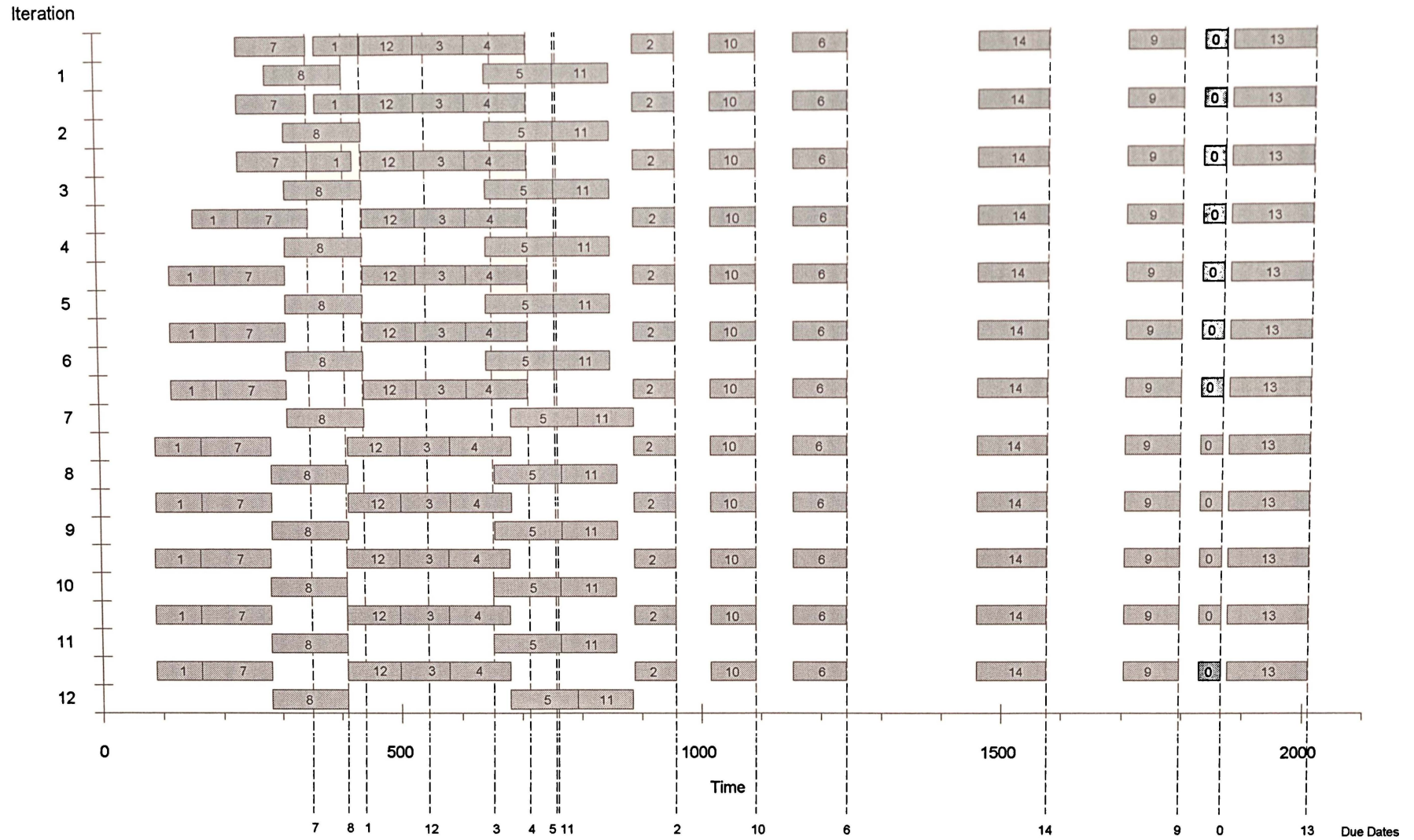


Figure A2-2 Problem IA With Dynamic Loop



APPENDIX III

COMMON DUE DATE ANOVA RESULTS

The following tables show the p-values when different searches are compared at different times.

Table A3-1. Common Due Date ANOVA Significance Levels of Search Method

Time	All Methods	All Excl. RevTS	CDD Date vs RevWinTS	RevCDD vs RevWinTS	Rev CDD vs CDD Date	CDD Date vs CDD Swap
0.0	0.000	0.000	0.000	0.934	0.000	1.000
312.5	0.000	0.000	0.000	0.000	0.160	0.484
625.0	0.000	0.000	0.001	0.000	0.533	0.642
937.5	0.000	0.000	0.014	0.000	0.683	0.720
1250.0	0.000	0.000	0.059	0.000	0.805	0.772
1562.5	0.000	0.000	0.154	0.000	0.859	0.792
1875.0	0.000	0.004	0.293	0.000	0.902	0.816
2187.5	0.000	0.013	0.461	0.000	0.927	0.830
2500.0	0.000	0.023	0.629	0.000	0.951	0.848
2812.5	0.000	0.034	0.788	0.000	0.965	0.861
3125.0	0.000	0.041	0.932	0.000	0.983	0.875
3437.5	0.000	0.053	0.938	0.000	0.991	0.881
3750.0	0.000	0.057	0.838	0.000	0.979	0.896
4062.5	0.000	0.053	0.753	0.000	0.971	0.904
4375.0	0.000	0.052	0.682	0.000	0.959	0.910
4687.5	0.000	0.047	0.623	0.000	0.948	0.918
5000.0	0.000	0.050	0.575	0.000	0.939	0.921
5312.5	0.000	0.046	0.533	0.000	0.932	0.925
5625.0	0.000	0.042	0.502	0.000	0.926	0.928
5937.5	0.000	0.038	0.350	0.000	0.915	0.931
6250.0	0.000	0.035	0.452	0.000	0.910	0.936
6562.5	0.000	0.036	0.434	0.000	0.904	0.936
6875.0	0.000	0.036	0.418	0.000	0.897	0.937
7187.5	0.000	0.035	0.405	0.000	0.893	0.938
7500.0	0.000	0.035	0.405	0.000	0.893	0.938

Table A3-2. Common Due Date ANOVA Significance Levels of Problem Class

Time	RevTS	RevWinTS ETS	CDD TS	Rev CDD	CDD Swap	CDD Date Swap
0.0	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
312.5	0.0000	0.0000	0.0000	0.0000	0.0000	0.3413
625.0	0.0000	0.0000	0.0000	0.0000	0.0000	0.3279
937.5	0.0000	0.0000	0.0000	0.0000	0.0000	0.3256
1250.0	0.0000	0.0000	0.0000	0.0000	0.0000	0.3251
1562.5	0.0000	0.0000	0.0000	0.0000	0.0000	0.3246
1875.0	0.0000	0.0000	0.0000	0.0000	0.0000	0.3244
2187.5	0.0000	0.0000	0.0000	0.0000	0.0000	0.3243
2500.0	0.0000	0.0000	0.0000	0.0000	0.0000	0.3242
2812.5	0.0000	0.0000	0.0000	0.0000	0.0000	0.3239
3125.0	0.0000	0.0000	0.0000	0.0000	0.0000	0.3239
3437.5	0.0000	0.0000	0.0000	0.0000	0.0000	0.3239
3750.0	0.0000	0.0000	0.0000	0.0000	0.0000	0.3239
4062.5	0.0000	0.0000	0.0000	0.0000	0.0000	0.3237
4375.0	0.0000	0.0000	0.0000	0.0000	0.0000	0.3237
4687.5	0.0000	0.0000	0.0000	0.0000	0.0000	0.3236
5000.0	0.0000	0.0000	0.0000	0.0000	0.0000	0.3236
5312.5	0.0000	0.0000	0.0000	0.0000	0.0000	0.3236
5625.0	0.0000	0.0000	0.0000	0.0000	0.0000	0.3236
5937.5	0.0000	0.0000	0.0000	0.0000	0.0000	0.3236
6250.0	0.0000	0.0000	0.0000	0.0000	0.0000	0.3236
6562.5	0.0000	0.0000	0.0000	0.0000	0.0000	0.3236
6875.0	0.0000	0.0000	0.0000	0.0000	0.0000	0.3236
7187.5	0.0000	0.0000	0.0000	0.0000	0.0000	0.3235
7500.0	0.0000	0.0000	0.0000	0.0000	0.0000	0.3235

APPENDIX IV

DISTINCT DUE DATE ANOVA RESULTS FOR LARGE PROBLEMS

Tukey One-Way ANOVAs are shown at the 5% level of significance. The notation to represent groups is for each group to be enclosed in square brackets. For example [12[3]45]6 means that the methods which are not significantly different from each other are 123, 345 and 6. Each group is numbered with Class G=1 and Class L=6.

Table A4-1. ANOVA of Simulated Annealing Techniques

Time	ANOVA			Tukey One-Way	
	One-Way Method	One-Way Problem	Two-Way Interaction	Method	Problem
0.0	0.065	0.000	0.147	[4[23]1]	[45163]2
312.5	0.000	0.000	0.000	[234]1	[45[61]3]2
625.0	0.011	0.000	0.000	[23[4]1]	[4[56][1]3]2
937.5	0.055	0.000	0.000	2341	[546][13]2
1250.0	0.127	0.000	0.000	2341	[654][13]2
1562.5	0.209	0.000	0.000	2341	[564][13]2
1875.0	0.295	0.000	0.000	2341	[654][1][3][2]
2187.5	0.388	0.000	0.000	2341	[654][1][3][2]
2500.0	0.455	0.000	0.000	2341	[654][1][3][2]
2812.5	0.497	0.000	0.000	2341	[654][1][3][2]
3125.0	0.567	0.000	0.000	2341	[654][1][3][2]
3437.5	0.637	0.000	0.000	2341	[65[4]1][3][2]
3750.0	0.679	0.000	0.000	2341	[65[4]1][3][2]
4062.5	0.710	0.000	0.000	2341	[65[4]1][3][2]
4375.0	0.740	0.000	0.000	2341	[65[4]1][3][2]
4687.5	0.754	0.000	0.000	2341	[65[4]1][3][2]
5000.0	0.777	0.000	0.000	2341	[65[4]1][3][2]
5312.5	0.763	0.000	0.000	2341	[65[4]1][3][2]
5625.0	0.800	0.000	0.000	2341	[65[4]1][3][2]
5937.5	0.808	0.000	0.000	2341	[6[54]1][3][2]
6250.0	0.819	0.000	0.000	2341	[6[54]1][3][2]
6562.5	0.832	0.000	0.000	2341	[6[54]1][3][2]
6875.0	0.841	0.000	0.000	2341	[6[54]1][3][2]
7187.5	0.844	0.000	0.000	2341	[6[54]1][3][2]
7500.0	0.844	0.000	0.000	2341	[6541][3][2]

No.	Technique
1	Standard
2	E/T
3	Smart E/T
4	Standard with E/T EDD Approximation Starting point

Table A4-2. Genetic Algorithms ANOVA Analysis

Time	One-Way Method	ANOVA		Tukey One-Way	
		One-Way Problem	Interaction	Method	Problem
0.0	0.008	0.000	0.000	[4[3]1]2	[546[1]3]2
312.5	0.000	0.001	0.000	[431]2	[4156[2]3]
625.0	0.000	0.001	0.000	[431]2	[4156[2]3]
937.5	0.000	0.001	0.000	[431]2	[4156[2]3]
1250.0	0.000	0.001	0.000	[431]2	[4156[2]3]
1562.5	0.000	0.001	0.000	[431]2	[4156[2]3]
1875.0	0.000	0.001	0.000	[431]2	[4156[2]3]
2187.5	0.000	0.001	0.000	[431]2	[4156[2]3]
2500.0	0.000	0.001	0.000	[431]2	[4156[2]3]
2812.5	0.000	0.000	0.000	[431]2	[4156[2]3]
3125.0	0.000	0.000	0.000	[431]2	[4156[2]3]
3437.5	0.000	0.000	0.000	[431]2	[4156[2]3]
3750.0	0.000	0.000	0.000	[431]2	[4156[2]3]
4062.5	0.000	0.000	0.000	[431]2	[4165[2]3]
4375.0	0.000	0.000	0.000	[431]2	[4165[2]3]
4687.5	0.000	0.000	0.000	[431]2	[4165[2]3]
5000.0	0.000	0.000	0.000	[431]2	[4165[2]3]
5312.5	0.000	0.000	0.000	[431]2	[4165[2]3]
5625.0	0.000	0.000	0.000	[431]2	[1465[2]3]
5937.5	0.000	0.000	0.000	[431]2	[1465[2]3]
6250.0	0.000	0.000	0.000	[431]2	[1465[2]3]
6562.5	0.000	0.000	0.000	[431]2	[1465[2]3]
6875.0	0.000	0.000	0.000	[431]2	[1465[2]3]
7187.5	0.000	0.000	0.000	[431]2	[1645[2]3]
7500.0	0.000	0.000	0.000	[431]2	[1645[2]3]

No.	Technique
1	E/T
2	Standard
3	E/T with E/T EDD Included
4	Standard with EDD + E/T EDD Included

Table A4-3. Revised Tabu Search ANOVA

Time	One-Way Method	ANOVA		Tukey One-Way	
		One-Way Problem	Two-Way Interaction	Method	Problem
0.0	1.000	0.000	1.000	45123	[46][53][1][2]
312.5	0.488	0.000	0.979	54213	[6[4]5][3][1][2]
625.0	0.439	0.000	0.978	54213	[645][3][1][2]
937.5	0.507	0.000	0.991	54213	[645][3][1][2]
1250.0	0.494	0.000	0.938	54213	[654][3][1][2]
1562.5	0.525	0.000	0.891	54213	[654][3][1][2]
1875.0	0.530	0.000	0.531	54213	[654][3][1][2]
2187.5	0.546	0.000	0.188	54213	[654][3][1][2]
2500.0	0.473	0.000	0.010	54213	[654][3][1][2]
2812.5	0.435	0.000	0.000	54213	[654][3][21]
3125.0	0.441	0.000	0.000	54213	[654][3][21]
3437.5	0.472	0.000	0.000	54213	[654][3][21]
3750.0	0.419	0.000	0.000	54213	[654][3][2][1]
4062.5	0.418	0.000	0.000	54213	[65[4]3][2][1]
4375.0	0.363	0.000	0.000	54213	[65[4]3][2][1]
4687.5	0.367	0.000	0.000	54213	[65[4]3][2][1]
5000.0	0.335	0.000	0.000	54213	[6543][2][1]
5312.5	0.310	0.000	0.000	54213	[6543][2][1]
5625.0	0.273	0.000	0.000	54213	[6543][2][1]
5937.5	0.275	0.000	0.000	54213	[6543][2][1]
6250.0	0.258	0.000	0.000	54213	[6543][2][1]
6562.5	0.240	0.000	0.000	54213	[6543][2][1]
6875.0	0.214	0.000	0.000	54213	[6543][2][1]
7187.5	0.194	0.000	0.000	54213	[6543][2][1]
7500.0	0.194	0.000	0.000	54213	[6543][2][1]

No.	List Size	Window Size
1	15	0.04
2	15	0.02
3	15	0.06
4	15	0.012
5	15	0.008

Table A4-4. Revised Early Tardy Tabu Search ANOVA

Time	ANOVA			Tukey One-Way	
	One-Way Method	One-Way Problem	Two-Way Interaction	Method	Problem
0.0	1.000	0.000	1.000	56123478	[5416][3][2]
312.5	0.572	0.000	0.947	81276534	[5[461]3]2
625.0	0.878	0.000	0.971	56816734	[56[4]1][3]2
937.5	0.940	0.000	0.705	53648127	[65][41][3][2]
1250.0	0.918	0.000	0.566	56348127	[65][41][3][2]
1562.5	0.895	0.000	0.427	53468127	[65][41][3][2]
1875.0	0.874	0.000	0.307	53468127	[65][41][3][2]
2187.5	0.878	0.000	0.191	53468127	[65][41][3][2]
2500.0	0.860	0.000	0.182	53468127	[65][41][3][2]
2812.5	0.848	0.000	0.122	53468127	[65][41][3][2]
3125.0	0.840	0.000	0.091	53468127	[65][41][3][2]
3437.5	0.836	0.000	0.059	53468127	[65][41][3][2]
3750.0	0.822	0.000	0.053	53468127	[65][41][3][2]
4062.5	0.798	0.000	0.041	54638127	[65][41][3][2]
4375.0	0.801	0.000	0.029	54638127	[65][41][3][2]
4687.5	0.792	0.000	0.035	54638127	[65][41][3][2]
5000.0	0.775	0.000	0.039	54638127	[65][41][3][2]
5312.5	0.758	0.000	0.040	56438127	[65][41][3][2]
5625.0	0.753	0.000	0.045	56438127	[65][41][3][2]
5937.5	0.753	0.000	0.050	65438127	[65][14][3][2]
6250.0	0.724	0.000	0.041	56438127	[65][14][3][2]
6562.5	0.723	0.000	0.036	65438127	[65][14][3][2]
6875.0	0.722	0.000	0.036	65438127	[65][14][3][2]
7187.5	0.710	0.000	0.034	64538127	[65][14][3][2]
7500.0	0.710	0.000	0.034	64538127	[65][14][3][2]

No.	List Size	Window Size
1	1	0.04
2	7	0.04
3	1	0.12
4	1	0.14
5	1	0.08
6	1	0.10
7	1	0.02
8	1	0.06

Table A4-5. Revised Window Tabu Search, Early/Tardy Starting Point ANOVA

Time	One-Way Method	ANOVA		Tukey One-Way	
		One-Way Problem	Two-Way Interaction	Method	Problem
0.0	1.000	0.000	1.000	6534789A12	[541][6][3][2]
312.5	0.990	0.000	0.815	5839124A67	[546][1][3][2]
625.0	0.978	0.000	0.001	58123964A7	[564][1][3][2]
937.5	0.944	0.000	0.000	5813296A47	[564][1][3][2]
1250.0	0.931	0.000	0.000	5812369A47	[564][1][3][2]
1562.5	0.889	0.000	0.000	5813269A47	[654][1][3][2]
1875.0	0.744	0.000	0.000	5831296A47	[65[4]1][3][2]
2187.5	0.625	0.000	0.000	5138269A47	[65[4]1][3][2]
2500.0	0.540	0.000	0.000	5138269A47	[65[4]1][3][2]
2812.5	0.430	0.000	0.000	5132869A47	[6541][3][2]
3125.0	0.357	0.000	0.000	5132689A47	[6541][3][2]
3437.5	0.260	0.000	0.000	5132689A47	[6541][3][2]
3750.0	0.165	0.000	0.000	5132689A74	[6541][3][2]
4062.5	0.116	0.000	0.000	513628A974	[6541][3][2]
4375.0	0.094	0.000	0.000	513628A974	[6541][3][2]
4687.5	0.060	0.000	0.000	513628A974	[6541][3][2]
5000.0	0.039	0.000	0.000	516328A974	[6541][32]
5312.5	0.028	0.000	0.000	51632A8974	[6514][32]
5625.0	0.019	0.000	0.000	51632A8974	[6514][32]
5937.5	0.013	0.000	0.000	[5[1632A897]4]	[6514][32]
6250.0	0.010	0.000	0.000	[651[32A897]4]	[6514][23]
6562.5	0.009	0.000	0.000	[651[32A897]4]	[6514][23]
6875.0	0.007	0.000	0.000	[6513[2A897]4]	[6514][23]
7187.5	0.005	0.000	0.000	[6513[2A897]4]	[6514][23]
7500.0	0.005	0.000	0.000	[6513[2A897]4]	[6514][23]

No.	List Size	Source	Destination
		Window Size	Window Size
1	15	0.04	0.20
2	30	0.04	0.20
3	7	0.04	0.20
4	15	0.008	0.40
5	15	0.20	0.04
6	15	0.12	0.12
7	15	0.20	0.20
8	15	0.04	0.10
9	15	0.02	0.20
A	15	0.02	0.40

Table A4-6. Revised Window Tabu Search with Early/Tardy Extension ANOVA

Time	ANOVA			Tukey One-Way	
	One-Way Method	One-Way Problem	Two-Way Interaction	Method	Problem
0.0	0.988	0.000	1.000	26734589AB1	[54[1]6][3]2
312.5	0.997	0.000	0.590	47A2B136589	[65[4]1][3]2
625.0	0.999	0.000	0.084	74A623B1589	[564][1][3]2
937.5	0.999	0.000	0.073	47623A18B59	[654][1][3]2
1250.0	0.999	0.000	0.003	476A371589B	[654][1][3]2
1562.5	0.999	0.000	0.003	7461A32598B	[654][1][3]2
1875.0	0.999	0.000	0.001	4761A23589B	[65[4]1][3]2
2187.5	0.999	0.000	0.000	67412A8539B	[65[4]1][3]2
2500.0	0.999	0.000	0.000	764128A539B	[65[4]1][3]2
2812.5	0.999	0.000	0.000	7641285A39B	[65[4]1][3]2
3125.0	1.000	0.000	0.000	67124589A3B	[65[4]1][3]2
3437.5	1.000	0.000	0.000	67125489A3B	[6[54]1][3]2
3750.0	0.997	0.000	0.000	76125894A3B	[6541][3]2
4062.5	0.993	0.000	0.000	76125984A3B	[6541][3]2
4375.0	0.982	0.000	0.000	76159824A3B	[6541][3]2
4687.5	0.929	0.000	0.000	76159284A3B	[6514][3]2
5000.0	0.854	0.000	0.000	76519284A3B	[6514][3]2
5312.5	0.792	0.000	0.000	[7[659182A43]B]	[6514][3]2
5625.0	0.634	0.000	0.000	[76[59182A43]B]	[6514][32]
5937.5	0.519	0.000	0.000	[76598[12A43]B]	[6514][32]
6250.0	0.421	0.000	0.000	[7698512[A43]11]	[6514][32]
6562.5	0.341	0.000	0.000	[7695821[A43]B]	[6514][32]
6875.0	0.289	0.000	0.000	[7698521[A43]B]	[6514][23]
7187.5	0.211	0.000	0.000	[7695821[A43]B]	[6514][23]
7500.0	0.211	0.000	0.000	[7695821[A43]B]	[6514][23]

No.	List Size	Source Window Size	Destination Window Size
1	15	0.04	0.20
2	7	0.04	0.20
3	7	0.02	0.20
4	7	0.04	0.10
5	3	0.04	0.20
6	7	0.20	0.04
7	7	0.12	0.12
8	1	0.04	0.20
9	3	0.12	0.12
A	7	0.02	0.40
B	7	0.01	0.40

Table A4-7. Early/Tardy Hybrid ANOVA

Time	One-Way Method	ANOVA		Tukey One-Way	
		One-Way Problem	Two-Way Interaction	Method	Problem
0.0	1.000	0.000	1.000	283AB145679CD	[5416][3]2
312.5	0.389	0.000	0.790	3674215A8CB9D	[54613]2
625.0	0.145	0.000	0.632	6234751AC8B9D	[54613]2
937.5	0.029	0.000	0.096	362475A1C8B9D	[54613]2
1250.0	0.069	0.000	0.000	36427A15C9BD8	[6[54][1]3]2
1562.5	0.070	0.000	0.000	3642A715CB9D8	[65[4][1]3]2
1875.0	0.028	0.000	0.000	3A46721CB59D8	[6[54]1]3]2
2187.5	0.015	0.000	0.000	[A[346721CB95D]8]	[6541][3]2
2500.0	0.008	0.000	0.000	3A4672C1B9D85	[6541][3]2
2812.5	0.013	0.000	0.000	A34672C1B95D8	[6541][3]2
3125.0	0.003	0.000	0.000	A34672CB19D58	[6541][3]2
3437.5	0.013	0.000	0.027	[A3467[2CB19D8]5]	[6514][3]2
3750.0	0.016	0.000	0.004	[A3467[2CB19D8]5]	[6514][3]2
4062.5	0.019	0.000	0.000	A3647C2B19D85	[6514][3]2
4375.0	0.025	0.000	0.000	[A3[746C2B19D8]5]	[6514]32
4687.5	0.033	0.000	0.000	A3764C2B195D8	[6514]32
5000.0	0.030	0.000	0.000	A3764C2B91D85	[6514]32
5312.5	0.024	0.000	0.000	A37C642B91D85	[6154]32
5625.0	0.002	0.000	0.000	[A37C64B[291D8]5]	[6154]32
5937.5	0.000	0.000	0.000	[A3C764B291D[8]5]	[6154]32
6250.0	0.000	0.000	0.000	[A3C764BC9[D18]5]	[6154]23
6562.5	0.000	0.000	0.000	[A3C67B492D[18]5]	[6154]23
6875.0	0.000	0.000	0.000	[A3C67B492D[18]5]	[6154]23
7187.5	0.000	0.000	0.000	[A3CB67492D1[8]5]	[1654]23
7500.0	0.000	0.000	0.000	[A3CB67492D1[8]5]	[1654]23

No.	E/T	List Size	Source	Destination
	Iterations		Window Size	Window Size
1	3	20	0.02	0.20
2	3	20	0.04	0.10
3	3	20	0.04	0.20
4	3	7	0.04	0.20
5	3	1	0.04	0.20
6	3	15	0.04	0.20
7	3	20	0.02	0.40
8	3	20	0.01	0.40
9	3	20	0.20	0.04
A	1	20	0.12	0.12
B	1	20	0.20	0.04
C	3	20	0.12	0.12
D	3	20	0.20	0.20

Table A4-8. Across Search Techniques ANOVA

Time	Method	ANOVA		Tukey One-Way	
		Problem	Interaction	Standard Methods	Best Methods
0.0	0.144	0.000	0.000	625143	478169235
312.5	0.003	0.000	0.000	[62513]4	[87[642913]5]
625.0	0.001	0.000	0.000	[62513]4	[678[94215]3]
937.5	0.000	0.000	0.000	[62513]4	[97864[215]3]
1250.0	0.000	0.000	0.000	[62513]4	[79864[215]3]
1562.5	0.000	0.000	0.000	[62513]4	[97864[215]3]
1875.0	0.000	0.000	0.000	[65213]4	[97864[251]3]
2187.5	0.000	0.000	0.000	[65213]4	[97864[521]3]
2500.0	0.000	0.000	0.000	[65213]4	[97864[521]3]
2812.5	0.000	0.000	0.000	[65213]4	[97864[521]3]
3125.0	0.000	0.000	0.000	[65213]4	[97846[521]3]
3437.5	0.000	0.000	0.000	[65213]4	[79846[521]3]
3750.0	0.000	0.000	0.000	[65213]4	[798465[21]3]
4062.5	0.000	0.000	0.000	[65213]4	[79846[521]3]
4375.0	0.000	0.000	0.000	[65213]4	[79846[521]3]
4687.5	0.000	0.000	0.000	[65213]4	[97846[521]3]
5000.0	0.000	0.000	0.000	[65213]4	[79846[521]3]
5312.5	0.000	0.000	0.000	[65213]4	[798456[21]3]
5625.0	0.000	0.000	0.000	[65213]4	[978465[21]3]
5937.5	0.000	0.000	0.000	[65213]4	[7[98465[2]1]3]
6250.0	0.000	0.000	0.000	[56213]4	[978[465[2]1]3]
6562.5	0.000	0.000	0.000	[56213]4	[987[456[2]1]3]
6875.0	0.000	0.000	0.000	[56213]4	[978[456[2]1]3]
7187.5	0.000	0.000	0.000	[56213]4	[897[456[2]1]3]
7500.0	0.000	0.000	0.000	[56213]4	[897[456[2]1]3]

No.	Technique
1	Standard SA E/T Start
2	E/T SA
3	E/T GA Heuristic Population Members
4	Standard GA Heuristic Population Members
5	Revised TS (15,0.008)
6	Revised E/T TS (1,0.08)
7	Revised Window TS ETS (15,0.20,0.04)
8	Revised Window TS E/T Extension (7,0.12,0.12)
9	E/T Hybrid TS (1,20,0.12,0.12)

Table A4-9. ANOVA Across Tabu Search Based Methods

Time	One-Way Method	ANOVA		Tukey One-Way	
		One-Way Problem	Two-Way Interaction	Method (4 Problems)	Method (10 Problems)
0.0	0.039	0.000	0.226	[78634[25]1]	[7683425]1
312.5	0.000	0.000	0.074	[7438265]1	[3748256]1
625.0	0.000	0.000	0.000	[7234856]1	[3724586]1
937.5	0.000	0.000	0.000	[5734286]1	[3547286]1
1250.0	0.000	0.000	0.000	[3547286]1	[3547286]1
1562.5	0.000	0.000	0.000	[534728[6]1]	[3547286]1
1875.0	0.000	0.000	0.000	[534728[6]1]	[35[4728]6]1
2187.5	0.000	0.000	0.000	[534728[6]1]	[35[4728]6]1
2500.0	0.000	0.000	0.000	[5347[286]1]	[35[4[72]8]6]1
2812.5	0.000	0.000	0.000	[5347[286]1]	[354[72]86]1
3125.0	0.000	0.000	0.000	[5347[286]1]	[35[4[7]2]86]1
3437.5	0.000	0.000	0.000	[3547[286]1]	[35[4[7]2]68]1
3750.0	0.000	0.000	0.000	[3547[826]1]	[354[7][26][8]1]
4062.5	0.000	0.000	0.000	[3547[862]1]	[354[7][26][8]1]
4375.0	0.000	0.000	0.000	[3547[862]1]	[354[7][6]281]
4687.5	0.000	0.000	0.000	[5347[682]1]	[354[7][6]281]
5000.0	0.000	0.000	0.000	[35[47[68]2]1]	[354[7][6]281]
5312.5	0.000	0.000	0.000	[354[7[6]82]1]	[354[7][6]281]
5625.0	0.000	0.000	0.000	[53[4[7[6]8]2]1]	[354[7][6]281]
5937.5	0.000	0.000	0.000	[354[7[6]8]21]	[354[7][6]281]
6250.0	0.000	0.000	0.000	[534[7[6]8]21]	[354[7][6]281]
6562.5	0.000	0.000	0.000	[543[7[6]8]12]	[354[7][6]218]
6875.0	0.000	0.000	0.000	[5347[6]812]	[345[7][6]128]
7187.5	0.000	0.000	0.000	[4537[6]182]	[543[7][6]128]
7500.0	0.000	0.000	0.000	[4537[6]182]	[543[7][6]128]

No.	Technique
1	Revised TS (15,0.008)
2	Revised E/T TS (1,0.08)
3	Revised Window TS ETS (15,0.2,0.04)
4	Revised Window TS E/T Extension (7,0.12,0.12)
5	E/T Hybrid TS (1,20,0.12,0.12)
6	Revised TS Ow Start Pt (15,0.008)
7	Revised Window TS Ow Start Pt (15,0.2,0.04)
8	Revised E/T TS Ow Start Pt (1,0.08)

APPENDIX V

STARTING POINT SENSITIVITY ANOVA RESULTS

Table A5-1. TABU SEARCH

Time	ANOVA			Tukey One-Way	
	One-Way Method	One-Way Problem	Two-Way Interaction	Method	Problem
0.0	0.000	0.000	0.000	[396045][71]28]	[41][56]2]3
312.5	0.000	0.000	0.000	[63907845]12	[45612]3
625.0	0.000	0.000	0.000	[63907845]12	[45612]3
937.5	0.000	0.000	0.000	[63907845]12	[54612]3
1250.0	0.000	0.000	0.000	[39607845]12	[56412]3
1562.5	0.000	0.000	0.000	[39608745]12	[65412]3
1875.0	0.000	0.000	0.000	[39608745]12	[65412]3
2187.5	0.000	0.000	0.000	[39608745]12	[65412]3
2500.0	0.000	0.000	0.000	[39608745]12	[65412]3
2812.5	0.000	0.000	0.000	[39680745]12	[65412]3
3125.0	0.000	0.000	0.000	[39680745]12	[65412]3
3437.5	0.000	0.000	0.000	[39867045]12	[6541][2]3]
3750.0	0.000	0.000	0.000	[39870645]12	[6541][2]3]
4062.5	0.000	0.000	0.000	[39870645]12	[6541][2]3]
4375.0	0.000	0.000	0.000	[39870645]12	[6541][2]3]
4687.5	0.000	0.000	0.000	[39870645]12	[6541][2]3]
5000.0	0.000	0.000	0.000	[39870645]12	[6541][2]3]
5312.5	0.000	0.000	0.000	[39870645]12	[65[41]][2]3]
5625.0	0.000	0.000	0.000	[39870645]12	[6541]23
5937.5	0.000	0.000	0.000	[39870645]12	[6541]23
6250.0	0.000	0.000	0.000	[39870645]12	[6541]23
6562.5	0.000	0.000	0.000	[39870645]21	[6541]23
6875.0	0.000	0.000	0.000	[39870645]21	[6541]23
7187.5	0.000	0.000	0.000	[39780645]21	[6541]23
7500.0	0.000	0.000	0.000	[39780645]21	[6541]23

No.	Technique
0	Revised Window E/T EDD Start (15,0.04,0.20)
1	Revised Window TS Good Start (15,0.04,0.20)
2	Revised Window TS Bad Start (15,0.04,0.20)
3	Revised Window TS E/T EDD Start (15,0.04,0.20)
4	Revised E/T TS Good Start (1,0.04)
5	Revised E/T TS Bad Start (1,0.04)
6	Revised E/T TS E/T EDD Start (1,0.04)
7	Revised ET Ext. Good Start (15,0.04,0.20)
8	Revised ET Ext. Bad Start (15,0.04,0.20)
9	Revised ET Ext. E/T EDD Start (15,0.04,0.20)

APPENDIX VI

30 JOB PROBLEM ANALYSIS

The following cost graphs show the cost of each job in each position in the schedule, starting with the cost of the first jobs scheduled at the base of the graph, through to the cost of the last job in the schedule. Each column therefore builds up to show the total cost of the schedule.

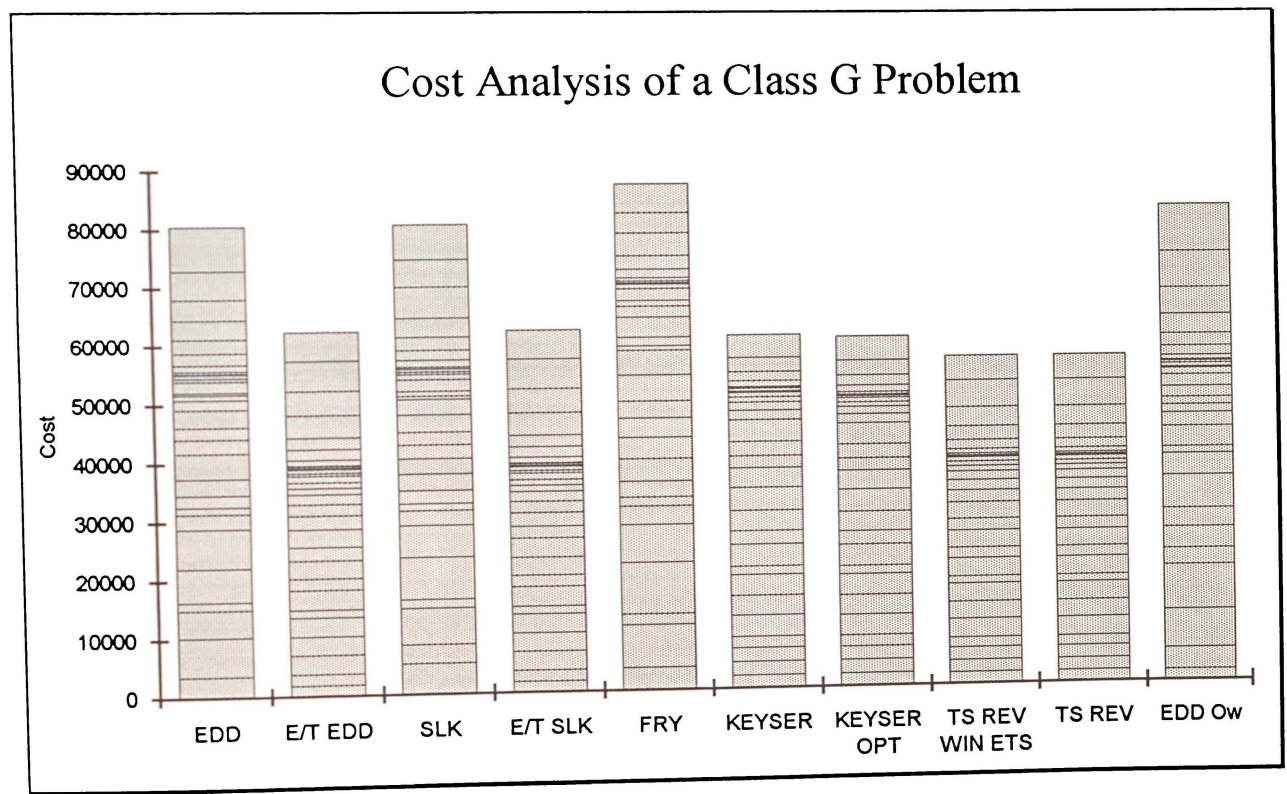


Figure A6-1.

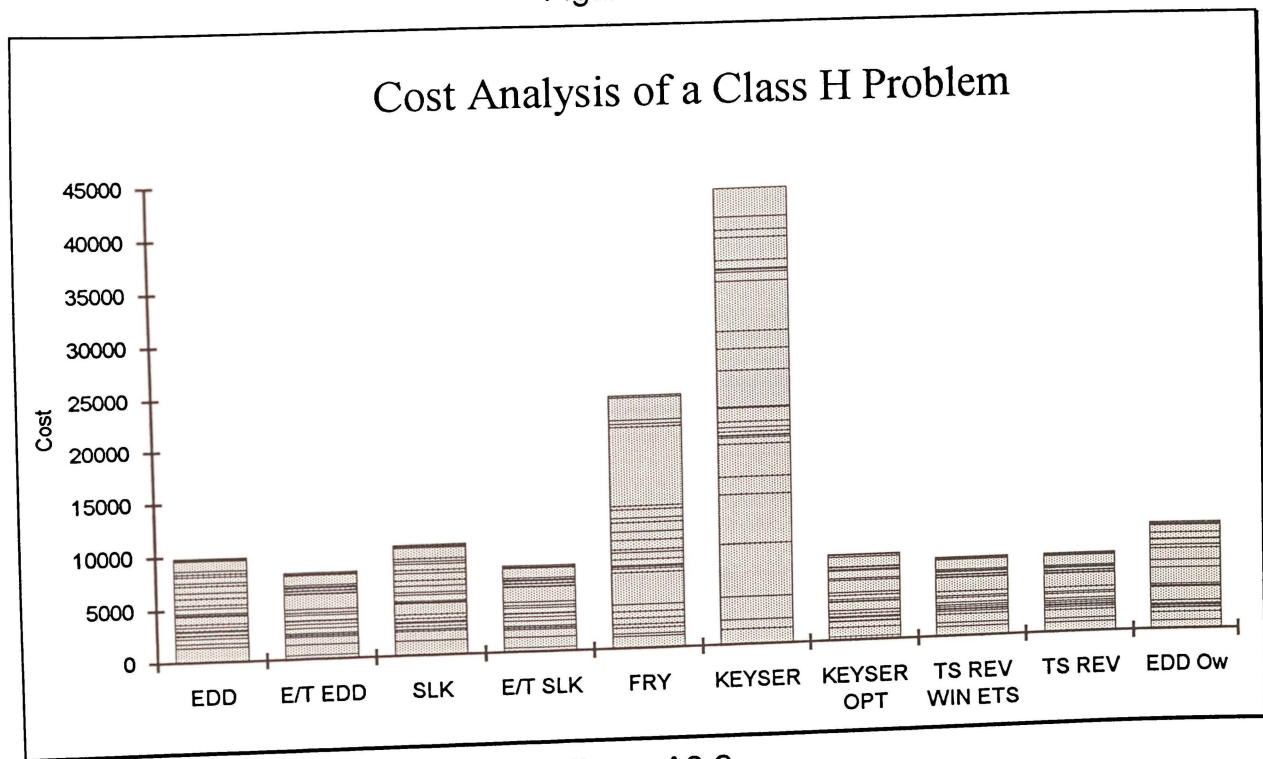


Figure A6-2.

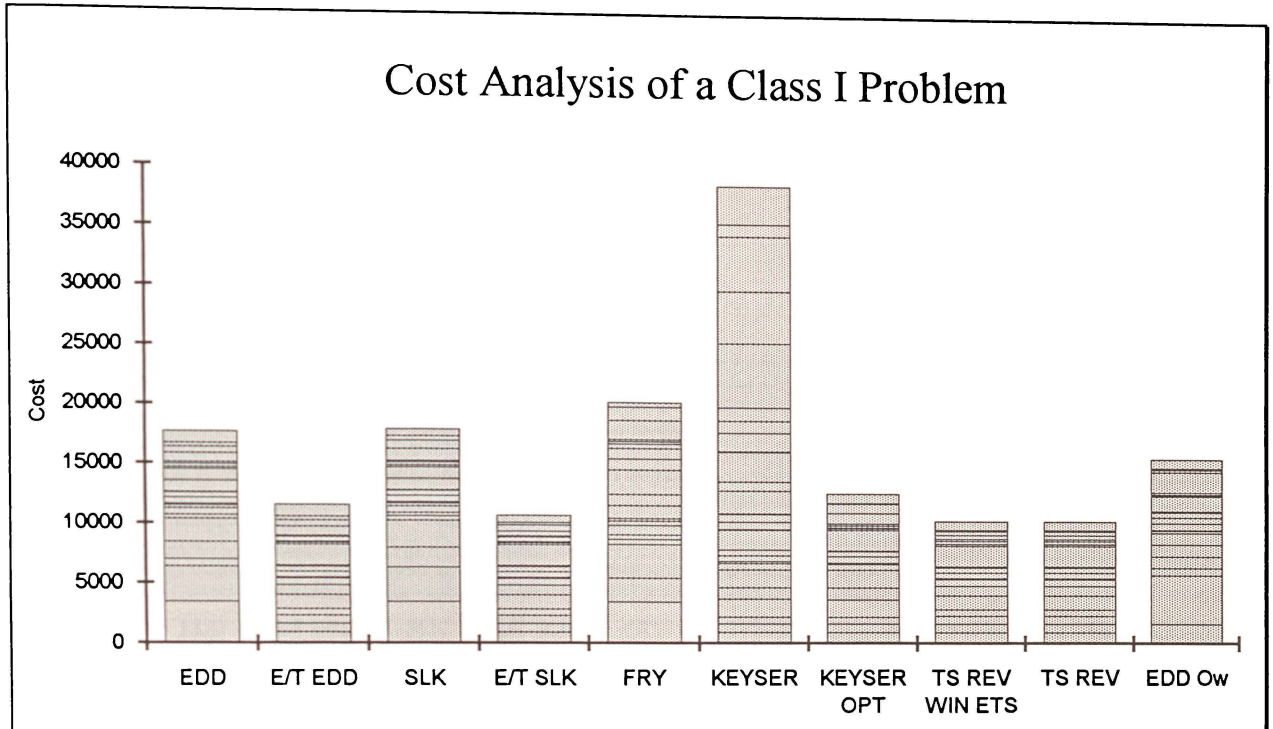


Figure A6-3.

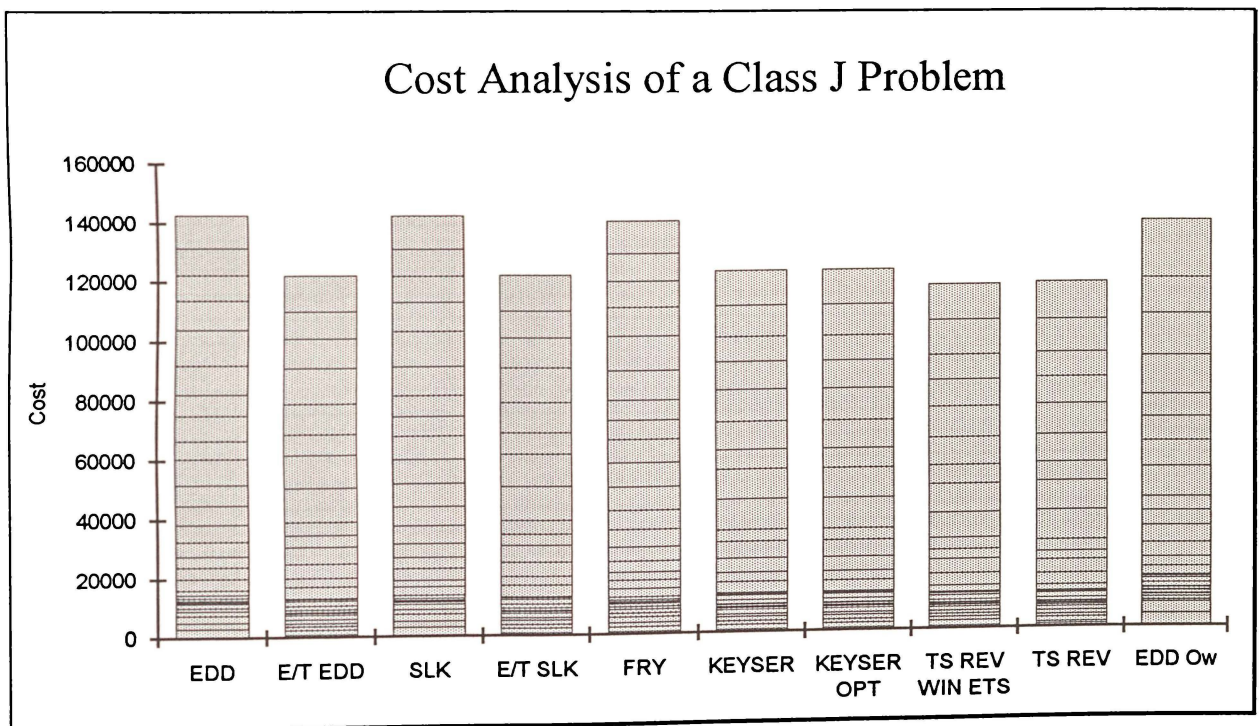


Figure A6-4.

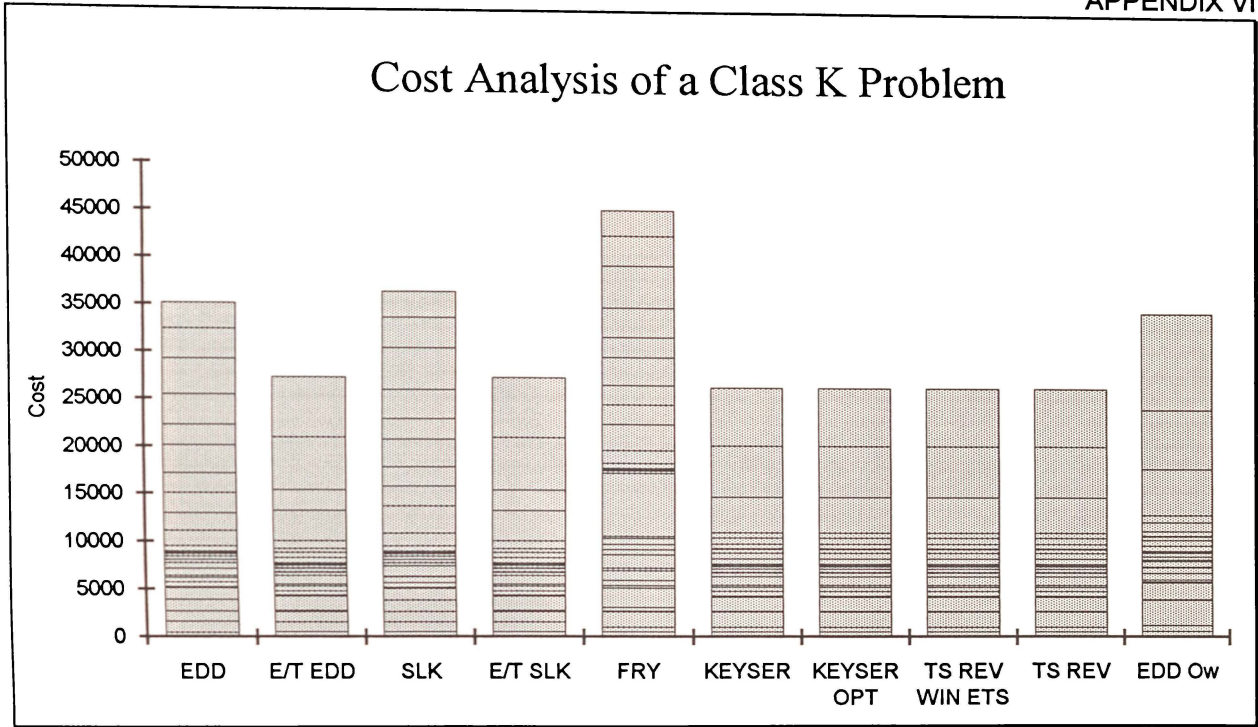


Figure A6-5.

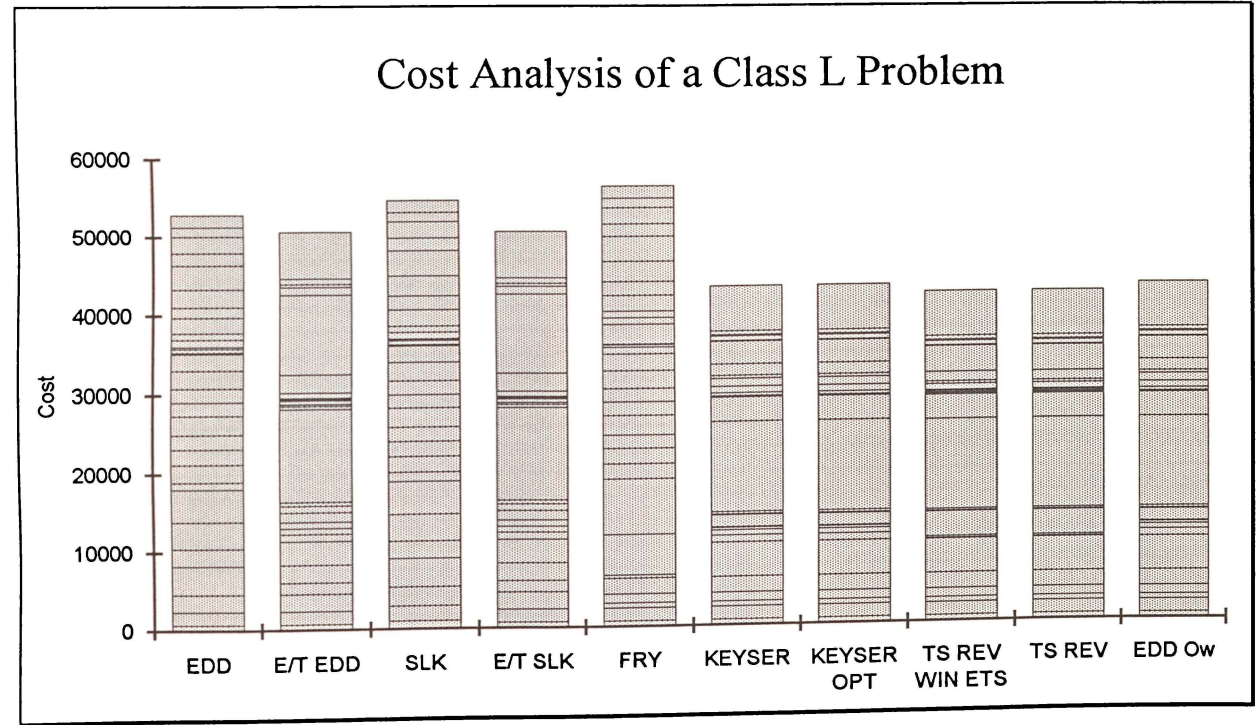


Figure A6-6.

Figure A6-7. Schedules of a G Class Problem



Figure A6-8. Schedules of a H Class Problem

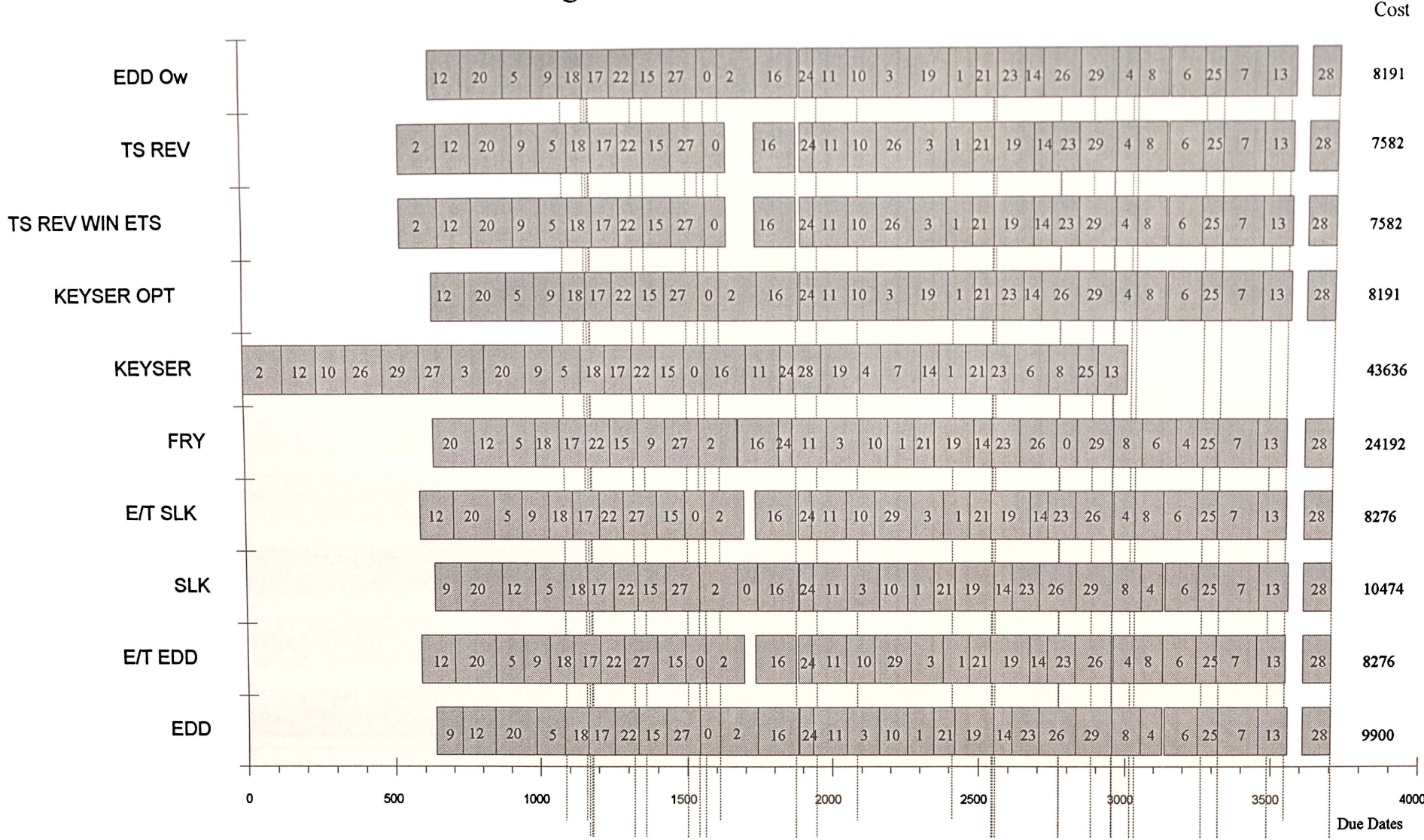


Figure A6-9. Schedules of an I Class Problem



Figure A6-10. Schedules of a J Class Problem

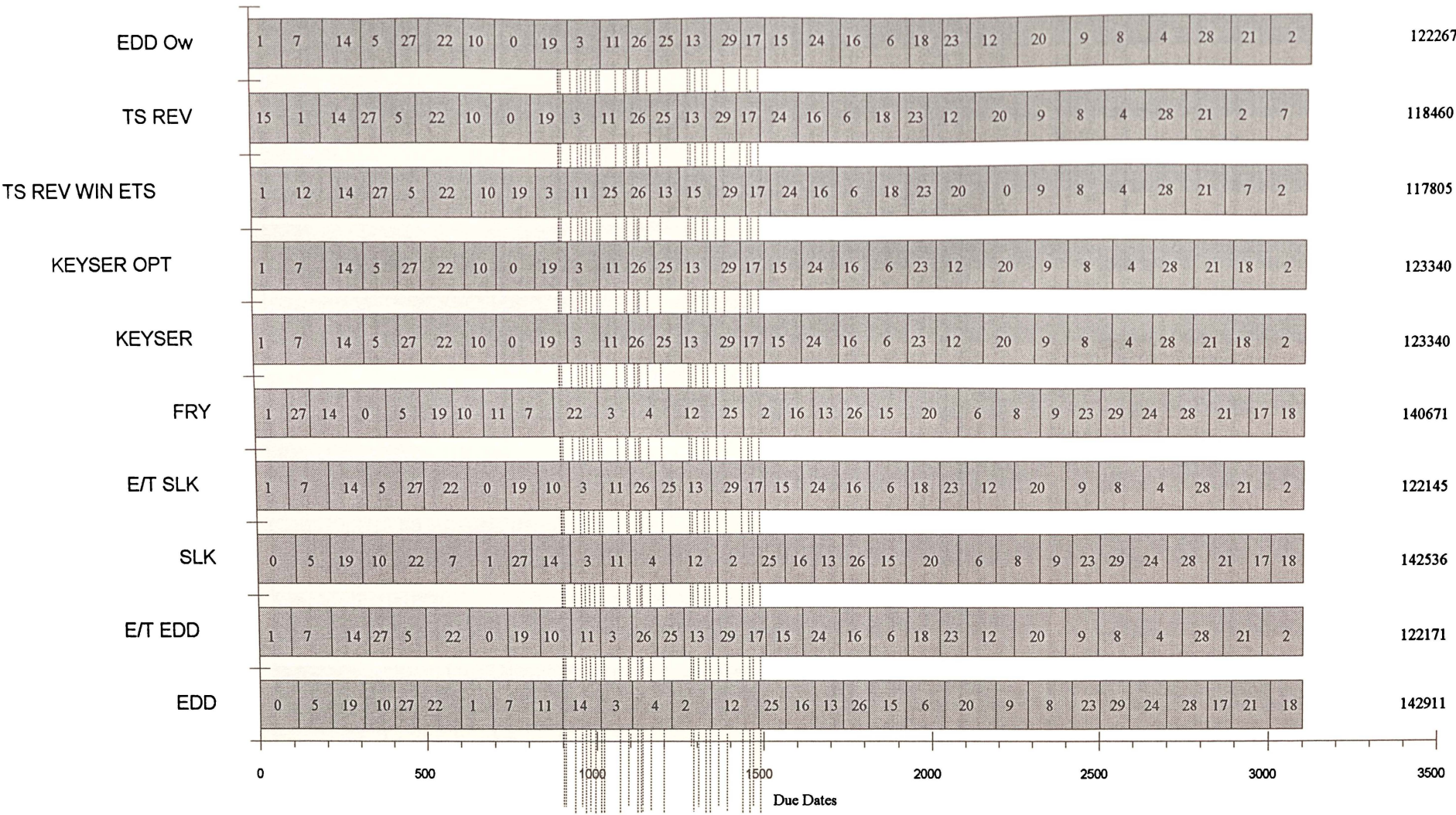


Figure A6-11. Schedules of a K Class Problem

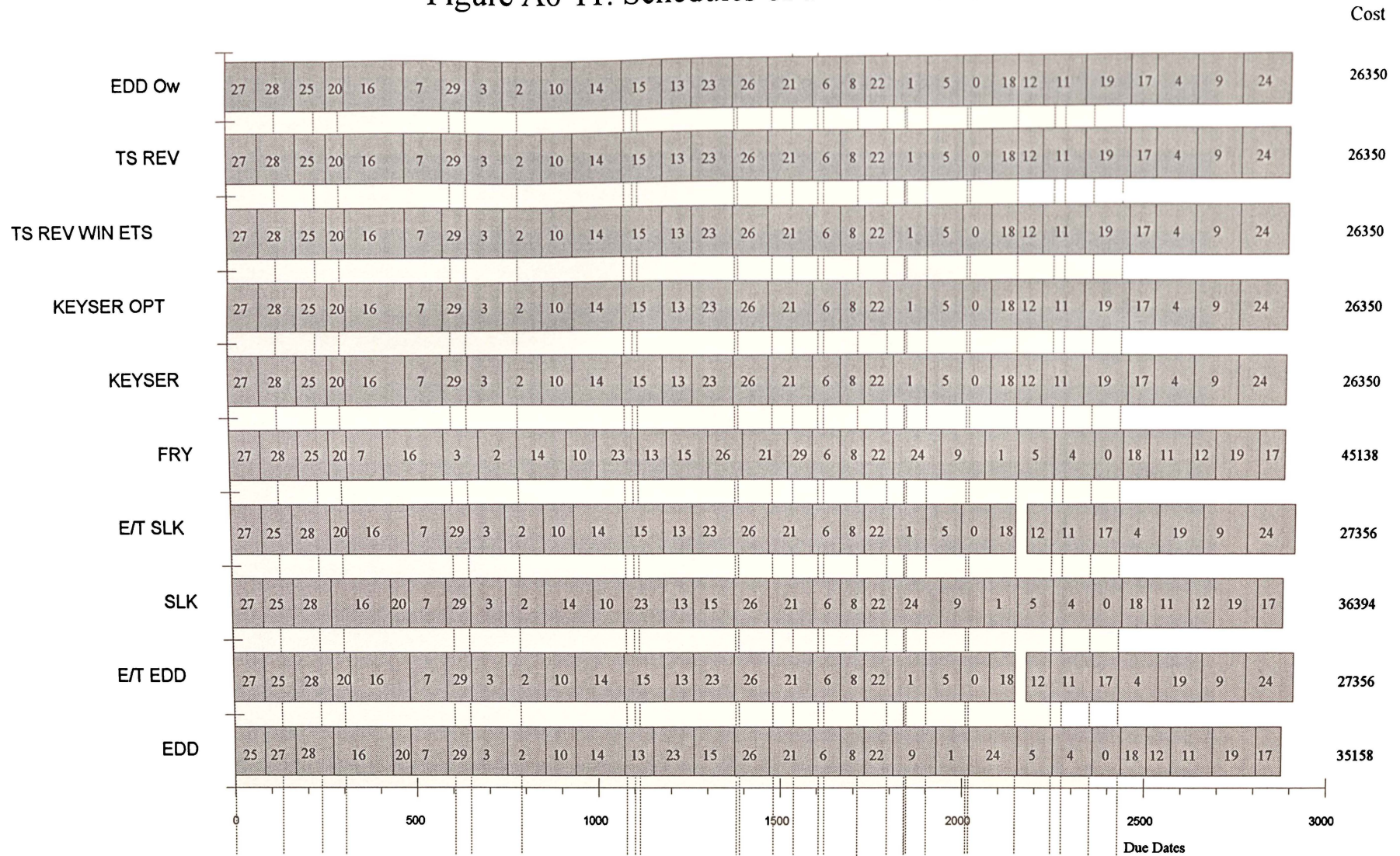


Figure A6-12. Schedules of a L Class Problem

Cost



Bibliography

- Abdul-Razaq T.S. and Potts C.N. (1988); "Dynamic Programming State Space Relaxation for Single-Machine Scheduling", *Journal of the Operational Research Society*, Vol 39, No 2, pp 141-152
- Abramson D. (1992); "A Very High Speed Architecture for Simulated Annealing", *Computer*, Vol 25, No 5, pp 27-36
- Adams J., Balas E. and Zawack D. (1988); "The Shifting Bottleneck Procedure for Job Shop Scheduling", *Management Science*, Vol 34, No 3, pp 391-401
- Adenso-Díaz B. (1992); "Restricted Neighbourhood in the Tabu Search for the Flowshop Problem", *European Journal of Operational Research*, Vol 62, pp 27-37
- Ahmed M.U. and Sundararaghavan P.S. (1990); "Minimizing the Weighted Sum of Late and Early Completion Penalties in a Single Machine", *IIE Transactions*, Vol 22, No 3, pp 288-290
- Bacchi U., Sullivan R.S. and Chang Y-L. (1986); "Minimising Mean Absolute Deviation of Completion Times about a Common Due Date", *Naval Research Logistics Quarterly*, Vol 33, pp 227-240
- Bagchi U., Sullivan R.S. and Chang Y-L (1987); "Minimizing Mean Squared Deviation of Completion Times about a Common Due Date", *Management Science*, Vol 33, No 7, pp 894-906
- Baker K.R. (1974); *Introduction to Sequencing and Scheduling*, John Wiley & Sons.
- Baker K.R. and Martin J.B. (1974); "An experimental Comparison of Solution Algorithms for the Single-Machine Tardiness Problem", *Naval Research Logistics Quarterly*, Vol 21, pp 187-199
- Baker K.R. and Scudder G.D. (1990); "Sequencing with Earliness and Tardiness Penalties: A Review", *Operations Research*, Vol 38, No 1, pp 22-36
- Barnes J.W. and Laguna M. (1991a); "A Review and Synthesis of Tabu Search Applications to Production Scheduling Problems", *Technical Report ORP91-05*, University of Texas at Austin
- Barnes J.W. and Laguna M. (1991b); "Solving the Multiple-Machine Weighted Flow Time Problem using Tabu Search", *IIE Transactions*, to appear
- Bector C.R., Gupta Y.P., and Gupta M.C. (1988); "Determination of an Optimal Common Due Date and Optimal Sequence in a Single Machine Job Shop", *International Journal of Production Research*, Vol 26, No 4, pp 613-628
- Biegel J.E. and Davern J.J. (1990); "Genetic Algorithms and Job Shop Scheduling", *Computers and Industrial Engineering*, Vol 19, No 1-4, pp 81-91
- Booker L. (1987); "Improving Search in Genetic Algorithms", Davis L. (ed), *Genetic Algorithms and Simulated Annealing*, Pitman, pp 61-73
- Chand S. and Schneeberger H. (1988); "Single machine scheduling to minimise weighted earliness subject to no tardy jobs", *European Journal of Operational Research*, Vol 34, pp 221-230
- Cheng T.C.E. (1984); "Optimal Due-Date Determination and Sequencing of n Jobs on a Single Machine", *Journal of the Operational Research Society*, Vol 35, No 5, pp 433-437

- Cheng T.C.E. (1991); "Optimal Assignment of Total-work-content Due-dates and Sequencing in a Single-machine Shop", *Journal of the Operational Research Society*, Vol 42, No 2, pp 177-181
- Cheng T.C.E. and Gupta M.C. (1989); "Survey of scheduling research involving due date determination decisions", *European Journal of Operational Research*, Vol 38, pp 156-166
- Coleman B.J. (1992); "Technical Note: A Simple Model for Optimizing the Single Machine Early/Tardy Problem with Sequence Dependent Setups", *Production and Operations Management*, Vol 1, No 2, pp 225-228
- Dagli C. and Sittisathanchai (1993); "Genetic Neuro-Scheduler for Job Shop Scheduling", *Computers and Industrial Engineering*, Vol 25, No 1-4, pp 267-270
- Dantzig G.B. (1963); *Linear Programming and Extensions*, Princeton University Press
- Davis J.S. and Kanet J.J. (1993); "Single-Machine Scheduling with Early and Tardy Completion Costs", *Naval Research Logistics*, Vol 40, pp 85-101
- Davis L. (1991); *Handbook of Genetic Algorithms*, Van Nostrand Reinhold
- Davis L. and Steenstrup M. (1987); "Genetic Algorithms and Simulated Annealing: An Overview", Davis L. (ed), *Genetic Algorithms and Simulated Annealing*, Pitman, pp 1-11
- De P., Ghosh J.B. and Wells C.E. (1993); "On the General Solution for a Class of Early/Tardy Problems", *Computers and Operations Research*, Vol 20, No 2, pp 141-149
- de Werra D. and Hertz A. (1989); "Tabu Search Techniques: A Tutorial and application to Neural Networks", *OR Spektrum*, Vol. 11, pp 131-141
- Dorndorf U. and Pesch E. (1992); "Evolution Based Learning in a Job Shop Scheduling Environment", *Working Paper, University of Limburg, The Netherlands*
- Eglese R.W., (1990); "Simulated Annealing: A tool for Operational Research", *European Journal of Operational Research*, Vol 46, pp 271-281
- Eglese R.W. and Rand G.K. (1987); "Conference Seminar Timetabling", *Journal of the Operational Research Society*, Vol 38, No 7, pp 591-598
- Ferris M.C. and Vlach M. (1992); "Scheduling with Earliness and Tardiness Penalties", *Naval Research Logistics*, Vol 39, pp 229-245
- Fry T.D., Armstrong R.D. and Blackstone J.H. (1987); "Minimizing Weighted Absolute Deviation in Single Machine Scheduling", *IIE Transactions*, Vol 19, No 4, pp 445-450
- Fry T.D., Leong G.K. and Rakes T.R. (1987); "Single Machine Scheduling: A Comparison of Two Solution Procedures", *Omega International Journal of Management Science*, Vol 15, No 4, pp 277-282
- Games P.A. and Howell J.F. (1976); "Pairwise Multiple Comparison Procedures with Unequal N's and/or Variances: A Monte Carlo Study", *Journal of Educational Statistics*, Vol 1, pp 113-125
- Garey M.R. and Johnson D.S. (1979); *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman and Company

- Garey M.R., Tarjan R.E. and Wilfong G.T. (1988); "One-Processor Scheduling with Symmetric Earliness and Tardiness Penalties", *Mathematics of Operations Research*, Vol 13, No 2, pp 330-348
- Glover F. (1977); "Heuristics for Integer Programming using Surrogate Constraints", *Decision Sciences*, Vol 8, pp 156-166
- Glover F. (1986); "Future Paths for Integer Programming and Links to Artificial Intelligence", *Computers and Operations Research*, Vol 15, No 5, pp 533-549
- Glover F. (1989); "Tabu Search - Part I", *ORSA Journal on Computing*, Vol 1, No 3, pp. 190-206
- Glover F. (1990a); "Tabu Search - Part II", *ORSA Journal on Computing*, Vol 2, No 1, pp 4-32
- Glover F. (1990b); "Artificial Intelligence, Heuristic Frameworks and Tabu Search", *Managerial and Decision Economics*, Vol 11, pp 365-375
- Glover F. (1990c); "Tabu Search: A Tutorial", *Working Paper*, University of Colorado at Boulder, February.
- Glover F. (1991); Memorandum on Developments and Speculations, *University of Colorado at Boulder*, May 3
- Glover F. and Greenberg H.J. (1989); "New Approaches for Heuristic Search: A Bilateral Linkage with Artificial Intelligence", *European Journal of Operational Research*, Vol 39, pp 119-130
- Glover F. and Laguna M. (1993); "Tabu Search", *Modern Heuristic Techniques for Combinatorial Problems*, Reeves C.R. (ed), Blackwell Scientific Publications, pp 70-150
- Goldberg D.E. (1989); *Genetic Algorithms in Search, Optimization & Machine Learning*, Addison-Wesley
- Goldratt E.M and Cox J. (1986); *The Goal: A process of ongoing improvement*, North River Press
- Graham R.L., Lawler E.L., Lenstra J.K. and Rinnooy Kan A.H.G. (1979); "Optimization and Approximation in Deterministic Sequencing and Scheduling: A Survey", *Annals of Discrete Mathematics*, Vol 5, pp 287-326
- Grant T.J. (1986); "Lessons for O.R. from A.I.: A Scheduling Case Study", *Journal of the Operational Research Society*, Vol 37, No 1, pp 41-57
- Grefenstette J.J. (1987); "Incorporating Problem Specific Knowledge into Genetic Algorithms", Davis L. (ed), *Genetic Algorithms and Simulated Annealing*, Pitman, pp 42-60
- Gue F., Heard E., Maranka P. and McEneny T. (1987); "Production Activity Control", *Production & Inventory Control Handbook*, Second Edition, McGraw-Hill, Greene J.H. (Ed), pp 16.1-16.57
- Hall N.G., Kubiak W. and Sethi S.P. (1991); "Earliness-Tardiness Scheduling Problems, II: Deviation of Completion Times about a Restrictive Common Due Date", *Operations Research*, Vol 39, No 5, pp 847-856
- Hall N.G., Sethi S.P. and Sriskandarajah C. (1991); "On the complexity of generalised due date scheduling problems", *European Journal of Operational Research*, Vol 51, pp 100-109

- Hall N.G. (1986); "Single- and Multiple-Processor Models for Minimizing Completion Time Variance", *Naval Research Logistics Quarterly*, Vol 33, pp 49-54
- Hall N.G. and Posner M.E. (1991), "Earliness-Tardiness Scheduling Problems, I: Weighted Deviation of Completion Times about a Common Due Date", *Operations Research*, Vol 39, No 5, pp 836-846
- Herrmann J.W., Lee C-Y. Hinchman J. (1993); "Global Job Shop Scheduling with Genetic Algorithm", *Working Paper, Department of Industrial and Systems Engineering, University of Florida*, September
- Herrmann J.W. and Lee C-Y. (1993a); "Solving a Class Scheduling Problem with a Genetic Algorithm", *to appear in ORSA Journal of Computing*
- Herrmann J.W. and Lee C-Y. (1993b); "Problem Space Genetic Algorithms for One-machine Scheduling", *Working Paper, Department of Industrial and Systems Engineering, University of Florida*, October
- Holland J.H. (1975); *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*, University of Michigan Press
- Howell D.C. (1992); *Statistical Methods for Psychology 3rd Edition*, Duxbury Press
- Hübscher R. and Glover F. (1992); "Applying Tabu Search with Influential Diversification to Multiprocessor Scheduling", *Working Paper, University of Colorado at Boulder*, October 11
- James R.J.W. (1992); "On the Optimal Positioning of a Sequence of Jobs with Distinct Due Dates which incur Earliness and Tardiness Penalties", *Research Report 1992-23, Department of Management Systems, University of Waikato*
- John J.A. and Whitaker D. (1992); "Construction of Resolvable Row-Column Designs Using Simulated Annealing", *Journal of the Australian Statistical Association*, to appear
- Johnson D.S., Aragon C.R., McGeoch L.A., Schevon C. (1989); "Optimization by Simulated Annealing: An Experimental Evaluation; Part I, Graph Partitioning", *Operations Research*, Vol 37, No 6, pp 865-892
- Kanet J.J. (1981); "Minimizing the Average Deviation of Job Completion Times about a Common Due Date", *Naval Research Logistics Quarterly*, Vol 28, No 4, pp 643-651
- Keyser T.K. and Sarper H. (1991); "A Heuristic Solution of the E/T Problem with Waiting Costs and Non-Zero Release Times", *Computers and Industrial Engineering*, Vol 21, No 1-4, pp 297-301
- Kim Y-D (1993); "Heuristics for Flow shop Scheduling Problems Minimizing Mean Tardiness", *Journal of the Operational Research Society*, Vol 44, No 1, pp 19-28
- Kirkpatrick S., Gelatt C.D. Jr, Vecchi M.P. (1983); "Optimisation by Simulated Annealing", *Science*, Vol 220, No 4598, pp 671-680
- Klockars A.J and Sax G. (1986); *Multiple Comparisons*, Sage Publications
- Kramer F-J and Lee C-Y (1992); "Common Due Window Scheduling", *Research Report 92-5, Department of Industrial Systems and Engineering, University of Florida*

- Kubiak W., Lou S., and Sethi S. (1990); "Equivalence of Mean Flow Time Problems and Mean Absolute Deviation Problems", *Operations Research Letters*, Vol 9, pp 371-374
- Laguna M., Barnes J.W. and Glover F. (1991); "Tabu Search Methods for a Single Machine Scheduling Problem", *Journal of Intelligent Manufacturing*, Vol 2, pp 63-74
- Laguna M. and Glover G. (1993); "Integrated Target Analysis and Tabu Search for Improved Scheduling Systems", *Expert Systems with Applications*, Vol 6, pp 287-297
- Laguna M. and González Velarde J.L. (1991); "A Search Heuristic for Just-In-Time Scheduling in Parallel Machines", *Journal of Intelligent Manufacturing*, Vol. 2, pp. 253-260
- Lash S. (1993); "Genetic Algorithms for Weighted Tardiness Scheduling on Parallel Machines", *Technical Report 93-01, Department of Industrial Engineering, Northwestern University*
- Lawton G. (1992); "Genetic Algorithms for Schedule Optimization", *AI Expert*, Vol 25, No 5, pp 23-27
- Lee C.Y. and Kim S.J. (1992); "Parallel Genetic Algorithms for the Arbitrarily Weighted Earliness-Tardiness Job Scheduling Problem", *Working Paper, Korea Advanced Institute of Science & Technology*
- Lundy and Mees (1986); "Convergence of an Annealing Algorithm", *Mathematical Programming*, Vol 34, pp 111-124
- Mason A.J. (1992); *Genetic Algorithms and Scheduling Problems*, Unpublished PhD Dissertation, Darwin College, University of Cambridge
- Matsuo H., Suh C.J. and Sullivan R.S. (1989); "A Controlled Search Simulated Annealing Method for the Single Machine Weighted Tardiness Problem", *Annals of Operations Research*, Vol 21, pp 85-108
- Maxwell S.E. and Delaney H.D. (1990); *Designing Experiments and Analysing Data: A Model Comparison Perspective*, Brooks/Cole Publishing Co., pp 181-186
- McKay K.N., Safaynei F.R. and Buzacott J.A. (1988); "Job-Shop Scheduling Theory: What Is Relevant?", *Interfaces*, Vol 18, pp 84-90
- Metropolis W., Rosenbluth A., Rosenbluth A., Teller A. and Teller E. (1953); "Equation of State Calculations by Fast Computing Machines", *Journal of Chemical Physics*, Vol 21, pp 1087-1092
- Mittenthal J., Raghavachari M. and Rana A.I. (1993); "A Hybrid Simulated Annealing Approach for Single Machine Scheduling Problems with Non-Regular Penalty Functions", *Computers and Operations Research*, Vol 20, No 2, pp 103-111
- Oakland J.S. and Sohal A. (1987); "Production and Operations Management Techniques in Manufacturing: Comparing the United Kingdom and the United States", *Production and Inventory Management*, Third Quarter, pp 12-15
- Ogbu F.A. and Smith D.K. (1990); "The Application of the simulated Annealing Algorithm to the Solution of the $n/m/C_{max}$ Flow shop Problem", *Computers and Operations Research*, Vol 17, No 3, pp 243-253
- Osman I.H. and Potts C.N. (1989); "Simulated Annealing for Permutation Flow-shop Scheduling", *OMEGA*, Vol 17, No 16, pp 551-557

- Ow P.S. and Morton T.E. (1989); "The Single Machine Early/Tardy Problem", *Management Science*, Vol 35, No 2, pp 177-191
- Papadimitriou C.H. and Steiglitz K. (1982), *Combinatorial Optimization: Algorithms and Complexity*, Prentice-Hall
- Potts C.N. and van Wassenhove L.N. (1991); "Single Machine Tardiness Sequencing Heuristics", *IIE Transactions*, Vol 23, No 4, pp 346-354
- Reeves C.R. (1993); "Improving the Efficiency of Tabu Search for Machine Sequencing Problems", *Journal of the Operational Research Society*, Vol 44, No 4, pp 375-382
- Robb D.J. and Silver E.A. (1994); "Scheduling in a Management Context: Uncertain Processing Times and Non-Regular Performance Measures", *Decision Sciences*, Vol 24, No 6, pp 1085-1107
- Sarin S.C., Erel E. and Steiner G. (1991); "Sequencing jobs on a single machine with a common due date and stochastic processing times", *European Journal of Operational Research*, Vol 51, pp 188-198
- Schniederjans M.J. (1993); *Topics in Just-In-Time Management*, Allyn and Bacon
- Schonberger R.J. (1982), *Japanese Manufacturing Techniques: Nine Hidden Lessons in Simplicity*, Collier MacMillan Publishers
- Sidney J.B. (1977); "Optimal Single Machine Scheduling with Earliness and Tardiness Penalties", *Operations Research*, Vol 25, No 1, pp 62-69
- Skorin-Kapov J. and Vakharia A.J. (1992); "Scheduling a Flow-Line Manufacturing Cell: A Tabu Search Approach", *Working Paper, State University of New York at Stony Brook*, March 5.
- Storer R.H., Wu D. and Vaccari R. (1992); "New Search Spaces for Scheduling Problems with Application to Job Shop Scheduling", *Management Science*, Vol 38, No 10, pp 1495-1509
- Sundararaghavan P.S. and Ahmed M.U. (1984); "Minimizing the Sum of Absolute Lateness in Single-machine and Multi-machine Scheduling", *Naval Research Logistics Quarterly*, Vol 31, pp 325-333
- Syswerda G. (1991); "Schedule Optimization Using Genetic Algorithms", Davis L. (ed), *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, pp 332-349
- Szwarc W. (1989); "Single-Machine Scheduling to Minimize Absolute Deviation of Completion Times from a Common Due Date", *Naval Research Logistics*, Vol 36, pp 663-673
- Taillard E. (1990); "Some Efficient Heuristic Methods for the Flow Shop Sequencing Problem", *European Journal of Operational Research*, Vol. 47, pp 65-74
- ten Kate H.A., Wijngaard J. and Zijm W.H.M. (1991); "Minimising Total Weighted Earliness, Tardiness and Set up costs", *Research Report RR1991-12, School of Management and Organisation, The Netherlands*
- Tsubakitani S. and Evans J.R. (1992); "Applying Tabu Search to the Mean Tardiness Sequencing Problem", *Working Paper, University of Cincinnati*, April.
- Vakharia A.J. and Chang Y-L. (1990); "A Simulated Annealing Approach to Scheduling a Manufacturing Cell", *Naval Research Logistics*, Vol 37, pp 559-577

- van Laarhoven P.J.M., Aarts E.H.L. and Lenstra J.K. (1992); "Job Shop Scheduling by Simulated Annealing", *Operations Research*, Vol 40, No 1, pp 113-125
- van Laarhoven P.J.M. and Aarts E.H.L. (1987); *Simulated Annealing: Theory and Applications*, D. Reidel
- Vempati V.S., Chen C-L and Bullington S.F. (1993); "An Effective Heuristic for Flow Shop Problems with Total Flow Time as Criterion", *Computers and Industrial Engineering*, Vol 25, No 1-4, pp 219-222
- Whitley D., Starkweather T. and Shaner D. (1991); "The Traveling Salesman and Sequence Scheduling: Quality Solutions Using Genetic Edge Recombination", Davis L. (ed), *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, pp 350-372
- Widmer M. (1991); "Job Shop Scheduling with Tooling Constraints: a Tabu Search Approach", *Journal of the Operational Research Society*, Vol 42, No 1, pp 75-82
- Widmer M. and Hertz A. (1989); "A New Heuristic Method for the Flow Shop Sequencing Problem", *European Journal of Operational Research*, Vol 41, pp 186-193
- Wiggins R. (1992); "Docking a Truck: A Genetic Fuzzy Approach", *AI Expert*, Vol 25, No 5, pp 28-35
- Yano C.A. and Kim Y-D (1991); "Algorithms for a class of single-machine weighted tardiness and earliness problems", *European Journal of Operational Research*, Vol 52, pp 167-178