

43054



To Ma and Deuta

- a New Year Gift

**Formulation of an Optimized Algorithm for
Resource Scheduling and Allocation in Projects**

:

A Genetic Algorithms Approach

*A thesis submitted in part fulfillment of the
requirements for award of the degree of
Doctor of Philosophy*

23

Tridib Ranjan Sarma

(Registration No. 14 of 2008)



School of Management Sciences
Department of Business Administration
Tezpur University

December 2008

Part 1.

1.1 Introduction to the problem

A large *Project* with complex *Resource and Precedence Constraints* gets exponentially complicated for Sequencing and scheduling. In practice, the Project Manager would be willing to 'accept an optimal solution (set)' within manageable time rather than 'wait for the exact solution'.

This need for optimization has provoked widespread study of the problem since the '50s. Operations Research pioneers have christened it as the "*Resource Constrained Project Scheduling Problem*" or the RCPS Problem.

1.4 Objectives of the work

The primary objective of this present work would be

To formulate an algorithm for an optimized solution to the RCPS problem.

To achieve this primary objective, the subsidiary objectives identified through a literature survey are :

- i) To study the approaches made till date, and the paradigm shifts in the approaches, for finding the solution to the RCPS Problem.
- ii) To identify knowledge gap(s) in one of the approaches.
- iii) Development of the proposed algorithm by incorporating novel and feasible concepts.

1.5 Methodology of the work

Background study was carried out in the field of Project Management, which provided insights about lacunae in its practical domain. The niche for this work (The RCPS Problem) was identified while literature review was carried out concerning theoretical and practical application aspects.

Based on the identified gaps, the algorithm improvement was carried out on the selected approach (Genetic Algorithms). This was then implemented through computational experiments. Parameters were set

for tuning the algorithm. The work is based on experimentation for tuning the test parameters, and analysis of the results thereof for accepting the algorithm.

Finally, the result set was compared with internationally accepted test results. This allowed the algorithm to be presented in its final form.

Part 2.

2.1 The RCPS Problem

In the most general form, the RCPS Problem asks the following question: 'Given a set of activities, a set of resources, and a measurement of performance, what is the best way to assign the resources to the activities such that the performance is maximized?'

The underlined segments above are the focus areas of RCPSP.

Part 3.

3.1 Literature Survey (partial, in Tabular form)

<u>Sl No</u>	<u>Year</u>	<u>Author / Researcher</u>	<u>Analysis, Review, Commentary</u>
<u>Exact Methods</u>			
<u>Network Based Approaches</u>			
1	'50s		Formulation of the RCPSP; definitions, etc. Network Methods (PERT and CPM) for small Projects
2	1983	Blazewicz	Dynamic variations into CPM approach.
3	1989	Slowinski and Weglarz	Stochastic variations were constructed into the CPM approach.
4	1990	Neuman	These variations were an attempt to bridge PERT and CPM, and approach reality
<u>Network Techniques offer excellent result for small projects. But fails miserably to manage large projects.</u>			
<u>Operations Research Approaches</u>			
5	1979	Hindelang and Muth	Dynamic Programming formulation for the problem in Decision CPM context.
6	1984	Patterson	LPP approach; gets gradually impractical with the increase in number of tasks and constraints.
7	1985	Davis	"Many of the constraints commonly found in real scheduling problems do not auger well to traditional Operations Research or Mathematical Programming techniques"
8	1997	De, et. al.	Have shown that the Hindelang-Muth procedure is flawed. (Adapted from Demeulemeester, Willy, 2002)

<u>Sl No</u>	<u>Year</u>	<u>Author / Researcher</u>	<u>Analysis, Review, Commentary</u>
9	1999	Khamooshi H	Attempted cross-breeding Dynamic Programming with the Dynamic Priority Scheduling Method (DPSM), which divides a project into phases (cycles), the length of which depends on the duration of the project and the period of clock cycle <u>While such Dynamic Programming approaches can significantly reduce computational effort, the chief concern is with the large amount of space required to store the intermediate results calculated by these algorithms</u>
<u>Enumerative Approaches</u>			
10	1978	Stinson et al	Branch-and-Bound methodology, as applied for
11	1995	Kolisch	Project Scheduling
12	1996	Sprecher & Drexl	
13	1996	Wall, MB	"Enumerative methods cannot solve large problems, the tree is simply too big"
14	2000	Stork, F	Generated Branch-and-Bound algorithms for stochastic RCPS
<u>Branch and Bound methods have limited applicability and success for large projects They require special heuristics to accommodate variations in resource constraint formulations</u>			
<u>Heuristic Methods</u>			
<u>Near-Optimal Heuristic Approaches</u>			
15	1977	Panwalkar and Iskander	A survey of scheduling rules, ranging from simple priority rules to more complex heuristics
16	1978	Garey et al	Upper bounds (assuming a minimizing objective) on the quality of a number of approximate heuristic solutions to RCPSPs
<u>Single Heuristic Approaches</u>			
17	1975	Davis and Patterson	Comparison of some standard heuristics for solving RCSPs, benchmark problems
18	1982	Kurtulus and Davis	Attempt to classify individual RCSPs for the purpose of identifying appropriate scheduling heuristics for their solution
19	late '80s	Alvarez-Valdes R et al	Described a heuristic algorithm based on empirical analysis for the RCPSP
20	1993	Lawrence and Morton	A single-heuristic approach to solving RCSPs that attempts to minimize weighted tardiness through the use of a combination of project-related, activity-related, and resource-related metrics
<u>Multiple Heuristic Approaches</u>			
21	1990	Boctor	Inclusion of certain heuristics can result in the more frequent development of near-optimal, and occasionally optimal, schedules
22	1994	Hildum, D W	Larger combinations of heuristics leads to increased ability to produce better quality schedules <u>For a relatively large project, with multiple constraints, heuristics provides better and feasible solution set</u>
<u>Artificial Intelligence Approaches</u>			
23	1994	Hildum	Grouped artificial intelligence (AI) approach to scheduling as either expert systems or knowledge-based
24	Late '90s till	Hartmann S and Kollisch R	AI Techniques for RCPS, also publishes comparative statements and benchmark changes of major works carried out for the RCPSP, and its different

<u>Sl No</u>	<u>Year</u>	<u>Author / Researcher</u>	<u>Analysis, Review, Commentary</u>
	date		versions.
<u>Simulated Annealing</u>			
25	1990	Fayer	Initial experimentation of using SA for scheduling problems
26	1993	Boctor	Reported fairly good performance by a simulated annealing approach on the Patterson problems
27	2000	Zbigniew and David	Accepts a better neighbor solution, but rejects any deterioration
28	2004	Smith	Developed variations for moving away from local optima
<u>Tabu Search</u>			
29	1989	Glover and Greenberg	Pioneer developers of Tabu Search (TS) methodology.
30	1994	Pinson et al	Tabu Search used for scheduling algorithms.
31	1997	Glover and Laguna	
32	2007	Glover, F	Further avenues of use of TS for use in Scheduling for Engineering applications
<u>Fuzzy System</u>			
33	2003	Hongqi Pan Chung-Hsing Yeh	Fuzzy RCPS metaheuristic approach
<u>Genetic Algorithm</u>			
34	1975	Holland H.J.	Pioneer developer of Genetic Algorithms methodology
35		Altus et al	Attempts to apply Genetic Algorithms to process resequencing
36		Rogers J.L.	
37	1996	Wall, M.B.	Applied GA to 10-300 activities, 3-10 resource PSPs, but parameter negotiations were not carried out (PhD work)
38	1997	Sharma, et al	Different GA operators (crossovers, mutation, cloning, skimming, etc)
39	2005	Zwikael, et al	Application approaches for Non-Delay scheduling
40	2005	Mendez, et al	Random key based GA, with an extensive study of the basic formula
41	2005	Kolisch, Hartmann	Update to a previous survey on different heuristic approaches. Concluded that GA (and TS) still features as the most preferred metaheuristic technique
42	2008	Petegham, V.V. and Vanhoucke	Applied bi-population GA for a modified RCPSP
<u>Multiple Technique application</u>			
43	1995	Rabelo L. et al	Attempted a hybrid approach for real time sequencing and scheduling problems by using Neural Networks, Genetic Algorithm, Simulation and Machine Learning
44	2006	Kim, J.L.	Designed an adaptive hybrid Genetic Algorithm by combining Simple Genetic Algorithm (SGA) for global search with Random Walk Algorithm (RWA) for local search, and arrived at a good result for the RCPSP.
<p><u>Kolisch and Hartmann (2005) concluded that GA (and TS) still features as the most preferred metaheuristic technique for approaching optimality of the RCPSP.</u></p>			

<u>Sl No</u>	<u>Year</u>	<u>Author / Researcher</u>	<u>Analysis, Review, Commentary</u>
<u>Dynamic Scheduling, Disruption Management</u>			
44	2002	Kocjan, W	Initiated a report on Dynamic Scheduling, by attempting to break 'static rules'.
45	2007	Kuster, et al.	Teaching the RCPS to proceed with alternative paths, in case of disruption.
<u>This is one of the contemporary focuses, and is still in a nascent stage</u>			

3.2 Gap Identification

There is an ever-expanding list of optimization methods with ample opportunity for modification and adaptation, to approach the RCPS Problem. Because of its relatively superior application potential for RCPS, Genetic Algorithm (GA) has been selected for the present work in an attempt to optimize the Problem.

Genetic algorithm has been used extensively for the RCPS Problem. However bulk of the focus was on the operators (mating and diversity). There has been less significant effort spared for the triggers, viz. the project selection parameter and the termination parameter. The present work would adapt accepted robust versions of the operators and attempt to expand the knowledge domain specifically of these two triggers.

Part 4.

4.1 Genetic Algorithms

Genetic Algorithms (GA) is adaptation of Nature's '*survival of the fittest*' dictum. Here problems are solved by an evolutionary process resulting in a best (fittest) solution (survivor). They are less susceptible to becoming 'stuck' at local optima compared with some other types of optimization techniques.

Part 5

5.1 The Proposed Algorithm

The major components of the proposed algorithm is described in sequence of GA is described in the following table

The Major Components of the Proposed Algorithm

Stage	Work description (<i>italicized portions marked with a '*' indicate novel contribution</i>)
Generate Initial Population	Collect Project information
	Calculate the normal, unconstrained MakeSpan - the length of the project
	Generate schedules, till the initial population is filled up
	Allocate a <i>Unique Number*</i> to each schedule, based on <i>Sequence and Constrained MakeSpan</i> , to indicate its 'fitness for survival'
	The 'best' solution set is cloned out as elites (Solution set)
Produce the Next Generation	<i>Depending on the 'fitness strength' of the parents, clones* are (virtually) made within the population</i>
	<i>From the (parent) population, two individuals are selected for mating, by a 'better spouse'* adaptation of the 'tournament' methodology</i>
	The two parents mate to produce two offspring, by the Precedence Set Crossover (binary) operator, which is a robust one for Project Scheduling
	Introduce diversity by using the Immigration (unary) operator
	The population schedules are allocated their Unique Number, and they are transferred into the Next Generation after cloning out the 'best' solution set. New generation members are retained by (adaptation of) the Struggle GA methodology
Terminating criteria	Two distinct criteria were tested - a) Project parameters dependent criterion, and b) Project complexity dependent (Adaptive) criterion A formula has been developed for the 'adaptive criterion'* that incorporates complexity level of the Project

Part 6.

6.1 Implementation of the Algorithm

The programming has been done in C and compiled using Borland C++ compiler.

6.2 Comparative Tests for tuning the Algorithm

For testing robustness, experimental tests were carried out on (selected sets of) internationally recognized benchmark instances for the evaluation of solution procedures for the RCPSP, provided at the PSPLIB. The last update of the Library was noted as on 2nd May, 2008.

Part 7.

7.1 Experimental results and Analysis

For the RCPSP, the 'fittest' solution for a Project would be the schedule with the lowest MakeSpan. The PSPLIB has provided the 'best' solution set of the benchmark instances. There are published research results on these benchmark instances. These were studied vis-à-vis the experimental outcomes of the Proposed Algorithm - how good it is by comparing the results with the benchmark results. The deviations and analysis thereof, provides the necessary information for tuning the algorithm for acceptance.

Part 8.

8.1 Conclusion

The present work proposes few acceptable concepts for the selection and termination triggers of GA application for approaching optimality of the RCPS. The experimental results are comparable to the best amongst the published result sets. Finally a few directions for further study are hinted at, along with indication of possible practical application of the proposed model.



TEZPUR UNIVERSITY

Certificate of the Supervisor

This is to certify that the thesis entitled Formulation of an Optimized Algorithm for Resource Scheduling and Allocation in Projects : A Genetic Algorithms Approach submitted to the School of Management, Tezpur University in part fulfillment for the award of the degree of Doctor of Philosophy in Management is a record of research work carried out by **Mr. Tridib Ranjan Sarma** under my supervision and guidance.

All help received by him/her from various sources have been duly acknowledged.

No part of this thesis has been submitted elsewhere for award of any other degree.

Supervisor :

Dr. G. Singaiah, M.Com, MBA, PhD

(Formerly)
Reader,
Department of Business Administration,
Tezpur University,
Assam, India

Professor and Head,
Department of Management
North-Eastern Hill University
Tura Campus,
Meghalaya, India

Acknowledgements

It was with utmost trepidation that the present work was embarked upon. But all along the way, words of supporting wisdom and persistent assurance of success from different quarters helped the final outcome. I take this opportunity to remember with gratitude.

- ✓ Prof. G. Singaah, my Supervisor, for his untiring support and advice. And more significantly, for coming to help me to move ahead with this work – at a very critical phase
- ✓ Members of my Research Committee, Dr.(Mis.) C. Goswami and Dr. M.K.Sarma for providing articulate insights into vital research issues, especially for methodology and analysis phase
- ✓ My colleagues at Tezpur University in general, and the Department of Business Administration in particular for supporting with their academic experience and technical expertise
- ✓ (Late) Prof. M.C Bora, to whom I offer my reverent gratitude for goading me into taking up this work – but whose untimely demise affected the progress in more than one way.
- ✓ My very special family members, who epitomizes the zenith of help and hindrance in their own ways that I shall always remember. I have a very special thanks to the apple of my eye, Shivangi (Mis2), for accepting me as I was during the period.
- ✓ Authors and Researchers, whose academic competency was my inspiration all throughout. I take this moment to recollect them with gratitude whose work I have referred, consulted and quoted. At no point of time do I have the courage to claim originality.
- ✓ All who would pick up this thesis, and hopefully, provide me with their valuable feedback that would be inspiration to move forward – I thank thee in advance

I once again express thanks to all the helping hands and thinking brains who helped me from those days till today.

Tridib Ranjan Sarma
Tezpur / Assam / India

31st of December, 2008

TABLE OF CONTENT

Abstract	ii - viii
Acknowledgements	x
List of Tables	xviii - xix
List of Figures	xx - xxi
<u>Chapters</u>	1 - 183
Chapter One : Introduction And Background	1 - 18
	Prologue 2
1.1 Introduction	3
1.2 Project (Mis)Management In Developing Countries	4
1.3 The Project Manager's Dilemma	6
1.4 Project Management Knowledge Areas	7
1.5 Project Monitoring Systems	11
1.6 Resource Allocation Problem	12
1.7 Boundary Specifications Of The Present Work	13
1.7.1 Title of the work	13
1.7.2 Objective of the work	13
1.7.3 Scope of the work	14
1.8 Methodology of the work	15
1.9 Content Outlines of this Thesis	17
Chapter Two : The Problem	19 - 27
2.1 Introduction	19
2.2 The Resource-Constrained Project Scheduling Problem	20
2.3 General Formulation of the Problem	21
2.4 Definition of Terminology	23

Chapter Three : Related Work	28- 51
3.1	Introduction 29
3.2	Solution Approaches 29
3.2.1	Exact methods 29
3.2.1.1	Network Based Approaches 30
3.2.1.2	Operations Research Approaches 31
3.2.1.3	Enumerative Approaches 33
3.2.2	Near-Optimal Heuristic Approaches 34
3.2.2.1	Single Heuristic Approaches 38
3.2.2.2	Multiple Heuristic Approaches 39
3.2.3	Artificial Intelligence Approach 40
3.2.4	Simulated Annealing 40
3.2.5	Tabu Search 41
3.2.6	Ant Colony Optimization 42
3.2.7	Fuzzy System 43
3.2.8	Genetic Algorithm 43
3.3	Additional Works 44
3.4	Three Candidate Approaches 45
3.4.1	Tabu Search 45
3.4.1.1	Advantages of Tabu Search 46
3.4.1.2	Disadvantages of Tabu Search 46
3.4.2	Simulated annealing 47
3.4.2.1	Advantages of Simulated Annealing 47
3.4.2.2	Disadvantages of Simulated Annealing 48
3.4.3	Genetic Algorithms 48
3.4.3.1	Advantages of Genetic Algorithm: 50
3.4.3.2	Disadvantages of Genetic Algorithm: 50
3.5	Choice of Approach 51
3.6	Test Data Set and Benchmark Results 51

Chapter Four : The Approach: Genetic Algorithm	52 – 84	
4.1	Introduction	53
4.1.1	History and Evolution	53
4.1.2	Application Area	54
4.1.3	Mechanism	55
4.2	Types of Genetic Algorithms	57
4.2.1	Simple Genetic Algorithm (Non-Overlapping Populations)	58
4.2	Steady-State Genetic Algorithm (Overlapping Populations)	58
4.2.3	Struggle Genetic Algorithm	58
4.2.4	Specially devised Genetic Algorithms	59
4.3	Features of Genetic Algorithms	60
4.3.1	Chromosome Representations	61
4.3.2	Initial Population	63
4.3.3	Selection Operator	63
4.3.3.1	Fitness Proportionate Selection	64
4.3.3.2	Tournament selection	65
4.3.3.3	Other selection methods	65
4.3.4	Offspring Generation Operator	66
4.3.4.1	Binary Reproduction Operator : Crossover	66
4.3.4.1.1	One Point Crossover	68
4.3.4.1.2	Two Point Crossover	68
4.3.4.1.3	Uniform Crossover	69
4.3.4.1.4	Arithmetic Crossover	70
4.3.4.1.5	Heuristic Crossover	71
4.3.4.1.6	Cut and Splice Crossover	71
4.3.4.1.7	Half Uniform Crossover	72
4.3.4.1.8	Crossover for Ordered Chromosomes	72
4.3.4.2	Unary Reproduction Operators	73
4.3.4.2.1	Mutation	73
4.3.4.2.2	Immigration	76

4.3.5	Fitness Function:	77
4.3.6	Termination Criteria	79
4.3.6.1	Generation Number	79
4.3.6.2	Evolution Budget	80
4.3.6.3	Fitness Threshold	80
4.3.6.4	Fitness Convergence	80
4.3.6.5	Population Convergence	80
4.3.6.6	Gene Convergence	81
4.3.6.7	Manual Inspection	81
4.4	Elitism	81
4.5	Building Blocks Hypothesis	82
4.6	Knowledge Domain	84
Chapter Five : The Proposed Algorithm		85 - 113
5.1	Introduction	86
5.2	Encoding and representation of solution	87
5.2.1	Basic Representation	87
5.2.2	Final Representation	89
5.3	Initial Population	90
5.4	Fitness Function	91
5.5	Elites, or 'Solution Set'	97
5.6	Population size	97
5.7	Clones	98
5.8	Selection	99
5.9	Crossover	100
5.9.1	The Precedence-Set Crossover	100
5.9.2	Crossover Points	102
5.9.3	Building Blocks	103
5.1	Intrusions	103
5.10.1	Immigration	103

5.10.2	Dormant-Forefather	105
5.11	Termination	107
5.11.1	Fixed Generations Termination	107
5.11.2	Fitness-Deviation Termination	108
5.11.3	Adaptive Termination	108
5.11.3.1	Adaptive Termination 1	110
5.11.3.2	Adaptive Termination 2	111
5.11.3.3	Adaptive Termination 3	112
5.12	Post Termination	113
Chapter Six : Implementation and Experimental Setup		114 - 147
6.1	A General Outline	115
6.1.1	Platform Description (hardware, software, etc)	115
6.1.2	Input Information – The Test Data-Set	115
6.1.2.1	The Input Data-set	115
6.1.2.2	Other Test Data-set	116
6.1.2.3	Modifications for changing to other forms	119
6.2	Encoding and Representation of Chromosome	120
6.3	Initial Population	121
6.3.1	Sequence Generation	121
6.3.2	Unconstrained makespan	122
6.4	Schedule Generation Scheme	123
6.5	Fitness Function	124
6.6	Elites, or ‘Solution Set’	126
6.7	Population size	127
6.8	Clones	127
6.9	Selection	128
6.1	Crossover	129
6.10.1	Crossover point	130
6.10.2	Mating schemes and number of offspring	130

6.11	Next Generation	131
6.12	Intrusions	132
6.12.1	Immigration	132
6.12.2	Dormant-Forefather	133
6.13	Termination	134
6.13.1	Fixed Generation Termination	134
6.13.2	Fitness-Deviation Termination	135
6.13.3	Adaptive Termination	136
6.14	Design of Experiment	139
6.14.1	The Design Matrix	140
6.14.2	The Experimentation Plan	140
6.15	Methodology for Result Analysis	145
 Chapter Seven : Analysis of Experimental Results		148 - 175
7.1	Introduction	149
7.2	Program Feasibility and Conformance	149
7.2.1	Program feasibility	149
7.2.2	Conformance to Benchmark Literature	153
7.3	Validation of Parameters and Operator Modes	155
7.3.1	Clones Factor and Better-Spouse Selection	155
7.2.2	Crossover Mode	157
7.2.3	Intrusion Mechanisms	158
7.2.4	Termination	159
7.2.4.1	Fixed Generation Termination	159
7.2.4.2	Fitness-Deviation Termination	160
7.2.4.3	Adaptive Termination	162
7.2.4.3.1	Adaptive Termination 1	162
7.2.4.3.2	Adaptive Termination 2	163
7.2.4.3.3	Adaptive Termination 3	167
7.3	Comparison with Published Results	172

7.4	Presentation of the Algorithm	174
Chapter Eight : Conclusions		176 - 187
8.1	Summary of Research Issues	177
8.2	Strength and Shortcomings of the Algorithm	179
8.3	Contribution to Knowledge Base	180
8.3.1	Sweep-Creep SGS	180
8.3.2	UnoSign based Fitness Value	180
8.3.3	Better-Spouse Selection	181
8.3.4	Adaptive Termination, AdapTerm3	181
8.4	Directions for Further Research	182
8.4.1	The Macro Perspective	182
8.4.2	The Micro Perspective	183
8.4.3	The Approach	184
8.5	Practical Utilization of BATGA	185
	Epilogue	187
<u>Appendices</u>		188 - 211
	Appendix I : Glossary	188
	Appendix II : Published Paper	191
	Appendix III : Extension of BATGA	209
<u>References and Bibliography</u>		212 - 234
	Digital Source References (major)	213
	Literature References and Bibliography	214

LIST OF TABLES

1.1	Project Time Management Overview	8
1.2	Project Cost Management Overview	9
4.1	Comparison Of Scaling Methods	79
6.1	Example Of Schedules, J30 Data-Set	124
6.2	First Stage Of Calculating <i>UnoSign</i> .	125
6.3	Third Stage Of Calculating <i>UnoSign</i> .	126
6.4	Calculation Of <i>Copy</i> (Permitted Clones)	127
6.5	Schedules List Sorted On <i>UnoSign</i>	128
6.6	Characteristics Of Crossover Utilized.	131
6.7.	The Design Matrix	141
6.8.	The Experimentation Plan	143
7.1	Experimental Results	150
7.2	The Initial Test Result Set vis-v-vis Benchmark Results	153
7.3	Experimental Results: Clones Factor	155
7.4	Experimental Results For <i>Better-Spouse</i> Selection	156

7.5	Performance Comparison Between BS And RR Selection	157
7.6	Comparison Of Crossover Mode	158
7.7	Performance Comparison Of Intrusion Techniques	158
7.8	Deviation Limits For Fitness-Deviation Termination	160
7.9	Experimental Results of Termination By Fitness-Deviation Criteria	161
7.10	Experimental Results of Adaptive Termination 1	163
7.11	Experimental Results of Adaptive Termination 2	165
7.12	Experimental Results of Adaptive Termination 3	169
7.13 (a)	Comparison with benchmark results : J30	172
7.13 (b)	Comparison with benchmark results : J60	173
8.1	Gap between literature result and commercial implementation (2001)	186

LIST OF FIGURES

2.1	A Project Network (Activity on Node)	26
4.1	Process of Genetic Algorithm	56
4.2	Representation of a Chromosome	62
4.3	One Point Crossover	68
4.4	Two Point Crossover	69
4.5	Uniform Crossover	69
4.6	Arithmetic Crossover	70
4.7	Cut and Splice Crossover	72
5.1	Basic Representation of the Chromosome	88
5.2	Example of a Basic Representation of the Chromosome	89
5.3	Final Representation of the Chromosome	89
5.4	A Project Network (Activity on Node)	100
5.5	The PSX1 Mechanism	101
5.6	Concept of the Dormant-Forefather (<i>Egyptian Mummy</i>)	106
5.6	The Damped-Acceleration Mechanism	112

6.1	PSPLIB Instance Input Format	118
6.2	Instance example of the J30 test data-set ; File name J301_1.rcp	119
6.3	The Algorithm for Sequence Generation	122
6.4	The <i>Sweep-Creep</i> Algorithm	123
7.1(a)	Summary of deviations, J30, Experiment ESN A2a2	154
7.1(b)	Summary of deviations, J60, Experiment ESN A2b2	154
7.2	Performance comparison between initial set and current (best) set of parameters and modes	160
7.3 (a)	Discrete <i>MaxGen</i> as outcome <i>MaxSdl</i> , Adaptive Termination 2 (Data-set J30, sorted on <i>MaxSdl</i>)	164
7.3 (b)	Discrete <i>MaxGen</i> as outcome <i>MaxSdl</i> , Adaptive Termination 2 (Data-set J60, sorted on <i>MaxSdl</i>)	164
7.4	Efficiency Index Vs Project Complexity, Adaptive Termination 2 (Data-set J30, sorted on <i>MaxSdl</i>)	166
7.5	Performance Changes, Adaptive Termination 2 (Data-set J30)	167
7.6 (a)	Discrete MaxGen as outcome MaxSdl, Adaptive Termination 3 (Data-set J30, sorted on <i>MaxSdl</i>)	168
7.6 (b)	Discrete MaxGen as outcome MaxSdl, Adaptive Termination 3 (Data-set J60, sorted on <i>MaxSdl</i>)	168
7.7	Efficiency Index Vs Project Complexity, Adaptive Termination 3 (Data-set J30, sorted on <i>MaxSdl</i>)	170
7.8 (a)	Summary of deviations, J30, ESN B4a12	171
7.8 (b)	Summary of deviations, J60, ESN B4b12	171

Chapter One

Introduction and Background

In this Chapter we provide background for a well-established problem. Starting from a macro-angle, this Chapter finally focuses on the niche in the knowledge domain of Project Management that features our chosen problem, which is known as the RCPSP. The rationale and background information for the work is discussed here. We specify the objective(s) of our work and its scope as boundary specifications. This is followed by outline of the methodology. Finally we present brief outline of all Chapters and other content of this thesis.

“

Laws of Project Management

1. *No major project is ever installed on time, within budget, or with the same staff that started it Yours will not be the first.*
2. *Project progress quickly until they become 90% complete, then they remain at 90% complete forever.*
3. *One advantage of fuzzy project objectives is that they let you avoid the embarrassment of estimating the corresponding costs.*
4. *When things are going well, something will go wrong.*
 - *When things just cannot get any worse, they will.*
 - *When things appear to be going better, you have overlooked something.*
5. *If project content is allowed to change freely, the rate of change will exceed the rate of progress.*
6. *No system is ever completely debugged. Attempts to debug a system inevitably introduce new bugs that are even harder to find*
7. *A carelessly planned project will take three times longer to complete than expected; a carefully planned project will take only twice as long.*
8. *Project teams detest progress reporting because it vividly manifests their lack of progress*

”

(As outlined by

American Production and Inventory Control Society (APICS : The Association for Operations Management)

in an attempt to explain the consequences of uncertainty on Project Management.)

Extracted from

Project Management Engineering, Technology, and Implementation

Shtub,A, Bard,J F, Globerson, S

Prentice Hall, Englewood Cliffs, NJ 07632

1994, Page 8

1.1 Introduction

Businesses, industries, organizations and nations of every size and focus count on professionally managed Project Management skills to make them succeed in the ever more competitive global marketplace. Project Management is an area that has a direct bearing on the National as well as International scenario for sustaining the economic as well as Industrial growth of the present day. The importance of this area can be gauged from the fact many countries has bodies to monitor infrastructure works of importance. Both Government as well as the Industry establishes such bodies. India has a full-fledged Union Ministry with the name of Ministry of Statistics and Project Implementation (MOSPI) to monitor macro projects. Amongst other activities, this body regularly publishes statement(s) regarding the status of progress of projects undertaken by the Union Government. On the international front, the leading body is the UNIDO. The professionals coming under the purview of this field have their own body with the self-explaining name Project Management Institute (PMI), whose headquarters is in Pennsylvania (USA), with Chapters all over the world.

Because of its visible impact, the field of Project Management has shown an extraordinary growth internationally, especially in the developed nations. Individuals skilled in the field of Engineering and Management are gradually gravitating towards the development and management of Projects. Project Management tools and techniques, and the related Information System - developed especially for especially for Engineering Projects, forms an integral part of modern Project Management.

1.2 Project (Mis)Management In Developing Countries

Nevertheless, in developing and under-developed nations, the importance attached to the area of Project Management, especially by the professionals and powers to be, seems to be relatively low. This is especially true in case of Government and Semi-Government funded projects. In India such projects are generally termed as Public Sector Unit (PSU) projects. More often than not, even a small-scale project fails to be completed within the budgeted Time/Cost frame. The resultant is massive cost overrun – running into billions of taxpayer money. High time overruns – running into years, again in turn is translated into cost overruns.

To stay on top, new projects and business development must be completed quickly, in time and within cost budget. Failure on any of these fronts would result in massive overruns of the two most important resources - time and cost. This gets negatively reflected on the business of a firm, position of the related industry and the economy of the nation as a whole.

Since Independence (1947) till about the turn of century, India has lost over US\$ 15 billion due to cost and time overruns in executing major and mega projects in the Public Sector, according to a report based on official statistics collated and released by the Government as well as FICCI, an apex body of business houses. The major contributors to this dubious distinction are the power, railways and steel sectors – combined they accounts for around three-fourths of this overrun. Citing specific examples of Indian context, even in short would produce volumes, and is left out of the scope of this thesis.

At this juncture, it would be pertinent to state a paradoxical fact that this data on cost overrun doesn't fully capture the extent of the problem. Paradoxical in the sense that there are certain sectors where the time overrun doesn't get converted to cost overruns. In areas like power and petrochemicals, equipment prices have declined dramatically over the years. So we land up in a peculiar situation where we

have time overrun without cost overruns. In such a situation we have to fall back on alternative course of action for converting the time overrun to cost parameters. One convenient and plausible way is to use the concept of Opportunity cost in terms of the profits foregone and extra costs in other ways. For example, take the Indian Oil Corporation's Panipat Refinery. It had a time overrun of 14 months (at the time of the said report), but no cost overruns, with costs frozen at around US\$ 1.2 billion (including US\$ 0.5 billion for an associated pipeline project). Yet this delay has meant that the country had to import more petrochemical products for that period. Also, since IOC would have been entitled to a 12% post tax return for the immediate period, when the APM dismantling began, the delay meant that IOC actually had lost out on an additional profit of around US\$ 0.15 billion, or around 15% of its total profits for the year.

Faulty planning will result in project failure, whereas high-quality project planning increases the project's chances of success. Zwikael and Globerson (2004) reports on development and implementation of a model (PMPQ) aimed at evaluating the quality of project planning. The use of the PMPQ model across industries enabled the authors to conclude that Construction and Engineering companies have the highest level, while Production and Maintenance companies have the lowest quality of project planning.

Nevertheless, faulty planning and mismanagement at certain quarters drives the cost of a Project to escalate much beyond budgetary limits or acceptable temporal variation. Fault into a Project may creep in at design level, control level or at execution level. But the most visible amongst these is execution level – level at which the Project Manager operates.

1.3 The Project Manager's Dilemma

The group, which had originally formulated or designed the Project, does in most cases lay down schedules that are too far away from achievable reality. In most cases the person who would finally control the execution of the plans – the Project Manager – is not made a party of the planning phase. Even if he is made so, during implementation phases, situation crops up which would call for changes in the plans – either subtle or drastic. Couple that with conflicting resource allocation and inadequate monitoring, and one can very well imagine the resulting chaos that the Project Manager faces. The Project Manager faces the unenviable dilemma of monitoring the Project for judicious allocation resources in face of multiple constraints.

A project is an open system, fully interacting with the environment – both for input(s) as well as output(s). The control and feedback mechanism installed for aiding the Project Manager is the only tool that attempts to keep the project within track to proceed towards its logical and physical conclusion, with minimal negative impact. At the macro level, such impacts are very much visible, and are open to criticisms.

The impact on many fronts can be gauged, controlled and remedied to a great extent at the micro level. For identifying the points where these checks can be incorporated, one needs to go into more details into the relevant portion within the field of Project Management.

1.4 Project Management Knowledge Areas

PMI have identified nine areas of Project Management Knowledge. *(It is worthwhile to note here that PMI discourages use of the term 'functions' in this context, as the term 'function' has been frequently misunderstood to mean an element of a functional organization.)* The nine Project Management Knowledge Areas as outlined in the PMBOK (2004) are :

- i. Project Integration Management,
- ii. Project Scope Management
- iii. Project Time Management
- iv. Project Cost Management
- v. Project Quality Management
- vi. Project Human Resource Management
- vii. Project Communication Management
- viii. Project Risk Management
- ix. Project Procurement Management

Within the scope of this work, we look further into the 3rd and 4th items of the above list, i.e. into Project Time Management and Project Cost Management.

Project Time Management knowledge area identifies five major processes :

1. Activity Definition – identifying the specific activities that must be performed to produce the various project deliverables,
2. Activity sequencing – identifying and documenting interactivity dependencies,
3. Activity duration estimation – estimating the number of work periods which will be needed to complete individual activities,
4. Schedule development – analyzing activity sequences, activity durations, and resource requirements to create the project schedule, and,
5. Schedule control – controlling changes to the project schedule.

In Table 1.1 we specify micro-level position within Project Time Management Knowledge area, understanding and controlling of which ultimately controls the macro impacts

Major Processes	Inputs	Tools and Techniques	Outputs
Activity Definition	Work Breakdown Structures Scope Statement Historical Information Constraints Assumptions	Decomposition Templates	Activity list Supporting details Work Breakdown Structure updates
Activity Sequencing	Activity list Product Description Mandatory dependencies Discretionary dependencies External dependencies Constraints Assumptions	Precedence diagramming method Arrow diagramming method Conditional diagramming method Network templates	Project network diagram Activity list updates
Activity Duration Estimating	Activity list Constraints Assumptions Resource requirements Resource capabilities Historical information	Expert judgment Analogous estimation Simulation	Activity duration estimates Basis of estimates Activity list updates
Schedule Development	Project network diagram Activity duration estimates Resource requirements Resource pool description Calendars Constraints Assumptions Leads and lags	Mathematical analysis Duration compression Simulation Resource leveling heuristics Project management software	Project schedule Supporting details Schedule management plan Resource requirement updates
Schedule Control	Project schedule Performance reports Change requests Schedule management plan	Schedule change control system Performance measurement Additional planning Project management software	Schedule updates Corrective action Lessons learned

Table 1.1 : Project Time Management Overview

(Adapted from PMBOK)

PMBOK (2004) has identified micro level components of Project Cost Management, and pinpoint control areas under scope. It includes processes required to ensure that the project is completed within approved budget, and involves 4 major processes, as depicted in Table 1.2

1. Resource Planning – determining what resources and what quantities of each should be used to perform the project activities,
2. Cost Estimation – developing an approximation of the costs of the resources needed to complete the project activities,
3. Cost Budgeting - Allocating overall cost estimate to individual work items, and,
4. Cost control – controlling changes to the project budget.

Major Processes	Inputs	Tools and Techniques	Outputs
Resource Planning	Work breakdown structure Historical information Scope statement Resource pool description Organizational policies	Expert judgment Alternatives identification	Resource requirements
Cost Estimating	Work breakdown structure Resource requirements Resource rate Activity duration estimate Historical information Chart of accounts	Analogous estimating Parametric modeling Bottom-up estimating Computerized tool	Cost estimates Support detail Cost management plan
Cost Budgeting	Cost estimates Work breakdown structure Project schedule	Cost estimating tools and techniques	Cost baseline
Cost Control	Cost baseline Performance reports Change requests Cost management plan	Cost change control system Performance measurement Additional planning Computerized tools	Revised cost estimates Budget updates Corrective actions Estimate at completion Lessons learned

Table 1.2 : Project Cost Management Overview

(Adapted from . PMBOK)

Project cost management is primarily concerned with the cost of the resources needed to complete project activities. From a business and economic point of view, it is always the cost figures that are displayed and makes all the difference. It may be noted that any overrun on the Time factor would invariably get reflected on the Cost factor due to obvious reasons. Thus it is imperative that these two are the prime factors to be controlled.

The most important tool in the hand of the Project Manager for this impact control and management is the Project Monitoring System. This has to operate by generating professionally laid down project schedules and correct resource allocation for the scheduled jobs. A plan is never static, especially for Projects. As soon as a plan is finalized, something endogenous and/or exogenous mandates a change in it. This instability goes on to upset the subsequent stages, most perceptibly the resource allocations. It thus becomes imperative that the resources, which are finite in numbers and quantity, be reallocated. For a relatively small project, manual reshuffling might be possible. But for major projects, this invariably demands fast and intelligent computer software, with necessary heuristics built in. (*The RCPS problem – to be discussed later – is a special case of resource leveling where the heuristic involved is a limitation on the quantity of resource available.*) The software incorporating this set of heuristics would be as fast as the algorithm it follows, *ceteris paribus*.

Based on the outcomes, the Project activities may be resequenced or rescheduled. These are invariably inter related, and demands maximum efficiency from the Project Manager. His only ally in this battle against Time-Cost overrun is the Computer, and (fast algorithm driven) Project Management software that forms the Project Monitoring System, which earlier was referred to as the control and feedback mechanism. Amongst the various facets presented in Table 1.1 and Table 1.2, we are referring to this very small but most important niche.

1.5 Project Monitoring Systems

The most important component of Project Monitoring System is the Project Management Information System (PMIS). The periodicity of reference and reporting by Project Monitoring System is a crucial factor, and aims at constant checking and monitoring of the present status of the project. Additionally it can be suitably upgraded to predict the future parameters of the project based on historical and other factors. This can be based on any planning technique used in the enterprise. Measurement, evaluation, trouble-shooting and improvement of performance are the prime objectives of the PMIS. Improvement comes through decision-making, which is based on information.

For the perspective of our study, Project Management Information System generally consists of three modules –

PMIS/T :: Time Management Information;

PMIS/C :: Cost Management Information; and

PMIS/R :: Resources Management Information.

In addition to these, the Project Monitoring System would consist of a module to handle information on quality and quality standards. Similarly there would be a module to provide information on combined exception report on the total performance.

The system works on certain algorithms, which are based on pertinent assumptions, rules and constraints. To improve on the working and performance of the software, the algorithms needs to be studied, and newer and faster technological innovations needs to be incorporated. In a comparative analysis of commercial Project Management software, Mellentien and Trautmann (2001) noted that there is still a significant performance gap between algorithms implemented and those available on research literature. They opined that closing this gap constitutes a challenge for future research.

1.6. Resource Allocation Problem

Progress and success of projects depend on timely availability of resources. But more often than not, these are limited in numbers and scarce to get. In many cases delay occurs in project completion due to the non-availability of the same resource for possible parallel jobs. The problem created due to inadequate resource allocation spills over to create a time-overrun problem, which in turn creates problem in the cost management aspect. Procuring additional resource could mitigate this problem. But it is a rule rather than exception that additional procurement generally fails to be justified from the economical point of view. Therefore there is a paradoxical relationship between resources requirement and availability, and their allocation.

To study and analyze this situation, the problem has been brought into focus worldwide by the name of Resource-Constrained Project Scheduling Problem, or RCPS Problem (RCPS). In an ongoing process, various tools and techniques have tried to approach optimization of this problem. The present work attempts to contribute to the knowledge domain of a specific area pertaining to an approach (*Genetic Algorithm*) for optimization of the RCPS.

Having placed this background information about the field of study, and gradually pinpointed deliberation on the specific problem, we now proceed to demarcate the boundary specifications of our work.

Subsequently, an outline of methodology for the work is presented, followed by content description of this thesis.

1.7 Boundary Specifications Of The Present Work

Within the total knowledge domain of Project Management and related Business Informatics, our work concentrates on the distinctively identified niche. We restrain ourselves within the identified spot to maximize focus and intensity by specifically defining the title, objectives and scope of the present work.

1.7.1 Title of the work

The present work was triggered by an inquisitive study into different aspects of Project Management, and associated lacunae. During the process, it gradually focused into a specific problem, optimization of which would benefit the Project Manager.

For formulating the optimization model, we sought for a viable approach, and finally zeroed onto the Genetic Algorithms approach.

With the intention of highlighting these two issues, the title of the study is composed as

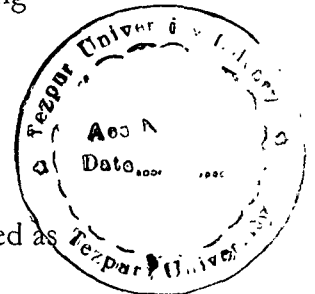
“Formulation of an Optimized Algorithm for Resource Scheduling and Allocation in Projects: A *Genetic Algorithms* Approach”

1.7.2 Objective of the work

Based on a background study, the primary objective of this work is outlined as

“To formulate an algorithm for approaching an optimized solution to the RCPS Problem towards minimization of the makespan.”

The primary objective would be achieved by convergence of three subsidiary objectives, which are identified through a literature survey:



- i) To study the approaches made till date, and the paradigm shifts in the approaches, for finding the solution to the RCPS Problem
- ii) To identify knowledge gap(s) in one of the approaches.
- iii) Development of the proposed algorithm by incorporating novel and feasible concept(s).

Subsequent Chapters of this thesis attempts to deliver pertinent portions of each of the subsidiary objectives, the whole of which assemble to address the primary objective.

1.7.3 Scope of the work

The scope of work is kept sharp on three aspects – the focus, the approach and the extent.

Focus of work is specifically on a Business Informatics area of Project Management, related to resource allocation for Project activity scheduling. The work proceeds with the assumption that Project Manager will be dependent on an ever more efficient PMIS. There are related possibilities of study within this niche with focus on cost optimization, labour optimization, risk management, etc. However those aspects are by themselves research areas involving extensive study, and are kept beyond the scope.

With an ever-expanding list of optimization approaches (tools and techniques) RCPSP has been studied from various perspectives. In this work we concentrate our attention on one school of approach, which allowed in-depth analysis. At this point it is acknowledged with humbleness that a number of novel and robust (combinations of) techniques have been propounded since we began our study. But as our work had already proceeded in the chosen path, we considered it prudent not to divert.

The extent of work is kept within theoretical boundary. The algorithm as designed in this work is tested using internationally recognized test data-set, and weighed against benchmark result thereof. Field-testing and/or empirical validation of the algorithm using real data-set is not carried out. Neither have we attempted mathematical validation of the few relationships developed for use in the algorithm, leaving that to our competent brethren.

Moreover, the extent of the present work remains within study on ‘single project’, thereby concentrating on the RCPS Problem. Resource scheduling for ‘multiple projects’ is studied under the RCMPS Problem – with a different set of test data-set, which we shall keep outside our scope.

1.8 Methodology of the work

Amongst research paradigms, the methodology utilized for present work is one normally used universally for algorithm design, which follows a general pathway of Design – Development – Validation.

For the Design and Development segment, we loosely follow stages of SAD, where the problem is understood first. This is followed by a study and presentation of alternatives for solution to the problem. Once an acceptable alternative is identified, it is analyzed in details for adaptation to develop the solution model. The model is finally subjected to testing and modifications for Validation of usage. Once validated, the model is ready to be presented as solution of the problem.

As the first stage, the RCPS Problem and approaches made for optimizing it is studied in possible details from available literature as a first step. This provided the insight to formulate basic framework for the proposed work.

Next an analytical study of the selected approach (*Genetic Algorithms*) is carried out to understand nuances and intricacies. From this we gain insight into possible opportunity to contribute into the knowledge domain, specifically in applying the approach to RCPS problem. A few short but critical knowledge gaps facilitate further work in to attempt contribution.

Based on knowledge gathered, the algorithm is designed by a combination of two types of segments – adaptation of robust portions as put forward by the literature and studies, and our self-designed portions.

Subsequently, the design is developed into a program for computational work.

The Design of Experiment (DoE) is made simultaneously by identifying parameters to be experimented with.

For validation of the algorithm, experimentation is carried out as per Design of Experiment. Predetermined alteration(s) of test parameters are carried out, and impact is studied on performance outcome. The results thereof are subjected to simple statistical analysis for validation of the algorithm from three angles, viz. effectiveness, accuracy and efficiency.

Depending on analysis of experimental results we fine-tune the algorithm, the program implementation, the parameters and their values, to finally comment on acceptance of the proposed algorithm as result of the work.

Relevant portion of the methodology are explained in details at pertinent Chapters.

1.9 Content Outlines of this Thesis

The main portion of this thesis is presented as eight Chapters, each dealing with a specific portion of our objectives. Next we present short appendices that act as supplement and reference material. Finally we include references and bibliography.

Chapter 1 is just being concluded. Here a discussion was made about background of the problem – from macro angle of field and area of our study to micro angle of niche identification for the present work. This Chapter has also spelt out boundaries and methodology of the work.

Chapter 2 provides introduction and discussion about the RCPS Problem, which is the specific problem taken up for study. The first two Chapters provide a background to the area of work, on which we have built our objectives.

Chapter 3 is a literature survey of different approaches made for addressing the RCPS Problem, and other related problems. This Chapter endeavor to focus on the first of the three subsidiary objectives.

Chapter 4 is a description of the ‘approach’ taken up for usage in the current study, and which would be taken up as basis of proposed algorithm. This Chapter deals with functional description of Genetic Algorithms.

Chapter 5 describes formulation of the proposed algorithm. The attempt at contribution of this present work to the knowledge domain is candidly outlined here in stages. Chapter 4 and Chapter 5 (in whole or partially) is an effort to address the second of the three subsidiary objectives.

Chapter 6 describes the implementation of the algorithm for computational and experimental works. The development of the conceptual algorithm into programmable simplification is described in this Chapter. The Design of Experiment is made in the later half of the Chapter.

Chapter 7 discusses experimental outcomes for tuning and acceptance of the algorithm. Chapter 6 and Chapter 7 attempts to fulfill the third subsidiary objective.

Chapter 8 is the concluding Chapter of this thesis, and is a condensed appraisal of the work. Few strengths and shortcomings of the proposed algorithm is discussed here. Finally, this Chapter transfers focus from the present work towards possible extensions and related future works.

The References and Bibliography enumerates digital library and Internet sources separately prior to literature listing. The literature listing contains references mentioned and/or quoted in this text as well as bibliographical entries that were referred to and consulted during research but not directly mentioned and/or quoted here.

Chapter Two

The Problem

In this Chapter we discuss the Problem taken up for study. The nature, intricacies, complexities, assumptions, etc are dealt to the extent feasible. Being a well-established problem, we leave out the (technical) details and mention general aspects of the problem. The first portion deals with description and formulation of the RCPSP. This we follow by short definition of terms used for the descriptions.

2.1 Introduction

The Resource-Constrained Scheduling Problem (RCSP) is not new. It was since 1950s that systematic solution approaches to planning and scheduling methods were taken up. RCSP has been studied from a number of angles, for varied applications. Each application field modified and adapted basic RCSP to satisfy requirement(s) its own peculiarities. But basic framework and objectives more or less remains the same. In a number of applications, resource allocation aspect is prime motivation. For many, the control and optimization of activity tardiness assumes higher importance. In saying this we declare that 'time' as a resource is considered separately than the other (physical) resources.

Study of RCS initially started with job sequencing in Shop-Floor, and allocating finite number of machines, operators, etc. Scheduling problems tend to be difficult, not just in theory, but in practice as well. Applegate and Cook(1991) remarked that the job shop problem is not only NP-hard, it also has the well-earned reputation of being one of the most computationally stubborn combinatorial problems to date. In their book, Muth and Thompson(1963) introduced a ten machine, ten job problem that took the Operations Research community more than two decades to arrive at a plausible solution set.

Project Management is a field where the RCSP has been utilized extensively. The focus of the present study is application of RCSP as adapted for the area of Project Management. For application into the area of Project Management, the Operations Research community has rechristened Resource Constrained Scheduling Problem (RCSP) as Resource-Constrained Project Scheduling Problem (RCPSP).

2.2 The Resource-Constrained Project Scheduling Problem

The Resource-Constrained Project Scheduling Problem (RCPSP) consists of a set of tasks, and a set of finite capacity resources. Each task puts some demand on the resources. A partial ordering of these tasks is given specifying that some tasks must precede others.

Generally the goal is twofold –

- a) to minimize makespan without violating the precedence constraints, and/or
- b) avoid over-utilizing the resources.

The focal point of this problem is formulation of sequence of tasks (events or activities) for optimal utilization of the resources (usu. reusable) keeping into account the temporal restrictions. Thus there are resource constraints as well as sequence rules.

Amongst researchers in the area of Project Scheduling, the quest has been to find out the best way of assigning the resources to the activities within spatio-temporal limitations so as to achieve the best objective(s) possibly within the prime constraint. Formulating total job sequence right from start of the project till its end (completion, or abandonment), with possible parallel paths, and simultaneously allocating resources has been attempted with the help of many methods and algorithms, classically with Network Analysis.

With time, contemporary techniques – most of them evolving in the area of Operations Research, are being employed to take the RCPSP towards its optimal solution. A study of literature in this area reveals that application of heuristic techniques and design of metaheuristics is an area of intensive research. Because of its nature that defies finalization of algorithm, ever-new methodologies are constantly and consistently evolving to take the RCPSP towards optimal. In the next section, we give the general formulation of RCPSP in short.

2.3 General Formulation of the Problem

The RCPSP may be described as follows :

- #1 Given
 - a) a set of activities that must be executed,
 - b) a set of resources and their capacity limitations to be utilized,
 - c) a set of quality objectives by which one may judge performances
- #2 All within the non-negotiable boundary of a (set of) constraints
- #3 What would be the best way of assigning the resources to the activities within the constrain limitations so as to achieve the best objective(s) of completion of the Project.

As adapted from Crawford (1996) the RCPSP can be depicted:

Given :

- a set of tasks, T ,
- a set of resources, R ,
- a capacity function, $C : R \rightarrow N$,
- a duration function, $D : T \rightarrow N$,
- a utilization function, $U : T \times R \rightarrow N$,
- a partial order, P on T , and
- a deadline, d .

Objective :

To achieve $\text{Min } \sum D$, by assignment of start times $S : T \rightarrow N$

Subject to the constraints :

- a) Precedence constraints : If t_1 precedes t_2 in the partial order P , then $S(t_1) + D(t_1) \leq S(t_2)$
- b) Resource constraints : For any time x , let $\text{running}(x) = \{t | S(t) \leq x < S(t) + D(t)\}$. Then for

all times x , and all $r \in R$,

$$\sum_{t \in \text{running}(x)} U(t,r) \leq C(r).$$

- c) Temporal Constraints : For all tasks t : $S(t) \geq 0$, and
 $S(t) + D(t) < d$.

For arriving at the optimal solution, the problem may be tackled from a number of angles, such as

- a) Formation of task sequence,
- b) Allocation of scarce resource to the tasks,
- c) Delimitation of time-windows of the tasks,
- d) Defining and designing of alternative mode of execution, etc

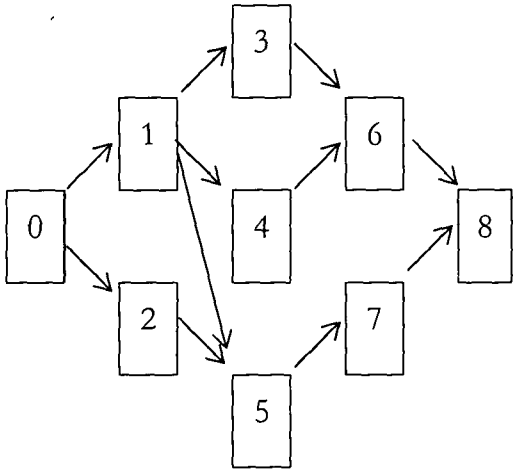
While the problem may be viewed from different angles, but all of them share certain common definitions for characteristics of the Project. At the end of this Chapter we define some of these for ready reference as well as adapted for our study.

2.4 Definition of Terminology

For our definitions, let us use a Bridge under construction as a continuous example.

i) Project	<p>An organized endeavor aimed at accomplishing a specific non-routine objective. This objective may be systematically broken down to a number of activities or tasks.</p> <p>For example, construction of a bridge is a project.</p>
ii) Activities or Tasks	<p>A (module of) work or the units of Work Breakdown Structures (WBS). In other words, the exact set of work that needs to be carried out, which in totality is the 'Project'. These activities have to be completed for the Project to qualify as completed.</p> <p>For example, the bridge construction consists of survey, purchase of construction materials, hiring of engineers and workers, soil testing, approach road, span, etc.</p>
iii) Resources	<p>These are the input(s) that would go into the activities. The Project may have a number of different resources. An activity may require more than one resource. Also, a resource may be used by more than one activity. Resources may be either reusable or consumable. They are usually constrained in their availability – both from time as well as quantity perspectives.</p> <p>For example, bridge construction would require trucks, cement mixing units, cement, bitumen, paint, steel, workers, water, etc.</p>
iv) Capacity Limitations	<p>The maximum amount of availability of a specific resource. Or the limit to which a certain resource may be utilized.</p> <p>For example, the Project Manager may have 200 workers, but at</p>

	any point of time not more than fifty may be allotted for approach road works.
v) Utilization	<p>The degree to which a certain specific resource is used – either at a certain point of time during execution of the project, or at completion of the project.</p> <p>For example, the bridge used up five hundred eighty tonnes of steel, out of six hundred tonnes ordered.</p>
vi) Partial Order	<p>The order in which the Tasks are set out to be executed. These are generally dependent of some precedence / succession requirements.</p> <p>For example, soil testing has to be carried out before span erection. Order for cement may be placed simultaneously with that for steel, irrespective of sequence.</p>
vii) Duration	<p>The time required to carryout a certain activity. In most of the Projects this is done by estimations based on previous experiences of similar activities. There are two distinct durations – Activity Duration, in which the specific activity is to be completed; and Project Duration, in which all activities of the project has to be completed.</p> <p>For example, the bridge has to be ready for traffic two years from now, when the Commonwealth Games would take place nearby. The Project Manager estimates that soil testing would take a minimum of eight days, but in worst case, shouldn't take longer than fifteen days.</p>
viii) Deadline	The time or date set specifically for completion of a certain activity. On the macro front, Project deadline is the specified time

	<p>by which the Project has to be completed, ready for delivery.</p> <p>For example, the sixth prestressed span unit has to be placed on 15th of September.</p>
<p>ix) Quality Objectives</p>	<p>A (set of) benchmark(s) by which the quality of execution of a specified (set of) activity(ies) is measured.</p> <p>For example, the maximum allowable offset between spans can be three millimeters.</p>
<p>x) Constraints</p>	<p>These are the limitations under which the Project has to be completed. As is expected, the various factors of task execution are not free-to-will, and they have to be reined in by certain rules.</p> <p>For example, the Project Manager has resource limitations.</p>
<p>xi) Precedence Constraints</p>  <p>Figure 2.1 : A Project Network, (Activity on Node)</p>	<p>The specific sequence by which tasks has to be taken up is defined as Precedence Constraint. There may be more than one '<i>preceding</i>' task which requires completion before a specific task is commenced. On the other hand, completion of a certain task can pave way for more than one '<i>succeeding</i>' tasks. The third option is '<i>parallel</i>' task, which may be carried out irrespective of (some other) task(s).</p> <p>In the Project depicted in Figure 2.1, task 6 can start only when task 3 and 4 are completed. Again, completion of task 1 allows the Manager to take</p>

	up any of task 3 and 4, but not 5 unless task 2 is also complete.
xii) Resource Constraints	The constraint of availability of project inputs is termed as Resource Constraints. It is worth mentioning here that 'time' is usually not included in the 'resource' list, even though apparently it is one such. Due to its critical nature, time is treated as a separate dimension on which other constraints are scaled.
xiii) Temporal Constraints	The constraint imposed on and by time in completion of activities of a Project is termed as its Temporal Constraint.
xiv) Assigning the Resources	The (set of) rule(s) by which resource(s) are scheduled and allocated to the tasks of the Project.
xv) Completion of the Project	The state of the Project in which all tasks are completed (or terminated in case of Project abandonment).

Apart from these common ones, we provide a Glossary of a few specifically coined definitions for our algorithm at Appendix I.

Chapter Three

Related Work

In this Chapter we present a literature survey of related work in the direction of evolving optimization techniques and their usage for RCPSP, and a comparative analysis of candidate approaches.

Information has been gathered by referring to published literature, both under copyright as well as public. It is understood that considering vastness of the work done, this Chapter is a sample representation. But we aspire to provide as fair a representation as possible on the different fronts.

We conclude the Chapter with a comparative discussion of three approaches to bring us into a convergence of the problem at hand and the chosen approach from amongst the candidates.

3.1 Introduction

The RCPSP is a well-known NP-hard problem, which means that there is no known optimal solution method in polynomial time. As with any other NP-hard problem it is obvious that solution approaches for this problem would evolve with time, and diversity of the approaches would be high.

Kolisch and Hartmann (2006) (*quoting Mohring, et al(2003)*) very nicely puts this in perspective, “Due to the fact that the RCPSP ‘is one of the most intractable problems in Operations Research’ it has ‘become a popular playground for the latest optimization techniques, including virtually all local search paradigms”

Before the use of computers for project scheduling, Project Manager carried out scheduling manually. This was both time consuming as well as erroneous. But more importantly, there was no guarantee of optimal solutions. This was truer if the number of activities and their precedence constraints were more.

Variation of the Resource Constrained Scheduling Problem has been suggested and their solution worked out for implementation and evaluation. There are two basic approaches for approaching this problem – exact and heuristic, as deliberated at length by Wall (1996).

3.2 Solution Approaches

3.2.1 Exact methods

This approach attempts to identify the exact optimal solution. Classical approaches that tend to pinpoint exact solutions falls under this category. In the earlier days when the number of activities and / or constraints was small, the exact methods provided crisp solutions. But with the rise in complexity of the problem these methods were impractical.

One of the pioneering work done in the field of scheduling was by Balas(1971) who laid down a structural approach that involves a generalization of both the disjunctive graph method in job shop scheduling and the order theoretic methods for precedence constrained scheduling.

Lawler et al (1982) provided a conceptual summary of works and developments done in the area of deterministic scheduling and scheduling.

Bartush (1988) and co-workers carried a number of pioneering and referential works in the area of algorithm generation for scheduling problems in construction industry. They have contributed literature for integrating computers with project scheduling.

A treatment for solving complexities of scheduling project networks with precedence constraints was carried out by Lenstra et al (1978).

3.2.1.1 Network Based Approaches

Programme Evaluation and Review Technique (PERT), its predecessor GERT and Critical Path Method (CPM) are the prominent methods of getting to possible solutions, and are based on Networks (or more precisely, activity networks).

First introduced in the 1950s during the development of the Polaris missile system, PERT is a forerunner of formal project scheduling. This method introduced the characterization and representation criteria set for precedence requirements and time estimation. However in its truest sense, PERT is not a scheduling method *per se*, but rather a method for organizing information and defining constraints. But the importance of activity representation as chalked out by PERT technique (or some derivative thereof) is a prerequisite for many solution methods. The prominent characteristic of PERT is its ability to absorb probabilistic estimates of task duration.

CPM provides the resource-unconstrained schedule for a set of precedence-constrained activities with deterministic durations. However it assumes availability of infinite quantum on resources, and provides the shortest possible makespan. Although CPM is useful for obtaining a rough idea of the difficulty of executing a plan, it does not consider temporal or resource constraints.

During the mid '80s, algorithmic work on project networks scheduling and resource allocation were carried out by Mohring R.H. as well as Radermacher F.J., both jointly as well as independently.

Blazewicz (1983) made an attempt of introducing dynamic variations into the CPM approach. To bring the Critical Path Method closer to reality, stochastic variations were constructed by Slowinski and Weglarz(1989), Neuman(1990), and others. Both attempts tried to incorporate the probabilistic estimates of task duration, thereby creating a hybrid approach by bridging PERT and CPM.

3.2.1.2 Operations Research Approaches

Linear Programming (LP) and Integer Programming (IP) are two classical methods for formulating many scheduling problems. But to attempt a solution, these methods require a significant level of simplification, which gets gradually impractical with the increase in number of tasks and constraints. Patterson(1984) compared a number of optimal solution methods for project scheduling using exact approaches.

Exact methods depend on characteristics of the objective function and specific constraint formulations. Many of the constraints commonly found in real scheduling problems do not auger well to traditional Operations Research or Mathematical Programming techniques (Davis (1985)). This restricts their usage in real life situations. In addition, the linear programming formulations typically do not scale well, so they can be used only for specific instances or small problems.

While Mathematical Programming techniques represents a well-understood and established problem-solving method, many of the formulations and algorithms for solving such programs optimally can be extremely opaque, and the formulations themselves can be difficult to specify, modify, and understand. In addition, the amount of computational effort required to solve an RCSP can fluctuate widely across a set of similar RCSPs, meaning that the same approach may incur reasonable computational effort for one problem and exponential effort for another. Finally, the introduction of real-world dynamic complications into large-scale RCSPs greatly increases the difficulty involved in their solution.

A slightly modified approach would be to develop an optimal schedule incrementally by first constructing for small number of tasks, and then extending that schedule by adding tasks until all tasks are scheduled. This is the method of Dynamic Programming. Hindelang and Muth (1979) offered a complete dynamic programming formulation for the problem in Decision CPM context. Later on, De Reyck et al (1998) however have shown that the Hindelang-Muth procedure is flawed, and they attempted to provide the necessary corrections. (Adapted from Demeulemeester, (2002))

Khamooshi(1999) attempted cross-breeding Dynamic Programming with the Dynamic Priority Scheduling Method (DPSM). DPSM divides a project into phases (cycles), the length of which depends on the duration of the project and the period of clock cycle selected. The scheduling process starts by allocating resources to the first phase/cycle using a variety of policies, then the best schedule is selected based on an objective function. The process continues till all the activities are scheduled. In DPSM the interaction between phases is ignored while the decisions of each phase or cycle will affect all the remaining phases. He opined that it might be possible to improve the quality of a schedule and reduce the duration of a project by optimizing the overall project schedule.

While such Dynamic Programming approaches can significantly reduce computational effort, the chief concern is with the large amount of space required to store the intermediate results calculated by these algorithms.

3.2.1.3 Enumerative Approaches

In another approach for finding solution to the Project Scheduling problem, a decision tree mechanism based on the precedence relation in the project plan is used. This branch-and-bound methodology was extensively researched by Sprecher and Drexler(1995), who noted that the enumerative methods can solve the problem with many different objectives. Enumerative methods are pseudo-heuristic in the sense that the sizes of the trees are typically bounded using heuristics. If allowed to propagate to its full size, the trees would branch into numerous, mostly illogical branches. Generally, a pruning algorithm is utilized for trimming new branches.

Variations of branch and bound solution methods were proposed much earlier for different applications. Stinson et al (1978) in their pioneering branch and bound approach, generated a tree by scheduling activities starting with the first task then adding a node to the tree for each task that could be scheduled based upon precedence and resource constraints Bounds based on partial schedules were used to prune the search tree.

Enumerative methods cannot solve large problems; the tree gets simply too big (Wall, 1996). Being quite effective in providing viable solution set, Patterson et al (1978), Kolisch et al (1992), Demeulemeester and Herroelen (1992), Sprecher et al (1995) amongst others, have refined the pruning algorithms for this method. Although significant progress has been made in the pruning techniques, branch and bound methods are still require special heuristics to accommodate variations in resource constraint formulations.

More often than not, a ‘good’ solution is good enough for the Project rather than seeking that elusive ‘exact’ solution. To overcome the basic constraint of size and complexity, researchers tried to seek out alternative methodologies and techniques.

The practical utility of scheduling is succinctly put by Zwikael, et al (2006). It often happens that a Project Manager who negotiated for resources in his project cannot afford to use a delay concept in scheduling. With this in mind, a new branch and bound based non-delay scheduling algorithm for a set of RCPSP was tested in their work that demonstrated its ability to find (near) optimal solutions very fast.

Deblaere, et al (2007) developed a procedures for allocating resources to the activities of a given baseline schedule in order to maximize its stability in the presence of activity duration variability. They proposed three integer programming–based heuristics and one constructive procedure for resource allocation.

3.2.2 Near-Optimal Heuristic Approaches

The degree of elusiveness of optimal solution for a RCPSP increases exponentially with the increase in the number of nodes. It gets worse with increase in number of constraints. And finally, when one realizes that in real world, stability of a project is a fool’s expectation (*we refer to the Prologue*), the situation goes totally haywire if work could be allowed to proceed only after finding the ‘exact’ solution. Thus evolved the concept of seeking for a ‘good’ solution, by possible juxtaposition of traditional approach with heuristic approaches.

The potential of research and application of heuristics for PMIS was aptly demonstrated when five commercial Project Management packages was evaluated for performance by Mellentien and Trautmann(2001) using accepted benchmark test data-set and procedures. The quality of resultant schedules decreased significantly when the packages dealt with Projects of realistic scenario – comprising a large number of activities and scarce resources. The results indicate

that none of the methods utilized in the packages could offer competition with (at that time contemporary) heuristic algorithms from the literature, even though all the packages utilized fast heuristics. On the other hand, the makespan deviation from solutions that can be achieved with modern RCPS methods would justify implementation of additional algorithms (into the packages).

As a prelude to the area, we give a generic description of heuristics –

A heuristic is a method to help to solve a problem, commonly informal. It is particularly used for a method that often rapidly leads to a solution that is usually reasonably close to the best possible answer. Heuristics are "rules of thumb", educated guesses, intuitive judgments or simply common sense.

In more precise terms, heuristics stand for strategies using readily accessible, though loosely applicable, information to control problem-solving in human beings and machines.

In computer science, a heuristic is a technique designed to solve a problem that ignores whether the solution can be proven to be correct, but which usually produces a good solution or solves a simpler problem that contains or intersects with the solution of the more complex problem.

Heuristics are intended to gain computational performance or conceptual simplicity, potentially at the cost of accuracy or precision.

Extracted from <http://en.wikipedia.org/wiki/Heuristic>, accessed on 8th June, 2008

Here the last sentence bears high significance, which hints at the popularity and usefulness of heuristics. People in the industry need to solve large-scale project scheduling problems quickly. In most applications, a near-optimal solution is preferable if hunting for the optimal solution consumes a high amount of time and computational resources. Heuristic methods typically require less time and/or space than exact methods. The exact solution to a project-scheduling problem requires extensive computational time which does not meet the Project Manager's

need for interactive use of software. This justifies use of heuristics, and the bouquet of research for its application to solve the RCPSP

Heuristics may be deterministic – they end up with the same result every time; or stochastic – each run provides a different result. Heuristic in scheduling are rechristened as *scheduling rules*. The definitions of these rules are often quite complex, and most are tailored for a specific type of problem with a very specific set of constraints and assumptions. In most of recent applications (within the last ten years or so), hybrid algorithm that employs multiple heuristics is being experimented with, and is throwing up ‘good’ results. Recognition of the problems of trying to achieve optimal solutions led to a shift in focus towards other methods for finding *near-optimal*, or *approximate* solutions to RCPSPs at less computational expense, in terms of both time and space.

Traditionally, for RCPSP, heuristic methods typically follow three steps i) Planning, ii) Sequencing, and iii) Scheduling. Some methods use heuristics to search the combinatorial space of permutations in task sequences, others use heuristics to determine feasible time/task/resource assignments during the schedule generation. Still others use heuristics to combine sequencing and scheduling. Precedence constraints typically dominate the search in the sequencer, whereas resource constraints dominate in the scheduler.

Bhaskar et al (2004) discuss uncertainty at different phases in project scheduling and then provides a method for handling uncertainty at the planning phase. The work proposes a priority rule for a new schedule generation scheme of RCPSP, which takes care of the criticality of the activities and the randomness involved in the current and future activities.

Xu et al (2007) augments priority rule heuristics by creating rollout procedures and proves their effectiveness using sampling to generate a set of schedules through probabilistic techniques and select the best schedule from this sample.

In the PhD thesis, Hildum (1994) noticed that within the context of the RCPSP a heuristic schedules those tasks having the earliest possible starting times, or the least available amount of slack time. An heuristic approach to solving an RCPSP operates by applying a heuristic (or collection of heuristics) to the set of unsolved subproblems comprising the RCPSP to determine the relative priority of each individual subproblem. In this sense, the heuristic serves as a rating function for determining the order in which activities are to be scheduled. The standard heuristic approach first orders the set of activities and then proceeds by assigning resources to each activity in sequence. In some cases, near optimal results can be achieved, and the approach incurs less computational expense. It is therefore more applicable to practical real-world problems having extremely large search spaces. A *single-attribute* heuristic is distinguishable from a *multiple-attribute* heuristic in terms of the degree of analysis undertaken. Examples of single-attribute scheduling heuristics are the MINEST and MINLFT rules, which assign priority to those activities having the earliest starting times(EST), and latest finishing times(LFT), respectively.

As aptly observed in the same work, main drawback of single heuristics is the lack of analysis performed. A narrow evaluation of the state of problem solving can allow important developments to go unnoticed and permit serious problems to develop in the future. In such situations, multiple-attribute heuristic performs a more extensive analysis of the current state of problem solving, and can therefore act to prevent such problems from developing. A survey of scheduling rules, ranging from simple priority rules to more complex heuristics, has been presented by Panwalkar and Iskander(1977). Upper bounds (assuming a minimizing objective) on the quality of a number of approximate heuristic solutions to RCPSPs are presented in Garey et al (1978).

A great deal of work has been directed towards the development of heuristics that produce near-optimal solutions, and the determination of the best heuristics to use

in certain circumstances. For the most part, however, with the exception of a few specific cases, it has not proven possible to establish a definitive classification for matching a particular class of RCSP with a particular scheduling heuristic.

3.2.2.1 Single Heuristic Approaches

A comparison of standard heuristics for solving RCSPs was undertaken by Davis and Patterson (1975) where eight single heuristics were applied to a set of single-project, multiple-resource problems. The results, compared to an early work by Davis and Heidorn (1971), indicated that minimum time and minimum slack based heuristics performed the best in terms of achieving the optimal schedule, or coming the closest in percentage to the optimal. The results also showed, however, that the performance of all heuristics suffered when resource were tightly constrained. While this study was applied to small problems, it suggests that heuristics that consider various kinds of time, and the degree of resource usage, generally produce better (near-optimal) results.

An attempt to classify scheduling problems for the purpose of identifying appropriate scheduling heuristics for their solution is described by Kurtulus and Davis (1982). Two metrics were defined for characterizing scheduling problems according to average resource load and average resource utilization. A collection of heuristics was tested on a set of sample problems, and the results suggested that two of the tested heuristics were generally the best performers. The first heuristic favored the shortest activity in the shortest project (job), while the second favored the activity with the highest combined required resource load (obtained by multiplying the activity resource requirement by the activity processing duration) and cumulative project resource load (obtained by summing the required resource loads for all activities already scheduled within the same project).

Lawrence and Morton (1993) describe a single-heuristic approach that attempts to minimize weighted tardiness through the use of a combination of project-related,

activity-related, and resource-related metrics. They defined a general priority heuristic that weighs the tradeoff between the cost of delaying an activity on a resource, and the benefit obtained by using that period of delay to assign the resource to some other activity. The results of this approach were compared against the results from twenty standard scheduling heuristics on a set of some 14,400 individual RCPSPs that ranged in size from 125 to 250 activities distributed among five projects, and varied in tardiness penalty, activity duration, and resource requirements.

3.2.2.2 Multiple Heuristic Approaches

Multiple heuristics for solving scheduling problems represents an extension to the application of standard single heuristic approach. The goal is to exploit the strengths of a number of different scheduling heuristics in an attempt to increase the chances of producing near-optimal (and occasionally optimal) schedules according to the particular scheduling objective.

A multiple heuristic approach can be seen as a formal application of the idea of consulting multiple scheduling perspectives. One heuristic might evaluate urgency based on a simple analysis of the time bound constraints on an. Another might consider the expected activity duration while a third might consider some combination of the two. A scheduler equipped with a variety of such heuristics is better able to react to variation in characteristics of search space.

Boctor (1990) indicates that inclusion of specific heuristics can result in frequent development of near-optimal, and occasionally optimal, schedules. Heuristics based on minimum slack (MINSLK) proved to be the most important of the single heuristics included in any combination. MINLFT heuristic, characterising finish time of an activity, proved to be a valuable companion to MINSLK. Hildum (1994) notes that larger combinations of heuristics demonstrated increased ability to produce higher quality schedules, suggesting that there is a clear benefit to

applying a wide range of scheduling perspectives in the process of determining the urgency of each individual activity.

3.2.3 Artificial Intelligence Approach

Hildum(1994) grouped artificial intelligence (AI) approach to scheduling as either expert systems or knowledge-based. He emphasized that his own method, DSS (Dynamic Scheduling System), is basically a multiple attribute, dynamic heuristic approach that focuses on the most urgent unsolved problem at any given time.

Rabelo et al (1995) attempted a hybrid approach for real time sequencing and scheduling problems. They explored hybridization of Neural Networks, Genetic Algorithm, Simulation and Machine Learning for their study.

Hartmann and Kolisch during early part of the century have made explorative studies of techniques to incorporate intelligent techniques for solving the RCPSp.

3.2.4 Simulated Annealing

Simulated annealing (SA) replicates the annealing process of metallurgy, in which a metal is strengthened by a process of heating and cooling. Starting with a (set of) initial solution, the 'neighborhood' solution (set) is generated by 'energizing' the current solution (set). In case the new solution is better, search proceeds towards that direction after accepting it.

SA may be considered as an extension of a simple greedy procedure, First Fit Strategy, which immediately accepts a better neighbor solution, but rejects any deterioration (Zbigniew and David 2000).

Simulated Annealing is a variation of hill-climbing where neighbors with less good objective values are sometimes selected to keep the search from getting stuck in local optima, observed Smith (2004).

Initial experimentation of using SA for scheduling problems was carried out by Boctor (1990). This work maintained precedence feasibility by restricting the neighborhood operator to only precedence feasible task swaps.

3.2.5 Tabu Search

Tabu Search (TS) developed by Glover and Greenberg (1989), is essentially a steepest descent / mildest ascent method. In other words, it evaluates all solutions of the neighborhood, and chooses the best – from which it proceeds further.

Logical question arises then that what if this search results in a recursive spiral between two neighbors. This is avoided by setting up a tabu list (*taboo: forbidden, 'tabu' being a different spelling of the same word*) of forbidden results in the first stage of TS. This stage is the *preliminary search* stage, and proceeds for a specified number of search iterations. In the second (*intensification*) stage of the search, it (a) starts with the best solution found so far (which is always stored throughout the entire algorithm), (b) clear the tabu list, and (c) proceed as in the first phase for a specified number of moves. Finally in the *diversification* phase, the tabu list is cleared again and set the most frequent moves of the run so far to be the tabu. Then a random location is targeted, and again process repeats from the first phase for a specified number of iterations.

Thus by searching from a different direction, a confirmation is made of the specific location where the solution exists. This phase holds a magnifying glass to promising regions discovered till the spot(s) are all identified, explains Glover and Laguna (1997).

Tabu search has been used extensively by researchers for scheduling algorithms and have achieved high quality results, as reported by Pinson, et al (1994).

The potential of TS is yet to be exploited fully. In the words of its progenitor, Glover (2007), not only does a great deal remain to be learned about Tabu Search,

but it is equally true that very little is yet known about how we ourselves use memory in our problem solving. It is worth stressing again that discoveries about effective uses of memory within our search methods may provide clues about strategies that humans are adept at employing—or may advantageously be taught to employ. The potential links between the areas of heuristic search and psychology are an intriguing concomitant to research now underway and have scarcely been examined. Progress in the design of tabu search methods, and the successful applications of TS that have occurred so far, provide encouragement that such issues are profitable to probe more fully.

3.2.6 Ant Colony Optimization

Ant Colony Optimization (ACO) studies artificial systems that take inspiration from the behavior of ants when they find and report food back to their mates in the ant-colony. They utilize natural chemicals of their body to mark and note locations, as well as pass on direction message to other ants. Thus one set of ant does the scouting, and others follow his path – depending on the marks left. In case a path is not used for a period of time, the mark evaporates, indicating fruitlessness of search in that direction.

This adaptation of the Natural World is being used to solve discrete optimization problems, and is a fairly new technique. Merkle and Schmeck (2002) report that they have achieved good results in project scheduling using this algorithm. Their proposed algorithm combines the direct (local) and summation (global) pheromone evaluation methods, to finally get rid of local minima. Further they discuss the changing strength of heuristic influence, the changing rate of pheromone evaporation over the ant generations.

3.2.7 Fuzzy System

Pan and Yeh (2003) present a Fuzzy Genetic Algorithm (FGA) metaheuristic approach that incorporates fuzzy set theory to model the uncertain activity duration times for optimizing the RCPSP. This study provides the framework of a metaheuristic approach for solving RCPSP involving uncertain activity duration times modeled by fuzzy numbers.

3.2.8 Genetic Algorithm

Genetic Algorithm (GA) is inspired by the process of biological evolution. Most of the search methods described above does local search. But GA considers a 'population' of solutions instead of one.

A set of Initial Population is generated, and new solutions are produced from them by mating two (or more) 'parents' and/or by altering the characteristics of a single one. The former procedure is termed as 'crossover', and the later is termed as 'mutation'. After producing new solutions, the fittest solutions survive and become the next generation, while others are deleted. The fitness value measures the quality of a solution, usually based on the objective function value of the optimization problem (Zbigniew and David (2000)

Holland (1975) provided the pioneering work in adapting Natural Systems' processes for Artificial Systems. He laid the foundations by introducing Genetic Algorithms, based on the transfer of genetic information in the natural world from the parents to progeny. A slightly detailed discussion of Genetic Algorithm is presented as a full Chapter later in this work.

3.3 Additional Works

Crawford(1996) approached the RCPSP by a combination of limited discrepancy search to arrive at a near final form heuristics for problems of realistic size and character. This work was run on a series of problems made available by Barry Fox of McDonnell Douglas and Mark Ringer of Honeywell, serving as Benchmark Secretary in the AAAI Special Interests Group in Manufacturing and in the AIAA Artificial Intelligence Technical Committee respectively.

To understand the mechanisms of a dynamic system, the use of System Dynamics is a good tool. Love et al (2001) used System Dynamics to understand change and reworks in the construction projects. This powerful tool can be adapted to generate scenarios, and synthesize instance sets.

Project Management was provided with a radically new methodology - SYDPIM, by Rodrigues(1997) which integrates the use of System Dynamics simulation models with the traditional PERT/CPM network models.

A related but not exactly the same field of study is process resequencing. Here the whole work sequence is altered, without violating rules and precedence, and alternative sequences are developed. Then these new sequences are subjected to study for identifying a (set of) better possible alternative to the original sequence. Attempts to apply Genetic Algorithms to this were initiated out by Altus et al (1996) and Rogers (1996).

Alvarez-Valdes et al (1989) described a heuristic algorithm based on empirical analysis for the RCPSP in the late '80s. Neumann and Franck provided structural questions and priority-rule methods for the RCPSP with time windows. In a few other works, Zimmermann J combined with Neumann to produce additional literature.

3.4 Three Candidate Approaches

Kolisch and Hartmann(2006), in the updated report have given short comparative description of application of heuristics and metaheuristics by different researchers for the RCPSP. On their list, most of the work that produced ‘good’ results are metaheuristics. Similar was the trend in two earlier compilations by Kolisch and Padman(1997) and Hartmann and Kolisch(2000).

The Committee on the Next Decade of Operations Research, CONDOR Report (1988), singled out Tabu Search, Simulated Annealing, and Genetic Algorithms as ‘extremely promising’ optimization methods for the years to come. The foresightedness of this report is vindicated by the fact that these three approaches are still being widely used for optimization – albeit with extensive adaptations and modifications. The application of the three methods for the RCPSP, along with pros and cons for doing so, are given below, as collated from literature of different authors.

3.4.1 Tabu Search

Dell’Amico and Trobian (1993) and Nowicki and Smutnicki (1996) have reported high quality project scheduling results using Tabu Search.

Thomas and Salhi (1998) introduced a Tabu Search method that operates directly on schedules. Since the resulting neighbor schedules may be infeasible, they employ a repair procedure to turn an infeasible schedule into a feasible one.

Klein (2000) develops a so-called Reactive Tabu Search method for the RCPSP with time-varying resource constraints. It is based on activity list representation and serial SGS. The neighborhood is given by swap moves, which include the shifting of predecessors or successors of the swapped activities if the resulting list would otherwise not be precedence feasible.

Nonobe and Ibaraki (2002) suggest a Tabu Search approach for a generalized variant of the RCPSP. Considering only the features that are relevant for the standard RCPSP, the heuristic employs the activity list representation, the serial SGS, shift moves, and a specific neighborhood reduction mechanism.

Artigues et al (2003) devised a Tabu Search procedure that essentially, selects iteratively an activity which is first deleted from the schedule and afterwards re-inserted with a network flow-based insertion algorithm.

3.4.1.1 Advantages of Tabu Search

- i) Since it 'knows' where it had been, a Tabu Search can provide faster search earlier by avoiding taboo zones.
- ii) Tabu Search would be very successful in situations where rules for acceptance are loose as compared to rules for rejection. Usage for Medical diagnostics falls in this category.

3.4.1.2 Disadvantages of Tabu Search

- i) The Tabu list might become extremely large, making search within the list a tedious affair
- ii) Because of a growing Tabu list, computational resource might tend to become scarce or even dry up.
- iii) In case of defective control mechanism, there exists a possibility of cyclical recursion.
- iv) Tabu Search operates with mutation as its mechanism by producing multiple mutants simultaneously. This makes it extremely vulnerable to 'localization', a factor to be avoided for scheduling problems.

3.4.2 Simulated annealing

As noted by Kirkpatrick et al (1983), Simulated Annealing approaches require a schedule representation as well as a neighborhood operator for moving from the current solution to a candidate solution. Annealing methods allow jumps to worse solutions and thus often avoid local sub-optimal solutions.

Aarts et al (1988) described one of the first Simulated Annealing approaches to scheduling problems. Boctor (1993) reported fairly good performance by a simulated annealing approach on the Patterson problems. In this work, Simulated Annealing was used to search the combinatorial space of sequence permutations. Given a sequence of tasks generated by the annealer, heuristics were then used to create a schedule. This method is directly analogous to the exact branch and bound solution, but whereas branch and bound is practically limited by the size of the decision tree, Simulated Annealing can be applied to much larger problems.

Valls et al (2004) tested a Simulated Annealing method that focuses on forward-backward improvement, where a neighbor is constructed by selecting the next activity either in the order of the original solution or by biased random sampling.

3.4.2.1 Advantages of Simulated Annealing

- i) Simulated annealing is a related global optimization technique that traverses the search space by testing random mutations on an individual solution. A mutation that increases fitness is always accepted. This implies that as soon as a possible optimal zone is reached, Simulated Annealing can approach the peak extremely fast.
- ii) Simulated annealing can be used with a standard Genetic Algorithm by starting with a relatively high rate of mutation and decreasing it over time along a given schedule. This hybridization has produced good results for RCPSP.

3.4.2.2 Disadvantages of Simulated Annealing

- i) Simulated Annealing plays the ‘blind-man-buff’ game, to arrive at the foothills.
- ii) Just as Tabu Search, Simulated Annealing also operates by mutation – but produces only one mutant.
- iii) Simulated Annealing arrives at a solution by ‘strengthening’ identified results. It is possible that due to distractions, annealing takes place at a wrong, or sub-optimal locality.

3.4.3 Genetic Algorithms

Alcaraz and Maroto (2001) developed a Genetic Algorithm based on activity list representation and serial SGS. Alcaraz et al (2004) extended the same by adding two features from the literature. First, they take the additional gene that determines the SGS to be used from Hartmann (2002). Second, they employ the forward backward improvement of Tormos and Lova (2001).

Hartmann (2002) proposes a so-called self-adapting Genetic Algorithm that extends the activity list representation by adding a gene, which determines whether the serial or the parallel SGS is to be used for transforming an activity list into a schedule. As a prerequisite for the procedure, it is defined how the parallel SGS can be used as decoding procedure for activity lists. The choice of the more successful SGS is left to the inheritance and survival-of-the fittest mechanisms.

A Genetic Algorithm based on the activity list representation, the serial SGS, and the related order-preserving crossover strategy where the initial population is produced by a pure random mechanism was suggested by Hindi et al (2002)

Coelho and Tavares (2003) present a Genetic Algorithm that makes use of the activity list representation and the serial SGS. They suggest a new crossover

operator for activity lists called late join function crossover that constructs a new individual by “adopting the father solution and swapping each adjacent pair that is in reverse order in the mother.”

Goncalves and Mendes (2005) use a random key representation and a modified parallel SGS. The modified parallel SGS determines all activities to be eligible which can be started up to the schedule time plus a delay time.

Valls et al (2003) extend a previous work on the activity list-based Genetic Algorithm with forward-backward improvement to what they call a Hybrid Genetic Algorithm. They develop a peak crossover operator that uses properties of the schedule when recombining activity lists.

Can et al (2004) implement *Genetic Algorithm for RCPSP* by introducing an additional component in the encoding of Alcaraz and Moroto (2001) to indicate the scheduling mode (forward or backward) used to generate the corresponding schedule.

Godley et al (2007) demonstrates two novel crossover approaches for Genetic Algorithm when applied to the optimization of time-series problems, with particular application to bio-control schedules. It is possible that adaptation of such novel improvisation hopefully breed fitter child-solution from fit parents.

Yassine et al (2007) proposes a Genetic Algorithm hybridized with a local search strategy, to minimize the overall duration or makespan without violating resource constraints or precedence constraints.

Mohsenin and Ali (2008) designed Genetic Algorithms operators for a new solution model of Resource Constrained Project Scheduling Problem with heterogeneous resources (operators). The model better resembles real-world projects and has more flexibility than previous models for manpower scheduling.

3.4.3.1 Advantages of Genetic Algorithm:

- i) A Genetic Algorithms can quickly scan a vast solution apace.
- ii) Bad solutions found during processing do not affect the end solution negatively as they are simply discarded.
- iii) Since it works away from problem-related characteristics, the Genetic Algorithm doesn't have to know any rules of the problem - it works by its own internal rules. This nature is very useful for complex or loosely defined problems.

3.4.3.2 Disadvantages of Genetic Algorithm:

- i) While the greatest advantage of Genetic Algorithm is the fact that they find a solution through evolution, this is also the biggest disadvantage. Evolution is inductive; in nature life does not evolve towards a good solution - it evolves away from bad circumstances. This can cause a species to evolve into an evolutionary dead end.
- ii) Genetic Algorithm is 'arrogant', in the sense that it operates on its own without taking cognizance of problem complexities. This calls for tighter reining in of the metaheuristic.
- iii) Unless properly formulated, Genetic Algorithm risk finding a sub-optimal solution, and the 'evaluator' may not know of it unless there is a (set of) comparative result.
- iv) There is a fair possibility that solution evolved at a later generation might have been discarded in an earlier generation, thereby laying waste processing effort.

3.5 Choice of Approach

Amongst these three methods, we select the Genetic Algorithm approach as an application to optimize the RCPS Problem. We propose to exploit advantages of Genetic Algorithm, as its features are more inclined for application in optimizing the RCPS.

But the same would not be implemented in its classical format. For implementing Genetic Algorithm to optimize RCPS, Kolisch and Hartmann(2006) observed that pure Genetic Algorithm are hardly developed any more. Instead, the basic Genetic Algorithm scheme is modified or extended by integrating additional features, such as path relinking, forward-backward improvement, self-adapting mechanisms, non-standard crossover, or even other metaheuristics.

3.6 Test Data Set and Benchmark Results

Kolisch and Sprecher (1996) provided a set of test instances for the evaluation of solution procedures for the RCPS, which are now internationally recognized and accepted test data-set. This is stored and available at the PSPLIB 'library', and is accessible to researchers for downloading and evaluating newer algorithms. PSPLIB is constantly updated and augmented by the RCPS researcher fraternity. The present work would use information from / of this source for test and evaluation of the proposed algorithm, and possibly contribute to it.

The same library also 'archives' optimal and current-best results. Compilations of results produced by this library are used for reporting performance of algorithms. Majority of RCPS literature refer to this library for the test instances, and then updated results for comparison.

Chapter Four

The Approach: Genetic Algorithm

In this Chapter we analyze major components of Genetic Algorithm for their functional intricacies. The information provided is a collation from various accepted and public literature. Certain portion of information is referred to the authors / researchers who had either proposed or propounded them.

(Generic definitions and explanations are adopted from available literature and websites)

4.1 Introduction

Genetic Algorithms are general-purpose search algorithms based upon the principles of evolution observed in nature. These algorithms combine selection, crossover, and mutation operators with the goal of finding the best solution to a problem by searching for ‘the optimal solution (set)’ until a specified termination criterion is met.

The solution to a problem is called a chromosome. A chromosome is made up of a collection of genes which are simply the parameters to be optimized. A Genetic Algorithm creates an initial population (a collection of chromosomes), evaluates this population, then evolves the population through multiple generations (using the genetic operators discussed above) in the search for a good solution for the problem at hand.

4.1.1 History and Evolution

Genetic Algorithms was developed by J H Holland (with his colleagues and students) at the University of Michigan. The primary theme of research on Genetic Algorithms has been robustness – the balance between efficiency and efficacy necessary for survival in many different environments. Features of the biological world that aids efficient balance like self-repair, self-guidance, and reproduction are being (attempted to be) replicated in artificial systems. A careful study of the intricacies and secrets of the biological world, and possible usage of these into artificial systems, led to the primary monograph on the topic, ‘*Adaptation in Natural and Artificial Systems*’ by Holland(1975). Subsequent users of Genetic Algorithms have time and again established that the beauty of Genetic Algorithms lies not only in simply replicating Nature, but its ability to provide robust search in complex search spaces.

4.1.2 Application Area

Having established itself as a valid approach to problems requiring efficient and effective search, Genetic Algorithms are finding widespread application in bioinformatics, computer science, engineering, economics, chemistry, manufacturing, mathematics, physics and other fields.

Two very convincing reasons are behind the widespread applications of Genetic Algorithms. These are computationally (relatively) simple, yet powerful in their search for improvement. A more powerful reason is that Genetic Algorithms are not fundamentally limited by restrictive assumptions about the search space.

Genetic Algorithms are highly effective in any situation where many inputs (variables) interact to produce a large number of possible outputs (solutions). Some example situations are:

- Optimization such as data fitting, clustering, trend spotting, path finding, ordering.
- Management: Distribution, scheduling, project management, courier routing, container packing, task assignment, University time-tables.
- Financial: Portfolio balancing, budgeting, forecasting, investment analysis and payment scheduling.
- Engineering: Structural design (eg beam sizes), electrical design (eg circuit boards), mechanical design (eg optimize weight, size & cost), process control, network design (eg computer networks).
- R & D : Curve and surface fitting, neural network connection matrices, function optimisation, fuzzy logic, population modeling, molecular modeling and drug design.

4.1.3 Mechanism

Goldberg(1989) described Genetic Algorithms as search algorithms based on the mechanics of natural selection and natural genetics. These are based on evolutionary process as observed in Nature. They follow the dictum of '*Survival of the Fittest*' among string structures with a structured yet randomized information exchange to form a search algorithm with some of the innovative flair of human search. In every (next) generation, a new set of artificial strings is created using bits and pieces of the fittest of the old (parent) generation. An occasional new part is tried for good measures – for introducing diversification.

When applied to problems whose search space is very large and where the ratio of the number of feasible solutions to the number of infeasible solutions is low, care must be taken to properly define the representation, operators, and objective function, otherwise the Genetic Algorithms will perform no better than a random search.

Pupong et al (2008) proposed a Genetic Algorithm based optimization tool that minimizes total costs associated within supply chain logistics. Their proposed model had with chromosome initialization procedure, crossover and mutation operations defined in a way that always guarantee feasible solutions to be embedded. A half fractional factorial design was carried out to investigate the influence of alternative crossover and mutation operators by varying GA parameters. The analysis of experimental results suggested that the quality of solutions obtained is sensitive to the ways in which the genetic parameters and operators are set.

Many Genetic Algorithms appear to be more robust than they actually are only because they are applied to relatively easy problems. Gruninger(1996) showed that the genetic operators must effectively balance exploration and exploitation so that the Genetic Algorithms will be able to both avoid local minima/maxima in global

search and find small improvements in local search. In addition, small changes to the algorithm and genetic operators had a significant impact on the algorithms performance.

After a population has evolved, all individuals typically end up with the same genetic composition – the individuals have converged to the same structure. If the optimum has not been found, then the convergence is, by definition, premature. In most cases, further improvement is unlikely once the population has converged. By maintaining diversity in the population, the algorithms have a better chance of exploring the search space and avoid a common problem of Genetic Algorithms – ‘premature convergence’. The flow of Genetic Algorithms process is illustrated in Figure 4.1.

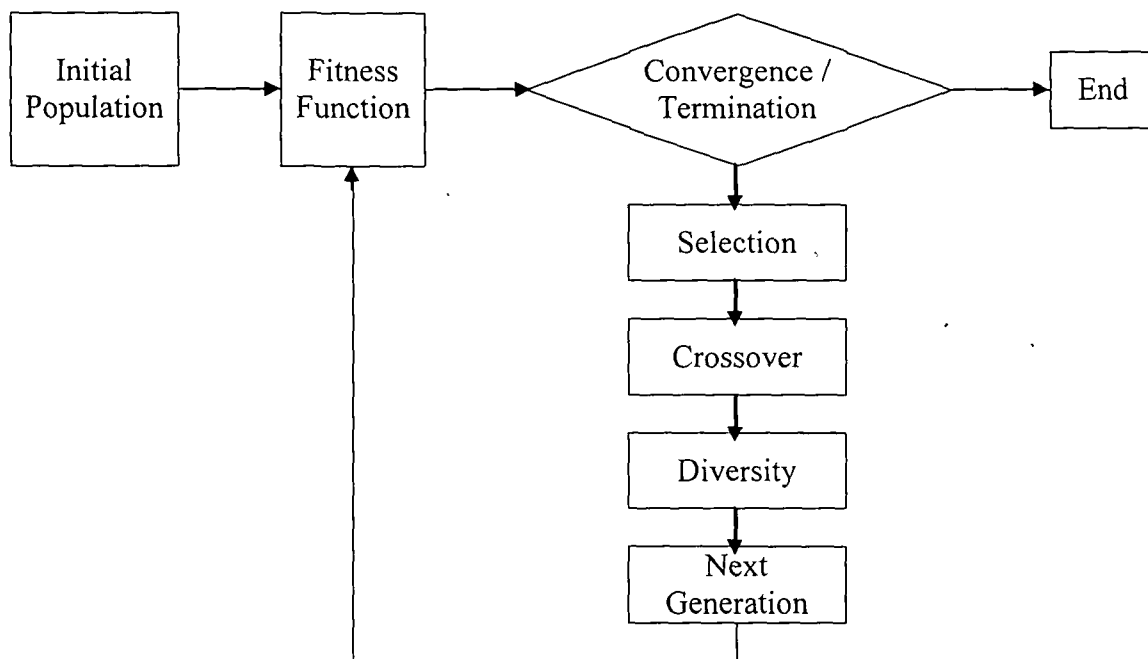


Figure 4.1: Process of Genetic Algorithm

4.2 Types of Genetic Algorithms

Genetic Algorithms operate independently of the problems to which they are applied. This makes application domain of Genetic Algorithms very dynamic, flexible and flexible. The flexibility of Genetic Algorithms lies in the fact that there is no guarantee a Genetic Algorithm will converge to an optimal solution, although experience suggests that a properly parameterized algorithm performs quite well. Parameters involved in a Genetic Algorithms generally include: population size, number of generations to simulate, mating selection method, diversification or mutation rate, and the reproduction strategy. The genetic operators are heuristics, but rather than operating in the space defined by the problem itself (the solution-space or phenotype-space), genetic operators typically operate in the space defined by the actual representation of a solution (the representation-space or genotype-space) (Wall(1996)).

Because of its adaptive and flexible characteristics, Genetic Algorithms are constantly evolving and are being exotically named. But the primary mechanism and process remains the same. By generic definitions, Wall(1996) have classified Genetic Algorithms into three types

- a) The Simple Genetic Algorithms : One of the more common and '*primitive*' Genetic Algorithm implementations.
- b) The Steady-State Genetic Algorithms : Made popular by the GENITOR program, and
- c) The Struggle Genetic Algorithms : A kind of speciating Genetic Algorithm developed by Gruninger (1996).

4.2.1 Simple Genetic Algorithm (Non-Overlapping Populations)

The Simple Genetic Algorithm uses non-overlapping populations. In each generation, the entire population is replaced with new individuals. Typically the best or the 'elite' (set of) individual is carried over from one generation to the next so that the algorithm does not inadvertently forget the best that it found. This is referred to as 'elitism'. Maintaining the best individual also causes the algorithm to converge more quickly; in many selection algorithms, the best individual is more likely to be selected for mating.

4.2 Steady-State Genetic Algorithm (Overlapping Populations)

The Steady-State Genetic Algorithm uses overlapping populations. In each generation, the newly generated individuals replace a portion of the population. At one extreme, only one or two individuals may be replaced each generation (close to 100% overlap). At the other extreme, the steady-state algorithm becomes a simple Genetic Algorithm when the entire population is replaced (0% overlap). Since the algorithm only replaces a portion of the population of each generation, the best individuals are more likely to be selected and the population quickly converges to a single individual. As a result, the Steady-State Genetic Algorithm often converges prematurely to a suboptimal solution.

4.2.3 Struggle Genetic Algorithm

In Struggle Genetic Algorithm rather than replacing the worst individual, a new individual replaces the individual most similar to it, but only if the new individual has a score better than that of the one to which it is most similar. This requires the definition of a measure of similarity (often referred to as a distance function). The similarity measure indicates how different two individuals are, either in terms of their actual structure (the genotype) or of their characteristics in the problem-space (the phenotype).

4.2.4 Specially devised Genetic Algorithms

Apart from the three classical Genetic Algorithms described above, researchers have proposed many variations of the technique. OLGA, IGA, NSGA-II, etc are some of these.

In OnLine Genetic Algorithm (OLGA) the fitness of an individual changes over time, as it is exposed to more examples. The key idea in creating such a Fitness Function is to support newly created individuals so that they are not replaced before they have seen a reasonable number of examples (and thus have some estimate of their true fitness) as described by Davison (1998).

Interactive Genetic Algorithm (IGA) uses human evaluation where it is hard or impossible to design a computational Fitness Function, for example, evolving images, music, various artistic designs and forms to fit a user's aesthetic preferences. These algorithms belong to a more general category of Interactive evolutionary computation.

NSGA-II or Non-dominated Sorting Genetic Algorithm-II (and its previous version, NSGA) proposed by Deb et al (2002) has low computational requirements. It is an elitist approach with parameter-less niching, and has simple constraint-handling strategy.

Other variants exists, viz. Multi-Objective Genetic Algorithm (MOGA), Neighborhood Cultivation Genetic Algorithm (NCGA), Vector Evaluated Genetic Algorithm (VEGA), etc., and the list is ever growing. Keeping basic framework of the technique the same, researchers work on different components and parameters. Depending on specific modification and/or application, the Genetic Algorithm is named appropriately.

Next we proceed to discuss the features and characteristics of Genetic Algorithm.

4.3 Features of Genetic Algorithms

The 'next' generation of the Natural world carries with it characteristics of the 'parent' generation. This transfer of characteristics is provided via parental efforts. Such characteristics are embedded in packets called 'chromosomes'. The parental 'genes' provide the survival probability to the offspring. Or in other words, the 'fitness for survival' of the offspring is inherited from such 'binary interaction' of the parents. Depending upon the degree of fitness, the offspring either survive or gets annihilated. Those that survive now assume the role of 'parents' for producing their offspring i.e. generates the 'next' generation.

But once in a while, changes occurs in (one or more of) the offspring to 'genetically modify' the inherited characteristics. Such 'mutation' occurs independent of parental efforts. This sporadic change introduces a novelty factor into the offspring generation, and previously unforeseen impacts might be displayed in subsequent generations. This change in genetic characteristics on singular entity all by itself may (paradoxically) be considered a 'unary interaction'.

Another unary interaction may be possible if an 'alien', who was not generated from the immediate parents, infiltrates the offspring set. It may either be self-generated, or migrate from another parent set, or travel down from a previous generation, or a combination of these. Such an 'immigrant' would also provide a change in the expected characteristics of the subsequent generations.

The unary operators provide diversification to a population. In Nature, this takes place mostly for facilitating adaptation to changes in the environment. However in Genetic Algorithms this technique is utilized for spreading out of the search locality within the search domain. It has been found that such a (forced) diversification is an extremely effective tool for avoiding local maxima and thereby evading premature convergence. Diversity is important in genetic algorithms because crossing over a homogeneous population does not yield new solutions.

The members of a certain generation who fails to live up to the expectations or survival requirements are gradually screened out of the population. Therefore only those who are found 'fit' will 'survive' and proceed to produce the 'next' generation. From the period when life first appeared on this planet, this has been the only procedure that the best have qualified to sustain their presence. This vindicates the robustness of the '*Survival Of The Fittest*' model as was propounded by Darwin.

Researchers, for developing the Genetic Algorithms, have successfully adapted this methodology of 'survival of the fittest'. The features of adaptation in Natural selection have been studied in minute details for adoption into Artificial systems, and these are termed as Genetic Operators. Even after decades of research, this is still an ongoing process with numerous possibilities for advancement of the genetic operators.

4.3.1 Chromosome Representations

In 1866, Mendel recognized that in Nature the complete (character) information for each individual lies in pair-wise 'alleles'. The genetic information that determines the properties, appearance and shape of an individual is stored in 'chromosomes' (David 2002).

A chromosome describes a string of a certain length where all the genetic information of an individual is stored. Although nature often uses more than one chromosomes (e.g. X- and Y- chromosomes in Human), most Genetic Algorithms uses only one chromosome for encoding the genotypic information. Each chromosome consists of many alleles. An allele is the smallest information unit in a chromosome. A gene is a region of a chromosome that must be interpreted together, and which is responsible for a phenotypic property (David 2002). Figure 4.2 describes representation of a chromosome, as generically used for Genetic Algorithms.

It is essential that a standard mechanism be agreed upon at the very outset for chromosome representation. A typical representation of the chromosome is as an array of bits. Arrays of other types and structures can be used in essentially the same way. Experimentation of representation by real values are also very much in vogue. Structures may be single-dimensional or multi-dimensional array. Tree-structures have also been experimented with.

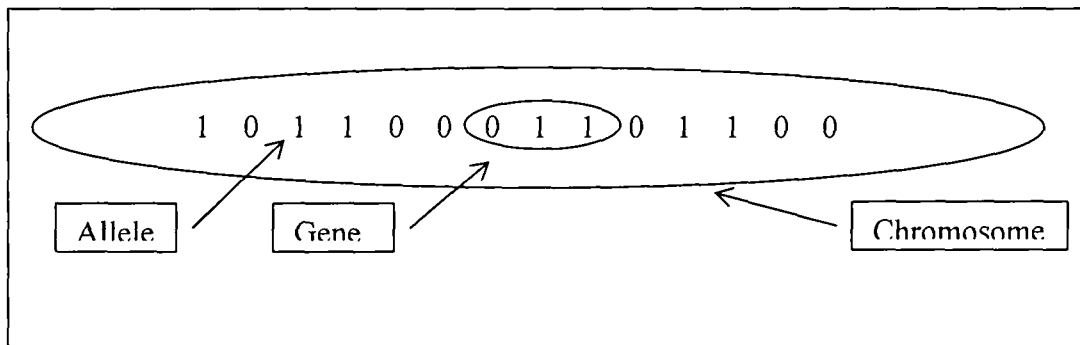


Figure 4.2 : Representation of a Chromosome

The alleles may be binary-coded or value-coded, depending on the nature of application of the proposed Genetic Algorithms. In case of value-coded representations, we may encode the chromosome directly with integers or real number values, or even some permutations. Multi-dimensional matrix as well as tree-structure representations is also worked with. Experiments with either pure representation or hybrids throws up further vistas of research. In a recent work, Pupong, et al (2008) experimented with multi-matrix real-coded Generic Algorithm (MRGA).

The main property that makes these genetic representations convenient is that their parts are easily aligned due to their fixed size, which facilitates simple crossover operation. Variable length representations may also be used, but crossover implementation would be more complex.

4.3.2 Initial Population

The initial population for a Genetic Algorithm is a set of solutions to the optimization problem. Just as an initial starting point dictates the quality of a gradient-based non-linear optimization algorithm, the initial population can affect Genetic Algorithms solution convergence. Some characteristics of any population are objective function value, feasibility of the solution, and level of infeasibility for any infeasible solutions. There are a variety of approaches to generating initial populations.

A common (often default) method of population generation is random generation. Occasionally, the initial solutions may be "seeded" in areas where optimal solutions are likely to be found. Hill(1999) proposed that initial populations for Genetic Algorithm applications be randomly generated based on problem knowledge. He devised a Monte Carlo based simple heuristic for randomly generating good initial populations for genetic algorithm applications to two-dimensional knapsack problems.

4.3.3 Selection Operator

Selection (*a 'trigger' vide our categorization*) chooses a chromosome from the current generation's population for inclusion in the (process of creating the) next generation's population. Before making it into the next generation's population, selected chromosomes may undergo crossover and / or mutation (depending upon the probability of crossover and mutation) in which case the offspring chromosome(s) are actually the ones that make it into the next generation's population. During each successive generation, (a proportion of) parents in current population are 'selected' to breed a new generation. Individual solutions are selected through a fitness-based process, where fitter solutions (as measured by a Fitness Function) are typically more likely to be selected. Certain selection methods rate the fitness of each solution and preferentially select the best

solutions. Other methods rate only a random sample of the population, as this process may be very time-consuming.

Most 'selection functions' are stochastic and designed so that frequent selection of 'stronger' strings are favoured and only a small proportion of less fit, or 'weak' strings are selected. This helps keep the diversity of the population large, preventing premature convergence on poor solutions. We describe a few generic Selection methodologies here.

4.3.3.1 Fitness Proportionate Selection

Fitness Proportionate Selection, also known as Roulette-Wheel selection, is a selection operator in which the chance of a chromosome getting selected is proportional to its fitness (or rank). This is where the actual utilization of the concept of survival of the fittest comes into play. In fitness proportionate selection the Fitness Function is used to associate a probability of selection with each individual chromosome, as a function of the population size. If $f(i)$ is the fitness of individual i in the population, its probability of being selected $p(i)$ is

$$p(i) = f(i) / \text{Summation } f(j), \quad \text{for } j \text{ from } 1 \text{ to } N,$$

where N is the population size.

The analogy to a roulette wheel can be envisaged by imagining a roulette wheel in which each candidate solution represents a pocket on the wheel; the size of the pockets are proportionate to the probability of selection of the solution. Selecting N chromosomes from the population is equivalent to playing N games on the roulette wheel, as each candidate is drawn independently.

While candidate solutions with a higher fitness will be less likely to be eliminated, there is still a chance that they may be. With fitness proportionate selection there is a chance some weaker solutions may survive the selection process; this is an

advantage, as though a solution may be weak, it may include some component that could prove useful following the recombination process.

A modified variation of this method is the Russian Roulette Selection, where the parent population keeps diminishing with each selection.

4.3.3.2 Tournament selection

In Tournament Selection, a "tournament" is run among a few strings (or individuals) chosen at random from the population, and selects the winner (the one with the best fitness) for crossover. Selection pressure can be easily adjusted by changing the tournament size. Tournament selection operator uses roulette selection N times to produce a tournament subset of chromosomes. The best string in this subset is then chosen as the selected chromosome. This method of selection applies additional selective pressure over plain roulette selection.

The chosen individual can be removed from the population that the selection is made from if desired, otherwise individuals can be selected more than once for the next generation. In the later option, we would have strings (or individuals) that have multiple copies or 'clones' within the population. Tournament selection is efficient to code, works on parallel architectures and allows the selection pressure to be easily adjusted.

4.3.3.3 Other selection methods

In addition to the ones mentioned above, researchers are trying out variations as well as novel selection methods for specific purposes, and have come up with different end results. A few of such methods are mentioned hereunder :

Top Percent Selection : A selection operator which randomly selects a chromosome from the top N percent of the population as specified by the user.

Best Selection : A selection operator which selects the best chromosome (as determined by fitness). If there are two or more chromosomes with the same best fitness, one of them is chosen randomly.

Random Selection : A selection operator that randomly selects a chromosome from the population.

Stochastic universal sampling: These have less stochastic noise, or are fast, easy to implement and have a constant selection pressure as explored by Blickle (1996).

This is only a partial list, as user specific Selection operators are being devised constantly by the Genetic Algorithm research fraternity.

4.3.4 Offspring Generation Operator

In Genetic Algorithms these are the most significant operators. The ‘parent(s)’ in one way or the other are operated upon and are modified to produce new solution(s), or ‘offspring’. Such operation may involve two parents for binary (or ‘sexual’) reproduction, or just one parent for unary (or ‘asexual’) reproduction.

4.3.4.1 Binary Reproduction Operator : Crossover

In Genetic Algorithms, ‘crossover’ is a genetic operator used to vary the programming of a chromosome or chromosomes from one generation to the next. It is analogous to reproduction and biological crossover, upon which Genetic Algorithms are based, and is inspired by the role of reproduction in the evolution of living things. Genetic Algorithms attempts to combine elements of existing solutions in order to create a new solution, with some of the features of each parent. The elements of existing solutions are combined in a crossover operation – adapted from the crossover of DNA strands that occurs in reproduction of

biological organisms. Arguably, crossover (or X-over, as is sometimes denoted as) is the most important of genetic operators.

Many crossover techniques exist for organisms which use different data structures to store themselves. The simplest way to do that is to (randomly) select a crossover point, copy everything before this point from the first parent, and then copy everything after the crossover point from the other parent. The string thus generated is the 'child' or the 'offspring', and is now a candidate for populating the 'next generation'. There are other ways to make crossovers, and can be quite complicated. It depends mainly on the encoding of the chromosomes, and nature of application of the Genetic Algorithms. Specific crossover model made for a specific problem can improve performance of the Genetic Algorithms. For use in their problem of maximising the efficiency of bio-control application utilising genetic algorithms, Godley et al(2007) described two specific crossover approaches – CalEB (Calculated Expanding Bin) and TinSSel (Targeted Intervention with Stochastic Selection). CalEB and TinSSel both use the number of interventions present in the parents to calculate the number required in the children, with CalEB utilising a "binning" approach to select the genetic material from the parents, whereas TinSSel contains an element of stochastic selection.

A poorly designed combination becomes a sort of mutation. Falkenauer(1998) noted that crossover means not only to use recombination, but that the recombination is indeed beneficial. There are several ways of testing whether a crossover technique performs correctly. One of them was proposed by Jones(1995), and is based on a idea : instead of mating two parents selected among the best in the population, generate one of the parents in random. If the crossover is really useful, this way of mating should lead to a significantly worse performance than the usual way. This would be because mating with a random parent amounts to performing a mutation instead of a crossover. We describe a few generic forms of crossover techniques here.

4.3.4.1.1 One Point Crossover

A crossover operator that randomly selects a crossover point within a chromosome then interchanges the two parent chromosomes at this point to produce two new offspring.

Consider Figure 4.3(a), where parents have been selected for crossover, and the arrow indicates the randomly chosen crossover point.

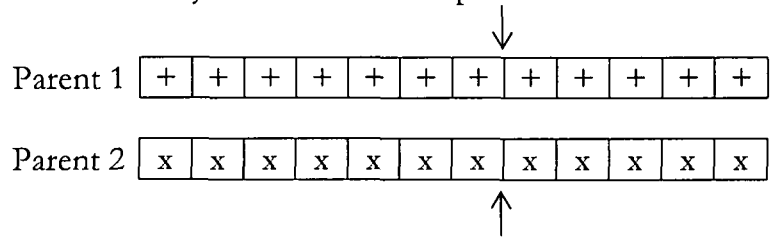


Figure 4.3 (a) : Parents ready for One Point Crossover

After interchanging the parent chromosomes at the crossover point, the offspring produced are depicted in Figure 4.3(b).

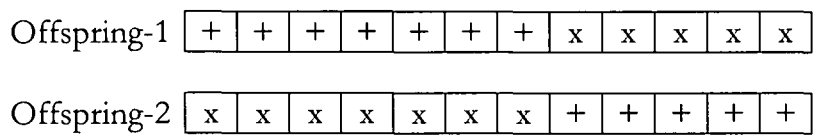


Figure 4.3 (b) : Offspring of One Point Crossover

4.3.4.1.2 Two Point Crossover

A crossover operator that randomly selects two crossover points within a chromosome then interchanges the two parent chromosomes between these points to produce two new offspring.

Consider the parents in Figure 4.4(a), which have been selected for crossover. The arrow indicate the randomly chosen crossover points.

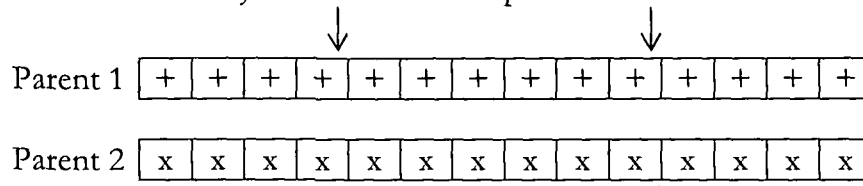


Figure 4.4 (a) : Parents ready for Two Point Crossover

After interchanging the parent chromosomes between the crossover points, the offspring are produced as given in Figure 4.4(b)

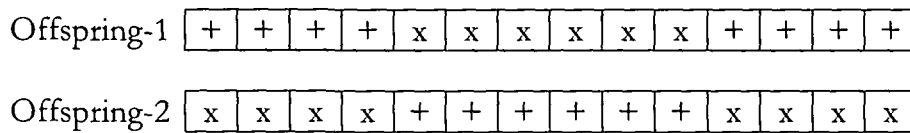


Figure 4.4 (b) : Offspring of Two Point Crossover

4.3.4.1.3 Uniform Crossover

A crossover operator that decides (with some probability) which parent will contribute each of the gene values in the offspring chromosomes. This allows the parent chromosomes to be mixed at the gene level rather than the segment level (as with one and two point crossover). Figure 4.5(a) the displays parents that have been selected for crossover:

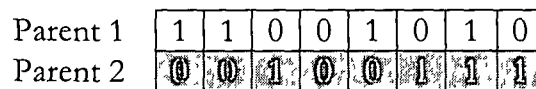


Figure 4.5 (a) : Parents ready for Uniform Crossover

If the probability is 0.5, approximately half of the genes in the offspring will come from parent 1 and the other half will come from parent 2. The probability of which unit to be exchanged may be determined using some function, or in random. Below is a possible set of offspring after uniform crossover:

Offspring 1	1	0	1	0	0	0	1	0
Offspring 2	0	1	0	0	1	1	1	0

Figure 4.5 (b) : Offspring of Uniform Crossover

4.3.4.1.4 Arithmetic Crossover

This is a crossover technique where the contents of the parents are not exchanged. Gene (or allele) values of the offspring are created by mating characteristics of the second parent to that of the first parent, using predetermined mating function. A simple mating function that linearly combines two parent chromosome vectors to produce two new offspring may be according to the following equations (or its modified form):

$$\text{Offspring1} = a * \text{Parent1} + (1 - a) * \text{Parent2}$$

$$\text{Offspring2} = (1 - a) * \text{Parent1} + a * \text{Parent2}$$

where a is a (random) weighting factor, chosen before each crossover operation.

Figure 4.6 depicts the selected parents (each consisting of 4 float genes)

Parent 1	0.3	1.4	0.2	7.4
Parent 2	0.5	4.5	0.1	5.6

Figure 4.6 (a) : Parents ready for Arithmetic Crossover

By using the above function with a = 0.7, arithmetic crossover would produce offspring as given in Figure 4.6(b):

Offspring 1	0.36	2.33	0.17	6.86
Offspring 2	0.402	2.981	0.149	6.842

Figure 4.6 (b) : Offspring of Arithmetic Crossover

This crossover allows a parent to retain positional integrity of the segments (genes or alleles) but at the same time be affected by corresponding units of the other selected parent. Thus the offspring are heavily biased towards the characteristics of their 'immediate' parent.

4.3.4.1.5 Heuristic Crossover

A crossover operator that uses the fitness values of the two parent chromosomes to determine the direction of the search. The offspring may be created according to the following relationship (or its modified form):

$$\text{Offspring1} = \text{BestParent} + r * (\text{BestParent} - \text{WorstParent})$$

$$\text{Offspring2} = \text{BestParent}$$

where r is a random number between 0 and 1.

It is possible that Offspring1 will not be feasible. This can happen if 'r' is chosen such that one or more of its genes fall outside of the allowable upper or lower bounds. For this reason, heuristic crossover has a user settable parameter 'n' for the number of times to try and find an 'r' that results in a feasible chromosome. If a feasible chromosome is not produced after 'n' tries, the WorstParent is returned as Offspring1.

4.3.4.1.6 Cut and Splice Crossover

The Cut and Splice crossover results in a change in length of the offspring strings. The reason for this difference is that each parent string has a separate choice of crossover point. Consider the following parents, each with a different crossover point determined by predefined function.

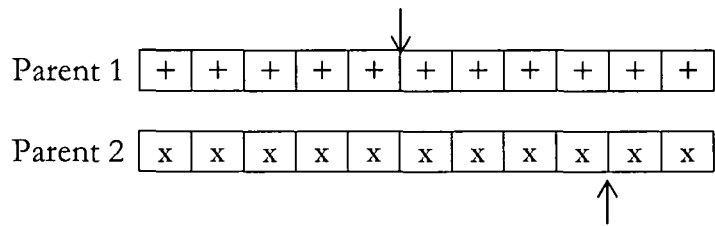


Figure 4.7 (a) : Parents ready for Cut and Splice Crossover

After crossover, the two offspring will be of different lengths, as depicted below

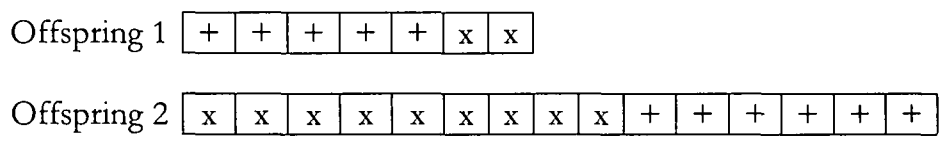


Figure 4.7 (b) : Offspring of Cut and Splice Crossover

4.3.4.1.7 Half Uniform Crossover

In the Half Uniform crossover scheme (HUX), exactly half of the nonmatching bits are swapped. Thus first the Hamming distance (the number of differing bits) is calculated. This number is divided by two. The resulting number is how many of the bits that do not match between the two parents will be swapped.

4.3.4.1.8 Crossover for Ordered Chromosomes

Depending on how the chromosome represents the solution, a direct swap may not be possible. One such case is when the chromosome is an ordered list, such as an ordered list the cities to be travelled for the traveling salesman problem. A crossover point is selected on the parents. Since the chromosome is an ordered list, a direct swap would introduce duplicates and remove necessary candidates from the list. Instead, the chromosome up to the crossover point is retained for each parent. The information after the crossover point is ordered as it is ordered in

the other parent. For example, if two parents are ABCDEFGHI and IGAHFDBEC and our crossover point is after the fourth character, then the resulting children would be ABCDIGHFE and IGAHBCDEF.

Other crossover techniques include the Precedence-Set crossover (PSX), Edge Recombination crossover (ERX), Partially Mapped crossover (PMX). In the Precedence-Set crossover (PSX) technique, the precedence integrity of one parent is maintained with respect to the positional preference of alleles in the other parent. Detailed description of PSX is carried out in a subsequent chapter.

4.3.4.2 Unary Reproduction Operators

Apart from binary (sexual) reproduction, where two parents contribute directly, offspring are produced by unary (or asexual) reproduction mechanism. More often than not, these operators are used for introducing diversity into a (possibly homogenous) population or generation.

4.3.4.2.1 Mutation

Mutation is a genetic operator that alters one or more gene values in a chromosome from its initial state. This can result in entirely new solution in the generation. With these new gene values, the Genetic Algorithm may be able to arrive at better solution than was previously possible. The purpose of mutation in Genetic Algorithms is to allow the algorithm to avoid local optima by preventing the population of chromosomes from becoming too similar to each other, thus slowing or even stopping evolution. This reasoning also explains the fact that most GA systems avoid taking only the fittest of the population in generating the next but rather a random (or semi-random) selection with a bias toward those that are fitter.

Mutation occurs during evolution according to a user-definable mutation probability. The classic example of a mutation operator involves a probability that an arbitrary bit in a binary genetic sequence will be changed from its original state. This probability is set fairly low (0.01 is a good first choice). If it is set too high, the search will turn into a primitive random search. The next few paragraphs describe a few common mutation techniques in Genetic Algorithms.

Flip Bit : A mutation operator that simply inverts the value of the chosen gene (0 goes to 1 and 1 goes to 0). In essence, 'flip' for binary representation is $ABS(\text{represented value} - 1)$.

Technically, it is also possible to use this method with other number methods, if the allele has upper and lower bounds.

For example, in case of positive single digit integer representation, flip can be a mutation with the transformation $(9 - \text{represented value})$ or in more generic form, $ABS(\text{represented value} - 9)$.

Boundary : A mutation operator that replaces the value of the chosen gene with either the upper or lower bound for that gene (chosen randomly).

Non-Uniform : A mutation operator that increases the probability that the amount of the mutation will be close to 0 as the generation number increases. This mutation operator keeps the population from stagnating in the early stages of the evolution then allows the Genetic Algorithm to fine tune the solution in the later stages of evolution.

Uniform : A mutation operator that replaces the value of the chosen gene with a uniform random value selected between the user-specified upper and lower bounds for that gene.

Gaussian : A mutation operator that adds a unit Gaussian distributed random value to the chosen gene. The new gene value is clipped if it falls outside of the user-specified lower or upper bounds for that gene.

Except for the flip method, other mutation operators can only be used for integer and float representation.

Selection is clearly an important genetic operator, but opinion is divided over the importance of crossover versus mutation. Some argue that crossover is the most important, while mutation is only necessary to ensure that potential solutions are not lost. Others argue that crossover in a largely uniform population only serves to propagate innovations originally found by mutation, and in a non-uniform population crossover is nearly always equivalent to a very large mutation (which is likely to be catastrophic). There are many references in Fogel (2006) that support the importance of mutation-based search, but across all problems the No Free Lunch theorem, propounded by Wolpert and Macready(1997), holds.

Sastry and Goldberg (2007) compared mutation with crossover head to head on exponentially scaled problems. They summarized that for deterministic exponentially scaled additively separable problems, mutation is more efficient than crossover. On the other hand, when noise (randomness induced multiple sub-optimal solution) dominates, crossover is more efficient than mutation. This is the premise on which our work have relied more on crossover leaving aside the mutation operator. Nevertheless, for diversity we test other unary operators. Leaving this debate to further research, another unary operator with its variants is discussed here.

4.3.4.2.2 Immigration

Garey et al (1979) proposed an immigration operator, which for certain type of functions, allows increased exploration while maintaining nearly the same level of exploitation for the given population size. Immigration, in one form or the other, is another unary technique of introducing diversity into the population. Immigrants can be introduced in more than one ways.

Alien Immigrant : As the name suggests, these are solutions for which the (immediate) parent generation is not responsible. ‘Aliens’ may infiltrate the present generation depending on overcoming a high barrier, i.e. the probability of immigration should be very low.

Dormant-Forefather Immigrant : During the process of Genetic Algorithm, a number of solutions are discarded along the way when the population moves into subsequent generations. If the search space is very large, there is high probability that these solutions are lost forever. The relatively higher fitness value of ‘false’ solutions would brush aside a ‘true’ solution with a lower fitness value.

The present work proposes to implement these two immigrants for introducing diversity in the population.

Frantz’s Immigrant

Goldberg (1989) mentioned about a partial complement operator, as proposed by Frantz. The partial complement operator (which Frantz called migration operator) complemented roughly a third of the bits of selected individuals in the population. These individuals were called immigrants (*we have added Frantz’s name to it, as above*) and were permitted to enter into the subsequent generation. This operator was intended to maintain diversity, but Frantz found out that the diversity was purchased at too high a cost – decreased performance.

4.3.5 Fitness Function:

The 'Fitness Function' measures the quality of the represented solution, and is defined over the genetic representation. It is always problem dependent. This is a specific objective function that quantifies the optimality of a solution in a Genetic Algorithms so that that particular solution may be ranked against all the other solutions. Optimal solution, or at least solutions which are more optimal, are allowed to breed and mix their datasets by any of several techniques, producing a new generation that will (hopefully) be even better.

An ideal Fitness Function correlates closely with the algorithm's goal, and yet has to be computed quickly. Speed of execution is very important, as a typical Genetic Algorithm must be multiple iterated in order to produce a usable result for a non-trivial problem. Definition of the Fitness Function is not straightforward in many cases and often is performed iteratively if the fittest solutions produced by GA are not what is desired. In some cases, it is very hard or impossible to come up even with a guess of what Fitness Function definition might be. In some problems, it is hard or even impossible to define the fitness expression; in these cases, Interactive Genetic Algorithms are suggested. Interactive Genetic Algorithms address this difficulty by outsourcing evaluation to external agents (normally humans).

In a Genetic Algorithm, the probability of reproduction directly depends on the fitness of each subject. That way the adaptive pressure of the environment is simulated. The implementation and evaluation of the Fitness Function is an important factor in the speed and efficiency of the algorithm.

But due to incorrect adaptation of the Fitness Function, there arises the possibility of 'polarization' when the population tends to converge towards the genome of a very strong solution, but which might be misleading. Another problem gets worse with the progress of the Genetic Algorithm. With time (or iterations), the

differences between fitness are reduced. The best ones then get quite the same selection probability as the others and the Genetic Algorithm stops progressing.

In order to palliate these problems, we discuss four scaling methods

- i) **Windowing** : For each subject, reduce its fitness by the fitness of the worse subject. This permits to strengthen the strongest subject and to obtain a zero based distribution.
- ii) **Exponential** : This method, proposed by Ladd(1996), consists in taking the square roots of the fitness plus one. This permits to reduce the influence of the strongest subjects.
- iii) **Linear Transformation** : For this, a linear transformation is applied to each fitness, i.e. $f' = a*f + b$, after ascertaining appropriate values of a and b. The strongest subjects are once again reduced.
- iv) **Linear Normalization** : Fitness are linearized. For example over a population of 10 subjects, the first will get 100, the second 90, 80 ... The last will get 10. Even if the differences between the subjects are very strong, or weak, the difference between probabilities of reproduction only depends on the ranking of the subjects.

To illustrate these methods, let us consider a population of four subjects to check the effect of scaling. For each subject, we give the fitness and the corresponding selection probability, as enumerated on Table 4.1

Careful observation of the result shows that Windowing eliminates the weakest subject - the probability comes to zero - and stimulates the strongest ones (the best one jumps from 50 % to 67 %). Exponential flattens the distribution. It is very useful when a super-subject induces an excessively fast convergence. Linear Transformation plays almost the same role than exponential. Linear Normalization

is neutral towards the distribution of the fitness and only depends on the ranking. It avoids as well super-subjects as a too homogeneous distribution. Apart from these, we may attempt to use Logarithmic and/or other composite transformations.

Subjects	1	2	3	4
	Fitness and the corresponding selection probability			
Initial or Calculated Fitness	50/50%	25/25%	15/15%	10/10%
Transformation Methods				
Windowing	40/66.7%	15/25%	5/8.3%	0/0%
Exponential	7.14/36.5%	5.1/26.1%	4.0/20.5%	3.32/16.9%
Linear transformation	53.3/44.4%	33.3/27.8%	20/16.7%	13.3/11.1%
Linear normalization	40/40%	30/30%	20/20%	10/10%

Table 4.1 : Comparison of Scaling Methods

4.3.6 Termination Criteria

Termination is the criterion by which the Genetic Algorithm decides whether to continue searching or stop the search. The termination criterion is checked after each generation to see if it is time to stop. Generally a single termination criterion is used, but multiple criteria combination by Genetic Algorithms is also in vogue. A few termination techniques are discussed here.

4.3.6.1 Generation Number

This is the most favoured termination criterion. It stops the evolution when the user-specified maximum number of evolutions has been run. This termination method is generally active.

4.3.6.2 Evolution Budget

A termination method that stops the evolution when the elapsed evolution time or cost exceeds the user-specified maximum. By default, the evolution is not stopped until the evolution of the current generation has completed, but this behavior can be changed so that the evolution can be stopped within a generation.

4.3.6.3 Fitness Threshold

A termination method that stops the evolution when the best fitness in the current population becomes less than the user-specified fitness threshold and the objective is set to minimize the fitness. This termination method also stops the evolution when the best fitness in the current population becomes greater than the user-specified fitness threshold when the objective is to maximize the fitness.

4.3.6.4 Fitness Convergence

A termination method that stops the evolution when the fitness is deemed as converged. Two filters of different lengths are used to smooth the best fitness across the generations. When the smoothed best fitness from the long filter is less than a user-specified percentage away from the smoothed best fitness from the short filter, the fitness is deemed as converged and the evolution terminates.

4.3.6.5 Population Convergence

A termination method that stops the evolution when the population is deemed as converged. The population is deemed as converged when the average fitness across the current population is less than a user-specified percentage away from the best fitness of the current population.

4.3.6.6 Gene Convergence

A termination method that stops the evolution when a user-specified percentage of the genes that make up a chromosome are deemed as converged. A gene is deemed as converged when the average value of that gene across all of the chromosomes in the current population is less than a user-specified percentage away from the maximum gene value across the chromosomes.

4.3.6.7 Manual Inspection

Instead of program triggered termination, in certain Genetic Algorithms the evolution or the process is terminated by manual inspection. This is usually done if the Fitness Function for the Genetic Algorithm defies definition or is too complex to be devised. For example, evolving images, music, taste of coffee, color set of the user interface, various artistic designs and forms to fit a user's aesthetic preferences, etc. Interactive Genetic Algorithm is one such application that uses human evaluation, and these have been generically termed as Aesthetic Selection.

4.4 Elitism

Elitism is the technique where the best solution (or a few best solutions) or 'elite(s)' is copied to the population in the next generation. The rest are chosen in classical way. Elitism can very rapidly increase performance of GA, because it prevents losing the best-found solution to date.

A variation is to eliminate an equal number of the worst solutions, i.e. for each "best chromosome" carried over a "worst chromosome" is deleted.

4.5 Building Blocks Hypothesis

The Building Block Hypothesis (BBH) proposed by Goldberg(1989) is a description of an abstract adaptive mechanism that performs adaptation by recombining "building blocks", i.e. low order, low defining-length schemata with above average fitness. That a Genetic Algorithm performs adaptation by implicitly and efficiently implementing this abstract adaptive mechanism is the premise of the Building Block Hypothesis.

Comparing his explanation of the Building Block Hypothesis with that of a child building a magnificent fortress out of simple wooden blocks, Goldberg claims that the hypothesis is supported by Holland's schema theorem. Goldberg describes the abstract adaptive mechanism as short, low order, and highly fit schemata are sampled, recombined, and resampled to form strings of potentially higher fitness. In a way, by working with these particular schemata, it reduces the complexity of the problem; instead of building high-performance strings by trying every conceivable combination, we construct better and better strings from the best partial solutions of past samplings.

For crossover operators which exchange contiguous sections of the chromosomes (like the PSX operator) the ordering of variables may become important. Despite advantages of building blocks, in many situations ignoring building blocks for generating favourable solutions have proved to be a better option. This is true for two specific reasons

- a) building block acts like sub-sets, thereby reducing number of possible combinations, and
- b) there is a possibility of premature convergence due to biased direction of search prompted by the building blocks.

The Building Block Hypothesis has been criticized on the grounds that it lacks theoretical justification and experimental results have been published that draw its veracity into question. Watson and Jansen (2007) contended that crossover in Genetic Algorithm can assemble short low-order schemata of above average fitness (building blocks) to create higher-order higher-fitness schemata. But they accepted that there has been considerable difficulty in demonstrating this rigorously and intuitively. Skepticism of the Building Block Hypothesis has previously been expressed on account of the weak theoretical foundations of this hypothesis and the anomalies in the empirical record of the simple Genetic Algorithm, notes Burjorjee(2008) in a yet to be published research paper.

On the theoretical side, for example, Wright (2003) state that the various claims about Genetic Algorithm that are traditionally made under the name of the Building Block Hypothesis have, to date, no basis in theory and, in some cases, are simply incoherent.

On the experimental side uniform crossover was seen to outperform one-point and two-point crossover on many of the Fitness Functions studied by Syswerda (1991). Summarizing these results, Fogel(2006) remarks that generally, uniform crossover yielded better performance than two-point crossover, which in turn yielded better performance than one-point crossover.

Syswerda's results contradict the Building Block Hypothesis because uniform crossover is extremely disruptive of short schemata whereas one and two-point crossover are more likely to conserve short schemata and combine their defining bits in offspring produced during recombination.

4.6 Knowledge Domain

As with all current heuristics and metaheuristics application for scheduling problems, it is worth tuning the parameters such as mutation probability, recombination probability and population size to find reasonable settings for the problem being worked on in this work. A very small change in acceptance probability rate may lead to genetic drift. A recombination rate that is too high may lead to premature convergence of the Genetic Algorithm. An immigration rate that is too high may lead to loss of characteristics of the Genetic Algorithm. There are theoretical but not yet practical upper and lower bounds for these parameters that can help guide selection.

Genetic Algorithms has been used extensively for RCPSP. Bulk of the focus has been on the operators for recombination and diversity. But relatively (but-significantly) less effort has been spared for the ‘triggers’, specifically for solution selection and termination criteria.

This present work attempts to address the RCPSP by adaptation of robust versions of the operators. The work proceeds to experiment with a few design models of triggers for expansion of the knowledge domain. In doing so, some parameters that affect the Genetic Algorithm would be fine-tuned to the extent feasible.

Chapter Five

The Proposed Algorithm

In this Chapter, we present adaptation of different components of Genetic Algorithms for the proposed algorithm. Both types of components are described – those adapted from literature, and the proposed. The Chapter proceeds as per flow of Genetic Algorithm.

5.1 Introduction

Genetic Algorithms by itself have been evolving over time and applications. It by itself has been proving the dictum on which it is based – *Survival of the Fittest*.

Researchers have suggested a number of modifications and adaptations. The different versions of Genetic Algorithm used presently, and along with their components (operators and triggers), are being subjected to modifications and evolutions.

In our work, we shall be using proven and pertinent operators. But as mentioned in an earlier Chapter, the triggers are generally a neglected lot since they are (apparently) less ‘visible’ in the Genetic Algorithms. The triggers will be designed fresh for incorporation in our (proposed) Genetic Algorithm. In short, we shall be exploiting the best operators on offer and suggest new triggers as per requirement.

In this Chapter, we discuss the adaptation of established components and suggestions proposed by our work for some of triggers.

For designing a Genetic Algorithm, there are three angles of study :

- a) Genetic Algorithm’s exploration and exploitation possibility,
- b) Convergence (efficiency and effectiveness) and diversity of the population, and
- c) Nature of the ‘problem’ for which it is being designed.

The pertinent information which plays vital role in arriving at the solution for the RCPSP are a) the tasks (for a feasible sequence), b) the task durations (for start/completion time), and c) the resource requirements (for resource allocation). Once these are made available now it is upto the proposed model how they are exploited to arrive at the optimum schedule, i.e.

- a) feasible sequence without violating the precedence constraints,
- b) with least tardiness within the time budget, and
- c) optimized allocation of resource(s) to the tasks.

The solution therefore has to carry with it the sequence of tasks and the start time of each task, assuming that the resource requirement is not diluted for any task. At every stage of the algorithm it becomes vital that these are not compromised.

5.2 Encoding and representation of solution

Hartmann(2002) observed that for many optimization problems, Genetic Algorithm don't operate directly on the solutions for the problems. Instead, they make use of problem specific representations of the solutions. The genetic operators modify the representation, which is then transformed into a solution by means of a so-called decoding procedure. It is therefore imperative that the chromosome representation be done appropriately.

5.2.1 Basic Representation

In general, there are mainly two chromosome encoding methods for representing the sequence of a set of numbers, as enumerated by Jean (1996): a) Adjacency, and b) *Path Representation, with their own set of genetic operators.*

a) Adjacency Representation is designed to facilitate the manipulation of 'edges'. The crossover operators based on this representation generate offspring that inherit most of their edges from the parent chromosomes. Adjacency representation is not very supportive of classical crossover operators, like one-point or two-point crossover. When used in RCPS, Adjacency representation might create 'illegal' solution (or impossible schedule), which straightaway removes it from option list.

b) Path Representation is the representation of a schedule, and is the favoured and simplest method for use with RCPSP. The schedule 5-1-7-8-9-4-6-2-3 is represented simply as [5 1 7 8 9 4 6 2 3].

Genetic Algorithm being a very iterative process, the demand on processor time (rather than on memory as with some other algorithms) would be high. Therefore any operation that taxes the processor would have to be properly justified.

We proceed by real-value encoding, where the task number by itself would be the value of any specified allele. The representation is carried out in two stages as described here.

We chose to define our chromosome as value (integer) coded path representation. The basic chromosome for a solution of the project with 'n' tasks where each task is designated as T_m , for $m=1$ to n , would be as depicted in Figure 5.1

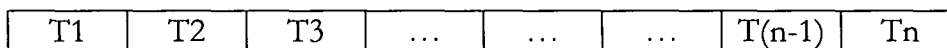


Figure 5.1 : Basic Representation of the Chromosome

For example, the schedule 5-1-7-8-9-4-6-2-3 is represented simply as in Figure 5.2.

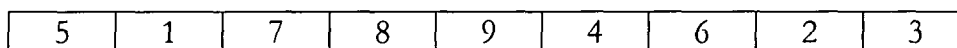


Figure 5.2: Example of a Basic Representation of the Chromosome

5.2.2 Final Representation

In addition to the sequence, our representation would have two more alleles, which would carry additional genetic characteristics.

The first additional allele would be based on its own properties, and carry information regarding fitness value or ‘own strength’ of the solution. It is a number generated by combination of two characteristics: a) the sequence of task, and b) the duration of the project. In essence, we are generating the fitness value as soon as the solution is generated, and allowing it to be carried by the chromosome with itself. We have termed this as ‘*UnoSign*’ – short for ‘Unique Number Signature’ as this is expected to be exclusive for each distinct solution.

The second allele would come into existence once the solution becomes part of a population. This would reflect number of clones or ‘relative strength’ of the solution. This allele is termed as ‘*Copy*’, as it would indicate the number of copies of itself available for mating.

Incorporating these two additional alleles, the actual representation of our solution would therefore have $(n+2)$ alleles, as depicted in Figure 5.3.

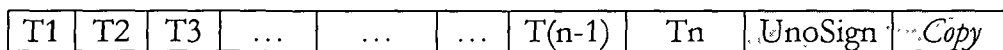


Figure 5.3: Final Representation of the Chromosome

The first n positions are input information. The $(n+1)$ and $(n+2)^{\text{th}}$ positions are calculated information of each individual solution. Detailed discussions of the methodologies for producing these two alleles as well as their specific usage are discussed elsewhere in this Chapter.

5.3 Initial Population

The initial population is of high significance for a Genetic Algorithm since it provides a starting platform for the algorithm. This is a set of feasible solutions that forms the primitive parent group to generate the first of 'next' generation. The size of this population is generally kept a relatively low, either as a predetermined number or as a (problem generated, program controlled) proportion of possible solutions. The former is practiced in most cases as it is easy to control.

For our actual testing we keep initial population number at 50 (fifty). These will also be the population size of every subsequent population.

The initial solutions are generated by forming feasible solutions that conform to the precedence constraints of the project. The precedence constraints are received from the Project input information. In addition to precedence constraints, the input would contain information about resource availability, resource requirement(s) and duration of tasks.

From the precedence constraints we generate the 'sequence', and get unconstrained project duration. Upon imposition of resource constraint information by a appropriately selected Schedule Generation Scheme (SGS), we get the 'schedule' – the project under constraint – where tardiness comes into effect.

In the present work, we devise a mechanism for checking the correctness or strength using *UnoSign* and *Copy*. For this, we combine these two calculated data to get the 'fitness value' of the solution, which then goes for usage by different segments of the Genetic Algorithm.

5.4 Fitness Function

The 'fitness value' of each solution provides the 'own strength' of that solution. This will decide the fate of the solution whether or not it will survive to advance into next generation. In the single mode RCPSP, the fitness value normally equals the makespan of the project.

Peteghem and Vanhoucke (2008) observed that a good Fitness Function gives appropriate feedback to the Genetic Algorithm. Hartmann (2001) defined Fitness Function for a solution as a function of the Random key, error distance and upper bound of the Project's makespan, which is given by the sum of the maximal durations of the activity. Jozefowska and Zimniak (2004) examined the differences between a Fitness Function with penalty, and one without. In most of the studies, infeasible solution demands a penalty factor – rather than rewarding the algorithm for generating feasible solutions at every attempt.

The present work devises a Fitness Function that rewards each individual, and respects computation effort of the algorithm. We have developed a simple yet effective and efficient algorithm to calculate the fitness value by combining quantitative factor with a qualitative aspect based on logical reasoning.

The most obvious fitness value for a schedule would be a function of 'tardiness' level – lower the tardiness, higher the 'strength' of the solution to go into the next generation and/or be selected for mating. A schedule that has the least tardiness is obviously the best solution.

However when we are operating in a large search space there is an equally obvious catch – more than one solution might have the same tardiness value. This is a situation where quantitative decision parameter values are exactly the same, but nevertheless we have to select from that subset. In such a circumstances we need another measure to resolve the dilemma when only one schedule will get preference. Rather than placing the decision on some other pure quantitative

parameter, we introduce a qualitative decision but which is based on the ‘numbers’ brought in as project information.

A solution (or sequence) contains parallel tasks within its linear string. Let us consider an example. In isolation of other tasks, let us consider three tasks – Task A (*foundation of plant*), and two of its immediate successors, Task B (*place order for machinery*) and Task C (*send engineers for training*). When sequencing, it is immaterial whether the three are placed as A-B-C or A-C-B, since in either case it will be the same project duration satisfying precedence constraints. While making the project description, it was immaterial either that they were placed in the order B written before C.

However, some judgmental factors had prompted the Project Designer to intuitively label the tasks in the order of A, B and C. According to work requirements, B and C can proceed in parallel. But his experience had made the designer to position ‘*order for machinery*’ before ‘*send engineers for training*’ in the Task list. Let this be termed as ‘design judgment’ parameter – which we decided to honour and utilize for building the fitness value. In deference to that logical reasoning we can believe that, *ceteris paribus*, schedule A-B-C is a better option and therefore ‘stronger’ than schedule A-C-B.

We proceed to design the Fitness Function as a combination of ‘tardiness’ and the ‘design judgment’ parameter. The former is objective, and has a demonstrable quantitative value. The latter is subjective, but we attempt to convert that into an objective value.

The sequence of the schedule is broken up (notionally) into a number of segments – each with equal number of tasks. For instance, if the project has 20 tasks, we may segregate them into blocks of five tasks, i.e. into four segments. A consistent algorithm converts each ‘segment’ into a ‘number’. These numbers are then added up to arrive at a total sum, which would act as the unique signature of the

sequence. In effect, we arrive at a reflection of judgmental preferences towards the (preferred) parallel tasks. For our experimentations, this number has demonstrated its ability to be (empirically) unique for each schedule with a high level of compliance.

For illustrating the technique, consider a project with twelve tasks, labeled 1, 2, ... 11, 12, with the unconstrained project duration, 'D_u', assumed to be 30 days. We take up two (feasible) solutions [#A] 1-2-3-4-5-6-7-8-9-10-11-12 and [#B]1-2-4-5-7-8-3-6-11-9-10-12. Only one of these would have to be selected, as equal opportunity is not being encouraged. The resource-constrained duration, 'D_c', of the project in both cases is (assumed to be) 35 days – which is the reason for dilemma. The tardiness measure is 5 days. The proposed algorithm attempts to resolve this dilemma by taking cues from the activity sequence.

We explain the algorithm of the Fitness Function in four stages.

Stage One :

The sequence is segregated into segments of four tasks (t=4) each. For [#A], we get (1-2-3-4), (5-6-7-8) and (9-10-11-12). The 'number' associated with each segment of the first sequence is calculated as:

$$\begin{aligned}
 1 \times 10^3 + 2 \times 10^2 + 3 \times 10^1 + 4 \times 10^0 &= 1234, \\
 5 \times 10^3 + 6 \times 10^2 + 7 \times 10^1 + 8 \times 10^0 &= 5678, \quad \text{and} \\
 9 \times 10^3 + 10 \times 10^2 + 11 \times 10^1 + 12 \times 10^0 &= 10122
 \end{aligned}$$

Similarly, for [#B], the segments will be (1-2-4-5), (7-8-3-6) and (11-9-10-12). And the 'number associated with the segments are :

$$\begin{aligned}
 1 \times 10^3 + 2 \times 10^2 + 4 \times 10^1 + 5 \times 10^0 &= 1245, \\
 7 \times 10^3 + 8 \times 10^2 + 3 \times 10^1 + 6 \times 10^0 &= 7836, \quad \text{and}
 \end{aligned}$$

$$11 \times 10^3 + 9 \times 10^2 + 10 \times 10^1 + 12 \times 10^0 = 12012$$

The sum of the three numbers derived of the three segments for [#A] is 17035. Similarly, for sequence [#B], the sum is 21093.

On comparison, we find that the 'logically' favourable sequence [#A] has a lower number associated with it. In our example above, the unbroken sequence [#A] is 'logically favourable' as compared to the sequence that has a broken continuation[#B]. (Theoretical and experimental proof for validating (or negating) this methodology is kept open for future work.)

This algorithm for generating the 'number' is summarized as :

$$N = \sum N_S \quad \text{OR}$$

$$= \sum N_S * 10^{\#1} \quad \text{where } S \text{ goes from } (1 \text{ to } n/g)^{\#2}$$

$$N_S = \sum [T_{(s,i)} * b^{(i-1)}] \quad \text{where } i \text{ goes from } (g \text{ to } 1)$$

Here,

N : the 'number' for the current sequence

n : number of tasks in the project

g : segment size

S : number of segments (= n/g)

N_S : 'number' for the current segment

i : the position of a task within a segment #3

T_(s,i) : the ith task of the Sth segment

b : the 'base' used for converting (*base10 system in our case*)

- #1 multiply by 10 to have an additional digit at the end, whose purpose will be discussed shortly. This operation is kept optional.
- #2 in case the last segment is a fraction of 'g', then the voids are filled up with zero(s) and treated as a full segment
- #3 the task at the leftmost position of the segment receives highest 'i', that goes on decreasing for subsequent task.

...Relationship 5.1

Care is taken to ensure that all such 'number' generated for each individual sequence is of same length, L, if needed be by adding zero(s) at the end. In the above example, L is six.

Stage Two (optional):

A modification is made to N to overcome the remote possibility of two (or more) schedules of equal D_c having equal N. The modification is made in the last digit of N (kept zero by default) utilizing the algorithm depicted as Relationship 5.2. In case this stage is not used, this last digit doesn't exist in the resultant.

If $[N \& D_c](S1) = [N \& D_c](S2)$

Calculate 'N_c', of each schedule

N_c, or 'curtailed N' is calculated by deducting the 'number' for the last segment from N

While $N_c(S1) = N_c(S2)$

Calculate subsequent N_c by advancing (one) more segment(s)

When $N_c(S1) \neq N_c(S2)$

Add 1 to the higher N #4

(i.e. change the last digit of the original N from zero to one)

#4 in case of comparison between more than two schedules, the last digits will be 0, 1, 2, and so on

...*Relationship 5.2*

Stage Two is kept optional, and is to be used if it is felt (or proved) that the remote possibility becomes actual. Mathematical veracity of this possibility is beyond scope of the present work.

Stage Three:

D_c of the current schedule is attached (or concatenated) to the front of the N to get the 'complete number'. In our example, if we employ Stage Two, the 'complete number' for the first sequence is 35170350 and for the second it is 35210930. We term this 'total number' as "*UnoSign*", and is the fitness value of a schedule.

Stage Four:

Finally "lower is better" is used for comparison between two schedules. Since our Genetic Algorithm produces feasible solution at every computational effort, therefore the question of fitness for a single individual in isolation is meaningless. The degree of fitness when a pair is compared is the relevant information.

To sum up, our Fitness Function is defined as

$[Lower\ is\ Better(UnoSign)]$

...*Relationship 5.3*

Each solution carries within itself this individual fitness value as one of its genes at the $(n + 1)^{th}$ position, where 'n' is the number of tasks.

5.5 Elites, or ‘Solution Set’

As mentioned in a previous Chapter, ‘elitism’ is the technique of skimming off the ‘best solution(s)’ or ‘elites’.

We use this method for retaining five elites, but after retaining the copies of the solutions in the population. This set is retained as the ‘current best solution set’ after generating a present population. Upon ‘termination’ of the Genetic Algorithm run, this set is presented to the Project Manager from which he may select the schedule of his choice.

Each solution would be having the least possible tardiness – preferably the ideal situation of zero tardiness. However each solution differs in the sequence of tasks. The Project Manager then has the liberty of applying his judgment and any other criteria of selecting the sequence for implementation.

5.6 Population size

The population size (we term this as *PopSize* for implementation) of every generation, including the initial population, is kept constant throughout the process. During processing within one generation, *PopSize* number of offspring are placed in a ‘notional next’ population.

For choice of ‘fittest’ solutions to allow them into next generation, we obtain a pool of double the size of *PopSize* – ‘parent’ set plus ‘offspring’ set – both of equal size. Note that elites are already a subset of ‘parent’ set. Thereafter *PopSize* numbers of ‘fittest’ solutions from the combined set is sent to ‘next’ generation. From the (presently) unfit solution set, ‘forefather(s)’ are retained (*mummified*) for possible inclusion into subsequent generation. And remaining unfit ones are discarded.

5.7 Clones

Within the current generation, the 'strength' of individual solution varies. As is with Nature, the 'stronger' individual is given higher number of chances for mating. This may be considered analogous to polyandry/polygamy, as well as multiple offspring from same mating pair.

The 'number of chance' is made possible by (notionally) creating 'clones' of the parent population members. We have termed this indication of the number of clones as '*Copy*'.

By default this value is made one or any positive integer, depending on an experimental parameter called '*Clone Factor*'. Stronger solutions would have a higher '*Copy*' value. In the population, every time a parent is 'successfully' selected, its *Copy* value is reduced by one and is eligible for selection till its *Copy* is not zero.

We formulate an algorithm for *Copy* by reworking Alcaraz and Maroto(2001)'s adaptation of Remainder Stochastic Sampling Without Replacement, which they employed to reduce stochastic errors associated with Roulette Wheel Selection.

$$\text{Copy}_i = \tau + C$$

where $\tau = \text{INT} \left[\frac{(S_w + 1) - S_i}{\sum [(S_w + 1) - S_j]} * P \right]$
here j goes from (0 to P)

Here,

Copy_i is '*Copy*' value of i^{th} solution within current population, and is calculated by taking the integer portion of τ

S_w is makespan of 'worst' solution within current population,

S is makespan of individual solution,

P is population size,

C is a '*Clone Factor*', a positive integer specified between 1 and 5

...Relationship 5.4

'*Copy*' is carried with the chromosomes at the $(n+2)^{\text{th}}$ position.

5.8 Selection

Selection is one of the most significant operators of Genetic Algorithm. We have placed it under the ‘triggers’ category, as it does not directly generate offspring. Within a population, Selection triggers the choice of a parent for mating.

Selection has to be meticulously carried out; otherwise the algorithm would degrade into a random-search methodology. We have devised a simple methodology of selection by assembling together acceptable high-quality characteristics from different Selection methodologies.

Quality or strength of a parent in our algorithm is decided by *UnoSign* value in its chromosome, where lower this factor, better is an individual. Using *UnoSign*, we sort the parent population. From this sorted lot, any parent is selected by random hit. This we mate with another parent whose *UnoSign* factor is higher than itself.

In doing so we select a ‘*better spouse*’ for the current parent to mate with. For obvious reason, the first parent is selected from the (truncated) set of population that (notionally) excludes the ‘best’ feasible individual.

Prior to selection of either parent, the algorithm has to ascertain that the individual chosen for possible ‘selection’ have a positive *Copy* value.

We term this selection algorithm as ‘*Better-Spouse* Selection’.

Better-Spouse Selection makes sure that a selected parent definitely mates with a stronger mate.

(To avoid any bias, we refrain from referring the ‘Better Spouse’ either as Father or as Mother).

5.9 Crossover

Crossover has been considered by many researchers as the most significant of genetic operators. We make use of a robust crossover techniques suggested for the RCPSP – the Precedence Set Crossover (PSX) technique. In this crossover genetic characteristics from both the parents are carried over to offspring without violating precedence constraints – a condition of utmost importance while scheduling a Project. We discuss the PSX technique as adopted for our algorithm.

5.9.1 The Precedence-Set Crossover

The PSX we describe here is an adaptation of the single-point crossover technique (let us term it PSX1). With further modification(s), it can be adapted as a two-point (PSX2) or multi-point crossover (PSXm).

PSX allows Offspring1 to inherit relative positions of tasks that belong to a (functionally derived) ‘set’ from Parent1, and rest of the tasks (the tasks which do not appear in the ‘set’) from Parent2. Offspring2 will inherit relative positions of tasks in the set from Parent2, and rest of the tasks from Parent1.

Given a (random) task j , the ‘set’ is created by placing the task j , its predecessors (not necessarily immediate), and its successors (not necessarily immediate) to the initially empty set. We depict a small project as Figure 5.4

Methodology for generating two offspring from a pair of (eligible) parents is represented in Figure 5.5. The first step consists of selecting a project task in a random way. In this example let task 7 be the selected task.

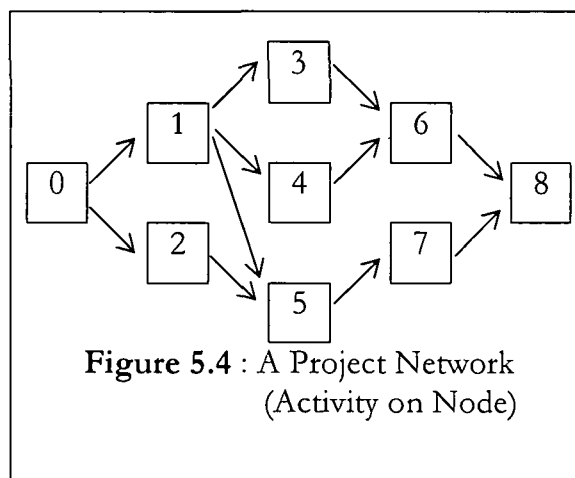
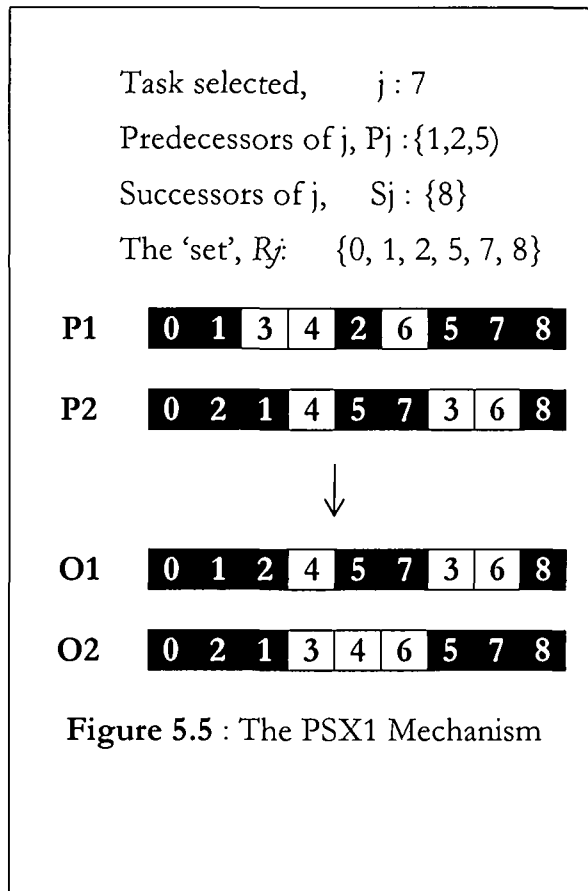


Figure 5.4 : A Project Network
(Activity on Node)

Then, precedence set R_j is constructed, as $R_j = \{1, 2, 5, 7, 8\}$. We have shaded these tasks in the parents. (This is the ‘functionally derived’ set we mentioned earlier)

Offspring1 must inherit positions of tasks in R_j with their relative order in Parent1’s sequence, and rest of the tasks, with their relative order in Parent2’s sequence. For generating Offspring1 we go through the Parent2’s sequence, task by task. When we find a task, before drawing it in Offspring1, we must be sure that it preserves relative order of shaded tasks in Parent1’s sequence (if it is a shaded task) or relative order of unshaded tasks in Parent2 (if it is an unshaded task).



The first task in the Parent2’s sequence is task 2, which is a shaded task, and therefore it must preserve relative order of shaded tasks in Parent1’s sequence. As in Parent1, task 2 appears after task1, in Offspring1 this order must be preserved, so first we draw task 1 and then task 2. The next task in the Parent2 is task 1, which has already been drawn in Offspring1. Next, task 4 is not a shaded task, so it can directly be drawn in Offspring1. Task 5 coming after that is a shaded task and tasks 1 and 2 must appear before it. As these tasks have already been drawn in Offspring1, task 5 can directly be drawn in Offspring1 sequence.

We follow this procedure until Offspring1’s sequence is completed. We can observe that in Offspring1, tasks 1, 2, 5, 7 and 8 preserve their relative order in

Parent1's sequence, and unshaded tasks – 3, 4 and 6 preserve their relative positions in Parent2's sequence. The precedence relations are fulfilled Offspring1.

To construct Offspring2, the procedure is the same, but now it will inherit positions of shaded tasks with their relative order in Parent2's sequence, and rest of tasks, with their relative positions in Parent1's sequence, so that Offspring2 is also a precedence feasible solution.

At the end of PSX we have a group of four solutions to choose from. Crossovers are carried out till

- a) offspring are generated equal in number to Population size (or its multiple, or some other predetermined number), or
- b) all the parents finish their '*Copy*' value –

whichever is earlier.

We have kept this 'offspring limit' to be equal to *PopSize*. From this pool of [2 X *PopSize*] solutions we copy off the elites and select *PopSize* number of individuals for the 'next' generation. These two pick-ups are made on the basis of fitness, i.e. those with lower '*UnoSign*' value are picked up.

5.9.2 Crossover Points

Having placed the mechanism of PSX, we experiment with this crossover technique on two modes:

- a) Mid-Point crossover, where the two spouse mates at their respective mid-point as the crossover point to produce one pair of offspring. This may be analogous to the '*one parent – one (pair of) child*' policy.
- b) Random-Point crossover, where the 'stronger' spouse mates multiple times with the same 'weaker' spouse, on different crossover points to

produce (possibly) more than one offspring. This may be analogous to the ‘one parents, multiple (number of) child’. The number of time they mate is decided by *Copy* value of the ‘better spouse’.

5.9.3 Building Blocks

For producing ‘better’ offspring, many researchers use the ‘building blocks’ mechanism. They provide a faster convergence.

We restrain our algorithm from using this methodology. Since the search space has dramatic variation – depending on the test problem – there is a possibility of false and premature convergence if building blocks are utilized.

Moreover, the converse of this mechanism is favoured for scheduling problems, especially if it is a tightly resource constrained situation. Building block mechanism is favoured if we are dealing at identifying patterns. But in our study area we deliberately break down patterns so that sampling is evenly but randomly distributed over the search space.

5.10 Intrusions

Genetic Algorithm advocates introduction of (sudden) diversity in a population to seek variation in search locality. We shall make use of ‘immigration’ and ‘dormant-forefather’ as two methods of introducing such diversity.

5.10.1 Immigration

The ‘alien’ is a solution freshly generated by the same algorithm as was used to generate members of the initial population. This alien would immigrate if entry is permitted into the ‘current’ population. The analogy of this mechanism is trans-border movement of population (e.g. people coming into a new country for

permanent residency). History has ample proof that immigrant population does bring in fresh gene pool. But this is a dangerous proposition as there is every possibility of worsening of the situation. The 'alien' has to prove its worthiness to be allowed the status of an 'immigrant'. This would be possible if two probabilities are in their favour

- a) a (very low) probability for generation of the alien, and
- b) a (even lower) probability for permission to allow infiltration

As is evident, vector multiplication of these two probabilities therefore makes the occurrence of an alien infiltration a very low possibility. Nevertheless, this technique would bring in sudden diversity to the population. But if the two probabilities are kept high, the search gets relegated into a simple fresh generating mechanism.

The first low probability decides whether or not to generate the alien. In case of favourable probability, a new individual – the alien – is generated. The alien now tries to infiltrate into the current population by identifying its probable position. This position is identified as any single individual who's *UnoSign* is lower than itself, or the weakest of the individuals within the population.

But the '*alien*' has to overcome yet another low probability. In case of favourable probability again, the alien replaces the identified individual of the population. The status of the '*alien*' is now converted into '*immigrant*' – a fact that has no further bearing, as it is considered at par with any other individual of the population.

We are favouring the '*immigrant*' method of diversity as it involves relatively less computational effort. A sequence depends entirely on precedence constraints. Mutation by means of (random) switching or alteration of genes (or alleles) usually results in performing a high number of backtracking and precedence checks to prove the correctness of a sequence. With lower amount of computational effort we generate a perfectly correct sequence, exploiting an already proven algorithm.

Many researchers use mutation, but discard the mutant if it violates constraint(s) – thereby laying waste the computational efforts.

5.10.2 Dormant-Forefather

We devise and experiment with use of another mechanism – ‘*dormant-forefather*’, for introducing diversity. This may be considered as an analogy to the *Egyptian Mummy* theory where the ancient Egyptians *mummified* the bodies of the departed. They believed that one day in the future these (strong and important) ‘people’ would be brought back to life into a civilization when science and technology is far improved. And they would get a fresh lease of life to perform all normal activities.

In many cases it is possible that strong genetic material that might otherwise have been lost would bring positive diversity to a much later generation, when the nearby optima is favourable and is possibly not a premature one. As explained for alien immigration, the probability factor for dormant-forefather immigration also needs to be kept low.

The concept is shown in Figure 5.6. Here solutions [1], [2], and [3] have higher fitness value, but are converging prematurely onto sub-optimal area [A] and [C]. But solution [4], which is in correct alignment with the optimal solution area [B], gets dominated by the other three since it is having a lower fitness value. At the time of survival of the fittest, point [4] gets eliminated. This means that we lose a potentially strong parent, but whose time has not yet come.

However once in a while a (randomly selected) solution may be temporarily stored, and be made a candidate for infiltration attempt in a much later generation. In Figure 5.6, as soon as solutions that start climbing up area [B] are visible in a generation, we might gain by reviving solution [4]. This is the concept behind our proposed *Egyptian Mummy* immigrant or *Dormant-Forefather* immigrant.

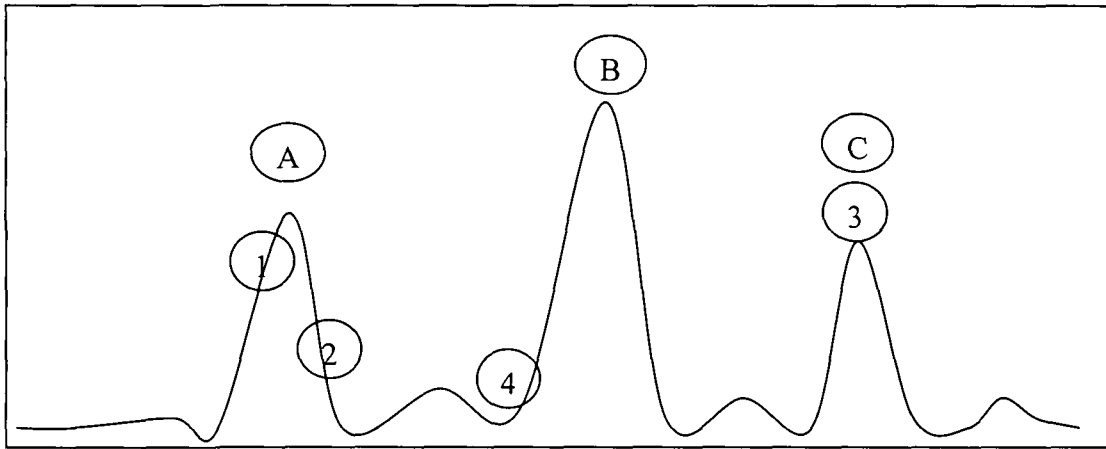


Figure 5.6 : Concept of the Dormant-Forefather (*Egyptian Mummy*)

One (or more) of the individuals of a generation who failed in the ‘survival of the fittest’ dictum would be considered for ‘*mummification*’. This individual is identified from the ‘ill-timed and unfit’ lot by a (pure or calculated) random function, if a low probability for doing so is triggered. If it overcomes that low probability, the individual is preserved for possible inclusion in a subsequent generation. The remaining ‘unfit’ individuals are discarded as per classical Genetic Algorithm norms. We term this mummified individual(s) as ‘*dormant-forefather*’.

While processing a subsequent generation of our Genetic Algorithm, at the juncture when the algorithm try for ‘*immigrant*’ infiltration, another low probability would trigger the possibility of revival of (one of) the ‘*dormant-forefather(s)*’.

If this probability factor is favourable, we revive (one of) the dormant-forefather and convert it’s status into that of an ‘*alien*’. Thereafter the possibility of infiltration, etc. is performed as was carried out for immigration.

The logic for ‘*dormant-forefather*’ approach is to recheck previously discarded search locations. It was possible that due to non-support from others, an individual of a

generation failed to hint at possible optima area. But at a later generation when we are more near optimization (*technologically and scientifically advanced !*), the dormant forefather can be revived to (possibly) speed up the process.

A Dormant Forefather would prove its worth upon revival if the search locality where the optimal lies is the locality from where the individual was originally discarded.

Other researchers have used the term '*forefather*' for their own usage in different context. Aporn Dewan, et al (2001) used it to denote a pair of individuals that act as forefather to a clan. In Genetic Algorithm literature, a clan refers to a set of strings which has a common trait (e.g. 1010, 1110, 1011 belongs to 1*1*). A clan is denoted by a probability vector, p' . The p' is a copy of vector p of which some $p[i]$ are randomly set to "0" or "1" according to the forefather.

Ogino, et al (2002) in their work for their Pedegree Analysis Programme tried to find out characteristics of a given strain data by going up to forefathers from a descendant.

5.11 Termination

To avoid a perpetual Genetic Algorithm, it has to be terminated once certain criteria are met. The most common criteria are discussed in another Chapter. We experimented termination based on different suggested methodologies, and have designed an Adaptive Termination algorithm.

5.11.1 Fixed Generations Termination

As the classical method of termination, we experimented with pre-specified number of generations. The Genetic Algorithm was allowed to continue till this maximum number of generations was processed irrespective of complexity of the

Project characteristics. Kolisch and Hartmann, (2006) have published results for the test data on 'fixed' number of schedules.

5.11.2 Fitness-Deviation Termination

The Genetic Algorithm was terminated as soon as fitness deviation of a population is within a pre-specified level (very small), and fitness deviation between successive generations also approach (another) pre-specified level (very small, possibly zero). *[In case we encounter divergent deviation between populations, a possible modification to our algorithm would be to increase the 'intrusion' probabilities.]*

5.11.3 Adaptive Termination

In deference to individual characteristics of each Project, we design an Adaptive Termination algorithm for termination.

Each Project has its own set of tasks and resource types. Moreover, Projects differ in their number of possible solutions, or the search space. This we term as Complexity level. The Complexity level of a Project increases by direct (linear or exponential function based) proportion to the precedence constraint of Project tasks. Keeping this in mind we designed an algorithm that would be adaptive to these three Project parameters for deciding the termination criteria. The adaptive algorithm is designed by evolving it to a final stage through gradual incorporation of parameters.

The Adaptive Termination algorithm is a function of three parameters, Project length or the number of tasks, number of resources under constraint, and complexity level. Combined, they arrive at *MaxGen*, as given in Relationship 5.5

$$G = f[(T, R), C]$$

where, G is number of generations, *MaxGen*

T is number of tasks,

R is number of constrained resources, and

C is Complexity level

...*Relationship 5.5*

We group up the first two factors and term the group as *Project Factor (P)*, and rework the last factor as *Complexity Factor (C)*. Thus as a general form, we get G as a function of P and C, or

$$G = f[P, C],$$

where $P = f[T, R]$

...*Relationship 5.6*

5.11.3.1 Adaptive Termination 1

Initially the algorithm is based on two assumptions –

- a. the Project is ‘not complex’ and hence doesn’t require much emphasis on its complexity level, and
- b. is dependent only on the Project Factors.

With such assumptions, we calculate G by leaving aside C , i.e. keeping it at 1, as given in Relationship 5.7

$$G = f[P], \text{ with } C = 1$$
$$\text{Where } P = f[T, R],$$

...*Relationship 5.7*

For example, if the project has 60 tasks and utilizes 4 types of constrained resources then the Genetic Algorithm is processed for 240 generations.

5.11.3.2 Adaptive Termination 2

Next we remove the assumption regarding 'Complexity' of the Project, and introduce C through combination of functions.

$$G = f[T, R, f'(\theta)]$$

where θ is the indicator of search space, i.e. the complexity level, and

$$C = f'(\theta)$$

...Relationship 5.8

The segment $f'(\theta)$ is combined with Relationship 5.7 to dampen the rate of change (or velocity) of complexity. This is the second stage of evolution of the proposed Adaptive Termination. Logarithmic functions act a good damper, and we propose to implement the same.

But this relationship has damped the complexity level to a very low level, almost making the rate of growth flat. The exponential nature of complexity is brought back and the Termination criteria allowed to accelerate, but under damping by a logarithmic function. Thus the algorithm is allowed to evolve further.

5.11.3.3 Adaptive Termination 3

We introduce a combination of functions for addressing more characteristics of project complexity. Exponential function is proposed to respect velocity, and Logarithmic function would try to control it, as depicted in Relationship 5.9

$$C = f'(\theta) = f''[\beta(\log(b, \theta, \phi))]$$

where

- β is the base of an exponential function for C, which is exponentially raised to a function of θ and ϕ ,
- b is base of the logarithmic operation and
- ϕ is a limiting number which depends on the search space.

... Relationship 5.9

The conceptual mechanism of this Damped-Acceleration due to combination of Logarithmic and Exponential function on an exponential set of data is illustrated in Figure 5.6.

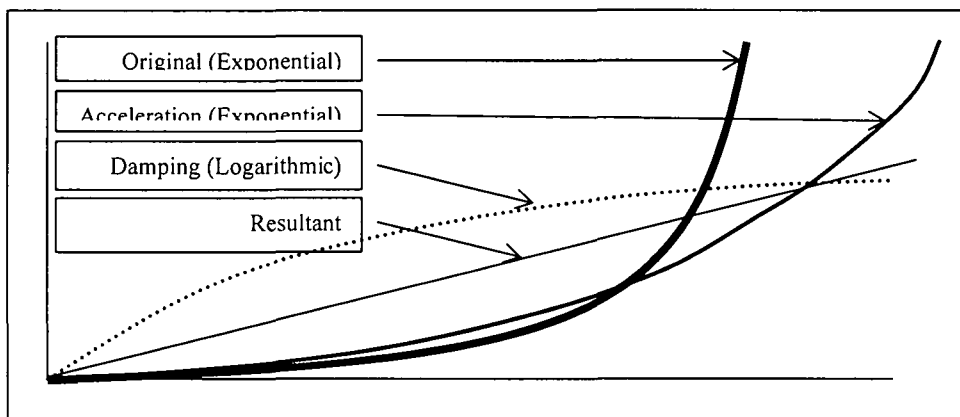


Figure 5.6 : The Damped-Acceleration mechanism

Implementation methodology of these Adaptive Termination algorithm relations is discussed in the next Chapter. We experimented by computational application of the algorithm, and mathematical proof is kept outside scope of our work.

5.12 Post Termination

Once the process terminates, the current set of 'elite' are presented as Result set to the Project Manager. Instead of presenting a single 'best' result, it is prudent to provide a set of 'good' results on which the Project Manager can apply qualitative judgment to arrive at the final decision.

Because our algorithm not permitting repetition within the elites – by employing *UnoSign* based sorting – the Result set would have unique alternative solutions.

The chromosome we developed does not carry the resource requirement, and start-finish time of activities. This was done to avoid transporting and processing a huge chromosome during program run.

Once the algorithm terminates, the program would run (a modified segment of) the scheduling algorithm on Result set individuals and provide pertinent output for the Project Manager.

Chapter Six

Implementation and Experimental Setup

In this Chapter, we describe implementation and experimentation, and analysis methodology of our work. The Chapter commences with a description about the platform for implementation, followed by a short description about data-set used for testing. After this, the Chapter is divided into two parts. Part A describes functions created as components of the program, and simultaneously elucidates tactics for conversion and adaptation of the (mathematical) relationships involved – as per flow of the algorithm. Part B describes the experimentation setup and parameter settings. We describe the Design of Experiment, where parameters taken up for tuning are formally charted. The Chapter concludes with a discussion of methodology of result analysis, and criteria for model validation.

6.1 A General Outline

Genetic Algorithm as implemented in our work is a blend of proven and proposed components. The proven components are collected from literature, and adapted for use by incorporating variations. As is done in most Genetic Algorithms, controlled use of Random Number has been done extensively.

For implementation we use commonly available platform and software mainly from convenience point of view. This also places us at par with most of other research work, which facilitates uncomplicated comparison.

6.1.1 Platform Description (hardware, software, etc)

Implementation of the algorithm is done in Structured C, and compiled with Borland®C++. In doing so, we exploited certain features of the compiler that is not typical of (Kernighan/Ritchie) C. For example, we incorporated features of C++ for file reading and writing.

The program is run on Intel Pentium4 machine of 2GHz speed with 512MB RAM under Microsoft Windows XP environment. Depending on data-set and parameters selected for testing and monitoring, run time for a full data-set ranged from under seven minutes (averaging 875 millisecond per instance) to just above twenty four hours (averaging three minutes per instance).

6.1.2 Input Information – The Test Data-Set

6.1.2.1 The Input Data-set

We tested our algorithm on internationally accepted standard benchmark instances provided by Kolisch and Sprecher (1996) for evaluation of scheduling techniques for the RCPSP. It is called PSPLIB, and is widely acknowledged in the literature for the purpose. As test instances, we employed the standard SMFF (*Single Mode, Full Factorial*) set of the PSPLIB. These are labeled as J30, J60, J90 and J120,

indicating the Project lengths. The sets J30 and J60 consist of (48 X 10 =) 480 project instances each, and we have used these two as our test data-set.

Each of the sets deals with four constrained renewable resources. Each task has one execution mode. This set is dependent on three parameters, roughly corresponding to the interconnectedness of the task dependencies, the number of resource types, and resource quantity available. Other test problems provided by the PSPLIB include SMCP, MMCP, and MMFF sets.

The comparison of performance between algorithms of different researchers who use this data-set is compiled regularly by the moderators of the library. We use the updated comparison of Kolish and Hartmann (2006) for benchmarking our experimental results. Here the authors of the paper have invited 'future studies' to make use of the compiled results for benchmarking. Previous comparison literature was made by Hartmann and Kolisch (2000) and Kolisch and Padman (1996).

In the same library, latest updated listing of best result on each instance set by different researchers is available. On our last access (October 2008), we located results of 2nd May, 2008 for J30, and 24th October, 2005 for J60 data-set.

6.1.2.2 Other Test Data-set

SMFF data-set of PSPLIB is the most widely used test data for the RCPSP. There are other standard data-sets within this library, as well as available from other libraries and authors which are used for benchmarking. A short description of these is provided here.

a) **PAT** : This relatively easy set of instances was introduced by James Patterson in his comparison of exact solution methods for resource constrained project scheduling. The Patterson set (PAT) consists of 110 project scheduling problems whose tasks require multiple resources and are defined with one execution mode.

The resource constraints are not very tight, and in many cases the optimal resource-constrained solution is the same as the resource-unconstrained solution.

b) **SMCP** : The Single Mode Ceteris Paribus set is similar to the Patterson set, but they range in size from 10 to 40 tasks and include more resource restrictions. The set includes 200 problems with 1 to 4 renewable resource types. Each task has only one execution mode.

c) **MMFF** : The Multi-Mode Full Factorial set consists of problems that include four resource types, two renewable and two non-renewable. Only about 85 percent of the instances in this set are known to have feasible solutions. The possibility of generating problems with no solution arises with the addition of non-renewable resources.

d) **BMRX** : The BenchMaRX problem was proposed by Barry Fox and Mark Ringer in early 1995. It is a single problem with 12 parts. Each part adds additional constraints or problem modifications that test various aspects of a solution method. The first four parts are fairly standard formulations. It gets harder from there. The problem is large: 575 tasks, 3 types of labor resources and 14 location-based resources. In addition to resource/location constraints, it includes many temporal restrictions such as three shifts per day with resources limited to certain shifts and task start/finish required within a shift or allowed to cross shifts. The last of the twelve parts includes multiple objectives. By varying resource availability and work orders after a schedule has been determined, the problem also tests the ability of solution methods to adapt to dynamic changes.

e) **Boctor** sets : Boctor has given a set of test data, which is termed as boctor50mm boctor100mm.

f) **Alvarez-Tamarit** sets : Three sets of test data has been designed by the authors, viz prob103, prob27 and prob51.

As noted earlier, we have experimented on the SMFF set proposed by Kolisch and Sprecher (1996). For further description regarding instance generation mechanism, the instance generator (*ProGen*), solution sets, etc we refer to the PSPLIB.

We depict the format of our test data (SMFF) as given in the PSPLIB in Figure 6.1. The Library have modified it from the ProGen format to the Patterson format.

```

Let us denote

j=1,...,J : jobs
r=1,...,R : resource types
S(j)      : number of immediate successor-jobs of job j
S(j,s)    : s-th immediate successor-job of job j
d(j)      : non-preemptable duration of job j
K(r)      : resource availability of resource type r within each
period
k(j,r)    : resource usage of job j w.r.t. resource type r

The format is:

J      R
K(1)   K(2)   ..   ..   K(R)
d(1)   k(1,1) ..   ..   k(1,R) S(1)   S(1,1) ..   S(1,S(1))
..
..
d(J)   k(J,1) ..   ..   k(J,R) S(J)   0

```

Figure 6.1: PSPLIB instance input format

Note that for a project with 30 activities, the instance size (j) is 32. The two additional tasks are the initial point and the conclusion point, which are dummy tasks. (For our purpose whenever we mention 'tasks', we would be referring to 'j', i.e. project size that includes the dummy activities.)

Based on the above format definition, an example of a test project instance of 32 tasks (i.e. 30 activities) is given in Figure 6.2. We shall refer to this instance for different components of our implementation.

6.1.2.3 Modifications for changing to other forms

Even though we used the SMFF data-set, but with a small modification of our program, we were able to test adaptability of our algorithm to accept other data-sets.

32	4							
12	13	4	12					
0	0	0	0	0	3	2	3	4
8	4	0	0	0	3	6	11	15
4	10	0	0	0	3	7	8	13
6	0	0	0	3	3	5	9	10
3	3	0	0	0	1	20		
8	0	0	0	8	1	30		
5	4	0	0	0	1	27		
9	0	1	0	0	3	12	19	27
2	6	0	0	0	1	14		
7	0	0	0	1	2	16	25	
9	0	5	0	0	2	20	26	
2	0	7	0	0	1	14		
6	4	0	0	0	2	17	18	
3	0	8	0	0	1	17		
9	3	0	0	0	1	25		
10	0	0	0	5	2	21	22	
6	0	0	0	8	1	22		
5	0	0	0	7	2	20	22	
3	0	1	0	0	2	24	29	
7	0	10	0	0	2	23	25	
2	0	0	0	6	1	28		
7	2	0	0	0	1	23		
2	3	0	0	0	1	24		
3	0	9	0	0	1	30		
3	4	0	0	0	1	30		
7	0	0	4	0	1	31		
8	0	0	0	7	1	28		
3	0	8	0	0	1	31		
7	0	7	0	0	1	32		
2	0	7	0	0	1	32		
2	0	0	2	0	1	32		
0	0	0	0	0	0			

Figure 6.2: Instance example of the J30 test data-set ; File name : J301_1.rcp

Each data-set consists of four components :

- a) Tasks (or activities or nodes, depending on the authors' preference of term),
- b) Resources (we focused on quantity constrained renewable resources),

- c) Precedence Constraints (which of predecessor or successor is depicted), and
- d) Completion time (of the task)

The major difference amongst input data-set lies in the position format of the above four components. Once this is analyzed and comprehended, and requisite change made in the 'input information' function, the remainder of our program remains the same. However at all times, we concentrated on the Single Mode aspect.

Part A : Implementation

6.2 Encoding and Representation of Chromosome

Project schedule has three generic components, viz.

- a) the Spatial component, that denotes position of a task in the chromosome,
- b) the Temporal component, that denotes (combination of) Start-time, Finish time and Duration, and
- c) the Resource component, that denotes resource requirement(s).

In our representation, the Spatial component is given priority since we focus our study on total makespan of the sequence. The Temporal and Resource components are not carried with the chromosomes. If needed be, with a minor modification these components can be incorporated, as was tested for viability. Reducing chromosome size lessened burden on computational resource especially when project parameters are on a higher side. Task duration and resource requirement being constant throughout processing for a Project, we store them only once and refer to that as and when required.

The chromosome is represented as a 'struct' data-type of C, with two distinct portions,

- a) a single dimension integer array of length equal to number of tasks to take in spatial information. Extending to multiple dimensions would allow the array to integrate temporal information as well as Resource requirement (and allocation) information of each task.
- b) two additional genes, one to hold the Fitness Value (*'UnoSign'*) and the second to hold the number of clones (*'Copy'*). These two genetic factors are described in the previous Chapter.

The length of the chromosome is programmed as (pseudo) adaptive, depending on the size of project under test. When we run on a project whose length is 'n', size of the chromosome becomes 'n+2'. Thus, when we test the J30 data-set which has 32 tasks (30 activities, plus two dummies at the end), our chromosome will have 34 genes. Upon modifying the program for running the J60 set, the size of chromosome would change to 64.

6.3 Initial Population

We use the same population size for initial population as well as for subsequent generations. A member of the initial population (sequence) is generated using a conventional methodology by ignoring resource constraints but in total deference of precedence constraints.

6.3.1 Sequence Generation

Individuals of the initial population is generated by a method of 'peers and successors', using a serial algorithm. The sequence is generated by the algorithm as described in Figure 6.1 :

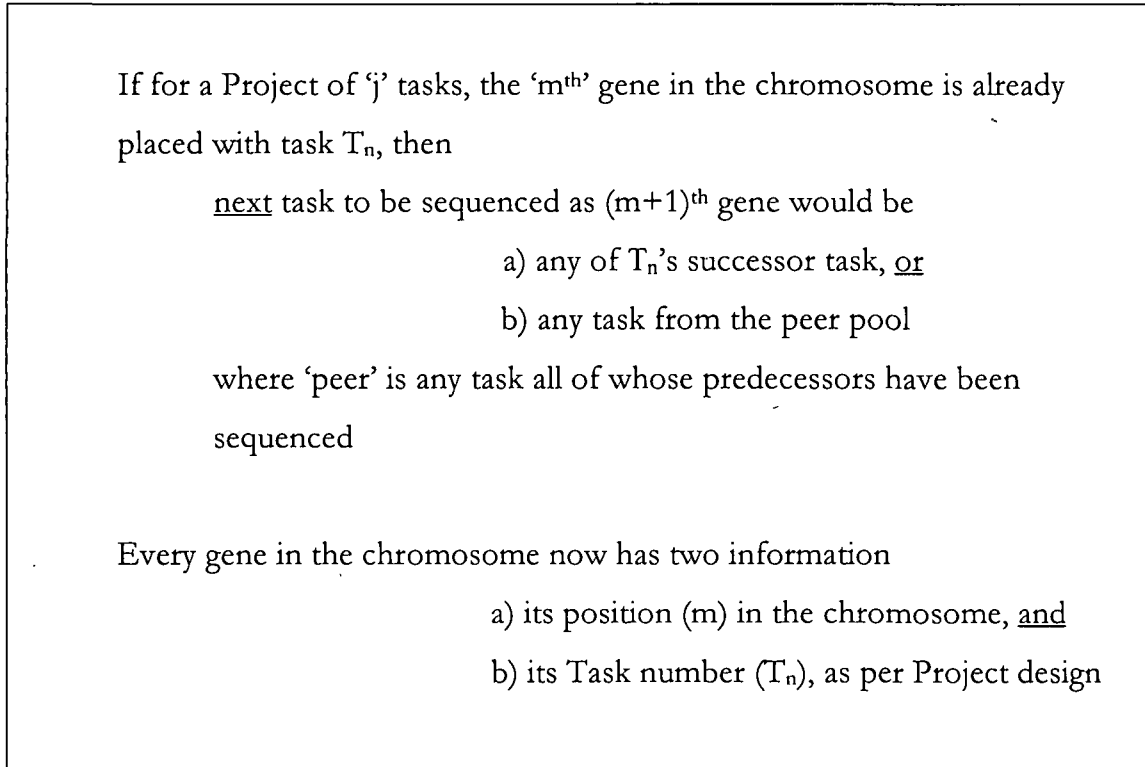


Figure 6.3 : The Algorithm for Sequence Generation

We proceed to fill up the chromosome serially from first position till nth position, with task selected at random but according to the above algorithm. This continues till all but two last genes are filled up. Obviously T_1 is the initial (dummy) task of the project, and T_j is the final (dummy) task. The last two positions are filled up after calculation of *UnoSign* and *Copy*.

6.3.2 Unconstrained makespan

The individuals sequences of the initial population being unconstrained sequence would have a makespan of ideal and minimum duration. The Finish-Time of T_j is the unconstrained makespan. Next we proceed to allocate resource to the tasks.

6.4 Schedule Generation Scheme

Allocation of constrained resource to a Project sequence generates the ‘Schedule’ for a specific sequence. We assume that partial allocation of resource is not permitted and tasks are of non-preemptable duration

There are two basic Schedule Generation Schemes (SGS) – Serial and Parallel. We favour the Serial SGS for adaptation into our algorithm. Hartmann(2002) pointed out that activity list representation (i.e. sequencing) together with Serial SGS (i.e. scheduling) leads to better results than other representation of the RCPSP.

For implementing the Serial SGS, an algorithm is devised that scans the already scheduled ‘stub’ (segment of the chromosome from first position) for locating position for the task at hand. This we term as ‘*SweepCreep*’ algorithm, since the task ‘sweeps’ for a slot, and if not possible to be scheduled, it ‘creeps’ to a subsequent slot. The *SweepCreep* algorithm is summarized in Figure 6.2.

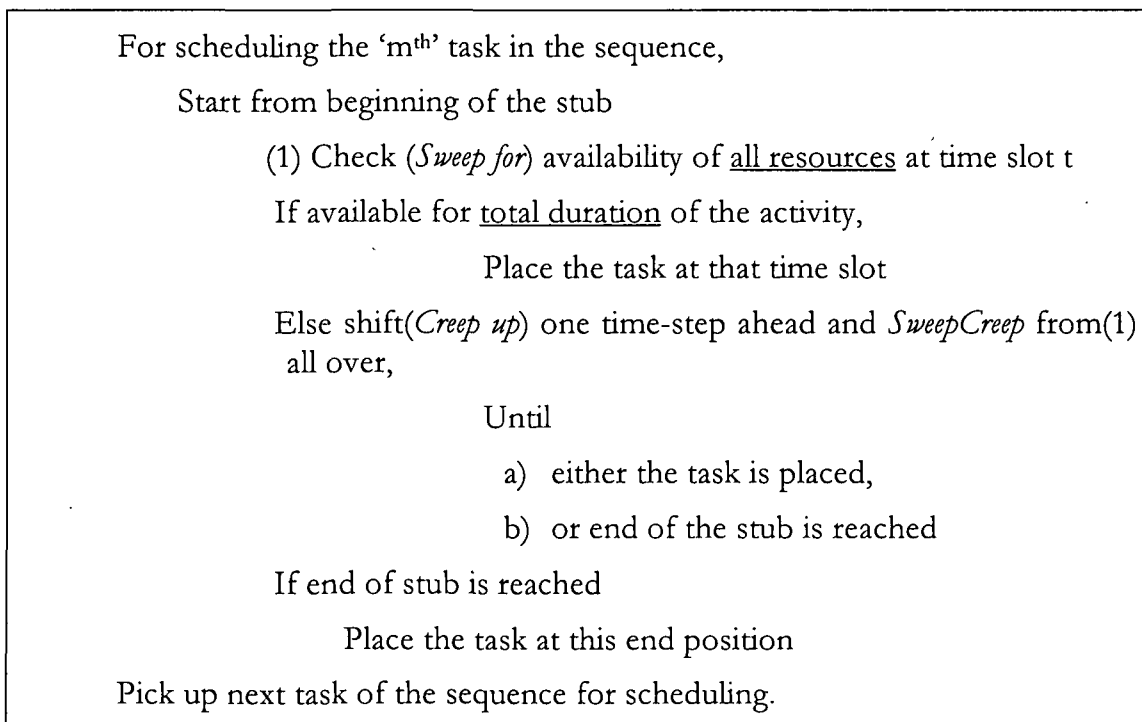


Figure 6.4 : The Sweep-Creep Algorithm

As mentioned earlier, resource requirement and duration of each task is placed separately, and is brought in for the pertinent task. The time window is temporarily created during generation of each schedule. From the time window for the last task, we get the (resource constrained) makespan of the current schedule, which is next utilized for calculating Fitness Value of the current schedule.

The Finish-Time of the last task, T_j , is now the resource-constrained makespan of the schedule. Henceforth, all reference to makespan would indicate the resource-constrained makespan, if not otherwise stated.

6.5 Fitness Function

To illustrate calculation of the Fitness Value, *UnoSign*, we take example of three (precedence feasible) schedules of the Project depicted in Table 6.1 alongwith their (resource constrained) makespan. The unconstrained makespan (or the unconstrained Critical Duration) of the Project is calculated to be 38, and optimum makespan (post allocation of constrained resources) is 43, according to results available at PSPLIB.

Schedule No	Sequence of the Schedule	Resource Constrained Makespan
#A	1-4-3-13-18-10-9-8-7-27-12-5-2-16-15-21-6-19-11-14-17-22-29-26-20-25-28-23-24-31-30-32	47 days
#B	1-3-8-4-5-19-12-13-9-7-18-29-10-14-17-2-27-11-16-6-15-20-21-25-28-26-31-22-23-24-30-32	47 days
#C	1-3-8-13-12-18-7-19-29-2-4-5-15-9-14-27-17-6-10-16-22-21-11-20-25-23-24-28-30-26-31-32	49 days

Table 6.1: Example of schedules, J30 data-set

UnoSign is calculated in multiple stages. We calculate ‘the Number’ as Stage One in the four steps for calculating *UnoSign*, as illustrated in Table 6.2.

Schedule No	Calculation, where we use ‘g’ = 5 and ‘b’ = 10	Calculation Result
#A	$(1 * 10^4 + 4 * 10^3 + 3 * 10^2 + 13 * 10^1 + 18 * 10^0)$ + $(10 * 10^4 + 9 * 10^3 + 8 * 10^2 + 7 * 10^1 + 27 * 10^0)$ + $(12 * 10^4 + 5 * 10^3 + 2 * 10^2 + 16 * 10^1 + 15 * 10^0)$ + $(21 * 10^4 + 6 * 10^3 + 19 * 10^2 + 11 * 10^1 + 14 * 10^0)$ + $(17 * 10^4 + 22 * 10^3 + 29 * 10^2 + 26 * 10^1 + 20 * 10^0)$ + $(25 * 10^4 + 28 * 10^3 + 23 * 10^2 + 24 * 10^1 + 31 * 10^0)$ + $(30 * 10^4 + 32 * 10^3 + 0 * 10^2 + 0 * 10^1 + 0 * 10^0)$ =	14448 + 109897 + 125375 + 218024 + 195180 + 280571 + 332000 = 1275495
#B	$(1 * 10^4 + 3 * 10^3 + 8 * 10^2 + 4 * 10^1 + 5 * 10^0)$ + $(19 * 10^4 + 12 * 10^3 + 13 * 10^2 + 9 * 10^1 + 7 * 10^0)$ + $(18 * 10^4 + 29 * 10^3 + 10 * 10^2 + 14 * 10^1 + 17 * 10^0)$ + $(2 * 10^4 + 27 * 10^3 + 11 * 10^2 + 16 * 10^1 + 6 * 10^0)$ + $(15 * 10^4 + 20 * 10^3 + 21 * 10^2 + 25 * 10^1 + 28 * 10^0)$ + $(26 * 10^4 + 31 * 10^3 + 22 * 10^2 + 23 * 10^1 + 24 * 10^0)$ + $(30 * 10^4 + 32 * 10^3 + 0 * 10^2 + 0 * 10^1 + 0 * 10^0)$ =	13845 + 203397 + 210157 + 48266 + 172378 + 293454 + 332000 = 1273497
#C	$(1 * 10^4 + 3 * 10^3 + 8 * 10^2 + 13 * 10^1 + 12 * 10^0)$ + $(18 * 10^4 + 7 * 10^3 + 19 * 10^2 + 29 * 10^1 + 2 * 10^0)$ + $(4 * 10^4 + 5 * 10^3 + 15 * 10^2 + 9 * 10^1 + 14 * 10^0)$ + $(27 * 10^4 + 17 * 10^3 + 6 * 10^2 + 10 * 10^1 + 16 * 10^0)$ + $(22 * 10^4 + 21 * 10^3 + 11 * 10^2 + 20 * 10^1 + 25 * 10^0)$ + $(23 * 10^4 + 24 * 10^3 + 28 * 10^2 + 30 * 10^1 + 26 * 10^0)$ + $(31 * 10^4 + 32 * 10^3 + 0 * 10^2 + 0 * 10^1 + 0 * 10^0)$ =	13942 + 189192 + 46604 + 287716 + 242325 + 257126 + 342000 = 1378905

Table 6.2 : First stage of calculating *UnoSign*.

(Multiplying by 10, we add ‘zero’ as the last digit to get $N_A = 12754950$, $N_B = 12734970$, and $N_C = 13789050$. N_C is larger than N_A and N_B , which already established that strength of C is lower. Since $D_A = D_B = 47$, we need to modify the last digit if and only if $[N_A] = [N_B]$ at Stage Two. Since this is not so, therefore no change is made to the Numbers. We have not used this optional Stage Two since the possibility of duplicate *UnoSign* is remote in our tests. A more practical reason for not using this stage is that the value might exceed the capacity of data type that we use for *UnoSign*.)

As Stage Three, we concatenate the (resource-constrained) makespan to the front of each Number.

This provides *UnoSign* for each schedule, as depicted in Table 6.3

Schedule No.	Sequence of the Schedule	Resource Constrained Makespan	<i>UnoSign</i>
#A	1-4-3-13-18-10-9-8-7-27-12-5-2-16-15-21-6-19-11-14-17-22-29-26-20-25-28-23-24-31-30-32	47 days	471275495
#B	1-3-8-4-5-19-12-13-9-7-18-29-10-14-17-2-27-11-16-6-15-20-21-25-28-26-31-22-23-24-30-32	47 days	471273497
#C	1-3-8-13-12-18-7-19-29-2-4-5-15-9-14-27-17-6-10-16-22-21-11-20-25-23-24-28-30-26-31-32	49 days	491378905

Table 6.3: Third stage of calculating *UnoSign*.

At Stage Four we use the fitness function: [*Lower is Better(UnoSign)*]. In our examples, since $UnoSign_B < UnoSign_A$, therefore B is a ‘better’ or ‘stronger’ schedule.

In our implementation we use a ‘long integer’ data type of C, to store *UnoSign* at the (n+1)th gene in the chromosome.

6.6 Elites, or ‘Solution Set’

The ‘best’ set of solutions of any current generation (including the initial population) is copied off as elites. At any given point of processing, the elites form the ‘solution set’ that may be presented to the Project Manager for his use. For our implementation, we have used an elite set of five solutions, and the (five) solutions with lowest *UnoSign* qualify to be in this set.

6.7 Population size

Population size for our algorithm is kept the same for initial population and subsequent generations. We call it *MaxPop*. For most part of our work, we used fifty as *MaxPop*. However we experimented with thirty as well to check the impact of such change.

During processing a generation, we allow the population to (notionally) swell up to double of *MaxPop* by generating new individuals. This way we use an adaptation of the 'Steady-State Genetic Algorithm'. The 'parents' are retained alongwith the 'offspring', both in equal numbers.

The 'next' generation is selected out of this 'double size' group

6.8 Clones

With an aim of allowing a 'stronger' parent to be able to produce more offspring, we produce (notional) clones or 'copies'. We demonstrate usage of our 'Copy' algorithm with the help of a set of makespan that we collected from thirty schedules of the Project of our example.

Sch No	$S(i)$	$(S_{w+1}) - S(i)$	τ	Copy
#1	47	32	6.275	7
#2	56	23	4.510	5
#3	45	34	6.667	7
#4	43	36	7.059	8
#5	67	12	2.353	3
#6	54	25	4.902	5
#7	48	31	6.078	7
#8	49	30	5.882	6
#9	54	25	4.902	5
#10	43	36	7.059	8
#11	57	22	4.314	5
#12	78	1	0.196	1
#13	54	25	4.902	5
#14	68	11	2.157	3
#15	56	23	4.510	5
#16	47	32	6.275	7
#17	55	24	4.706	5
#18	46	33	6.471	7
#19	76	3	0.588	1
#20	44	35	6.863	7
#21	48	31	6.078	7
#22	50	29	5.686	6
#23	57	22	4.314	5
#24	52	27	5.294	6
#25	46	33	6.471	7
#26	51	28	5.490	6
#27	47	32	6.275	7
#28	58	21	4.118	5
#29	56	23	4.510	5
#30	49	30	5.882	6
$S(w) = 78$		$\Sigma = 769$		

Table 6.4 : Calculation of *Copy*, i.e permitted clones

We use Relationship 5.4, with Clone Factor, $C = 1$.

The worst schedule (#12, makespan 78) has a τ less than one. Similarly with another schedule (#19, makespan 76). To allow these ‘worse’ schedules to (potentially) mate, we have to retain at least one copy each in the fresh parent population. This justifies the significance of our ‘Clone Factor’.

Note that due to ‘cloning’, we notionally have a total of 167 parents in our example.

Every time a parent mates successfully, its *Copy* is reduced by one. A parent is eligible to mate till it has a positive *Copy* value.

Copy is carried with each individual as $(n+2)^{th}$ gene of the chromosome. Incorporation of this gene into the chromosome completes representation of a schedule, and makes a schedule ready to be selected for mating.

6.9 Selection

For selection of (a pair of) parents to mate, we utilize the ‘*Better-Spouse Selection*’, as described in the previous Chapter. For illustrating the usage, we employ the results of Table 6.4 and have included *UnoSign*. Table 6.5 is sorted in ascending order of *UnoSign*, abiding by our Fitness Function. We notice that the schedule with least

Priority	Sdl No	Makespan	Copy (C = 1)	UnoSign
p1	#10	43	8	431264976
p2	#4	43	8	431303180
p3	#20	44	7	441197373
p4	#3	45	7	451247336
p5	#18	46	7	461232932
p6	#25	46	7	461438728
p7	#16	47	7	471273497
p8	#1	47	7	471275495
p9	#27	47	7	471399203
p10	#7	48	7	481146718
p11	#21	48	7	481502204
p12	#30	49	6	491313714
p13	#8	49	6	491378905
p14	#22	50	6	501247662
p15	#26	51	6	511791490
p16	#24	52	6	521663820
p17	#13	54	5	541297582
p18	#9	54	5	541617016
p19	#6	54	5	541901366
p20	#17	55	5	551494499
p21	#29	56	5	561400711
p22	#15	56	5	561481315
p23	#2	56	5	561535584
p24	#11	57	5	571405738
p25	#23	57	5	571592580
p26	#28	58	5	581614600
p27	#5	67	3	671743986
p28	#14	68	3	681362700
p29	#19	76	1	761818987
p30	#12	78	1	781766235

Table 6.5 : Schedules list sorted on UnoSign

UnoSign has top priority, and highest value of *Copy*. The implication is that a ‘fitter’ schedule will have a higher probability of getting selected.

On random, we chose any parent from amongst p2 to p30. Let this be the p(n)th schedule in the sorted list. We select the spouse for this schedule from anyone ‘better than’ the nth one. For either selection, if *Copy* value of the selected schedule is found to be zero, we discard the choice and chose another.

As soon as a pair of parents’ selection (and thereafter their mating) is successful, we reduce the *Copy* value of the two selected parents by one. It is possible that the *Copy* value now becomes zero, in which case such parent is no longer eligible to be selected.

For our experimentation, we compared *Better-Spouse* Selection (BS) with the Russian Roulette Selection (RR) technique. Selected parents are now invited for mating to produce offspring by crossover.

6.10 Crossover

For crossover we have adopted a methodology that is robust and proven for the RCPSP. Single Point Precedence Set Crossover (PSX1) guarantees conformance to precedence-constraint of a project, while maintaining sequence conformance of the two parents in generating offspring. For our usage, we have adjusted and amended certain aspects of PSX1.

We allow mating to be successful, without any deterrent of the crossover probability. Or stated in another form, in deference to those authors who advocate a high crossover probability, we use a probability of one (i.e. hundred percent chance) for mating.

Once selected, the parents' chromosomes exchange genes only from the first 'n' positions, where Project Tasks feature. *UnoSign* and *Copy* genes are not exchanged (obviously). We start off with an empty chromosome (of normal size, i.e. $n+2$) for the offspring. Based on mode of crossover – Mid-Point or Random-Point, pertinent genes from mating parent are copied into the new chromosome.

Crossover point, mating scheme, number of offspring produced, etc are dependent on the mode of crossover chosen.

6.10.1 Crossover point

- a) For **Mid-Point** crossover mode, the point of crossover is taken as integer portion of $n/2$, i.e., the operation $[\text{INT}(n/2)]$ is used.
- b) In case of **Random-Point** crossover mode, we select the point from anywhere between $[2^{\text{nd}}$ to $(n-1)^{\text{th}}$] position.

6.10.2 Mating schemes and number of offspring

- a) For **Mid-Point** crossover mode, we have two mating schemes, viz.
 - i. Parent1 (*mates with* Parent2) to produce Offspring1, and
 - ii. Parent2 (*mates with* Parent1) to produce Offspring2

The two mating schemes generate two different offspring.

- b) In **Random-Point** crossover mode, we denote the '*Better-Spouse*' as Parent1. This parent is allowed to mate (possibly multiple times) with the (first selected) other parent; but the reverse scheme of mating is not permitted. Mating is permitted at least once (to produce one offspring) with the

maximum number of possible mating limited to current *Copy* value of Parent1.

Therefore the mating scheme is

Parent1 (*mates* [C time(s)] *with*) Parent2 to produce C Offspring(s)
 where C is the current Copy value of Parent1.

Once an offspring is produced, its *UnoSign* is determined immediately. To avoid possible premature convergence, we discard any duplication of offspring, and also check its possible duplication with any parent. After crossover operation is complete *MaxPop* individuals are taken to the next generation.

Characteristics of the crossover being utilized is summarized in Table6.6.

Crossover technique : Precedence Set Crossover, Single Point, or PSX1		
Characteristics	Mode	
	Mid-Point crossover	Random-Point crossover
Crossover point	INT [n/2]	RANDOM [2 nd to (n-1) th]
Mating schemes (symbol we use χ to denote 'mates with' \ni to denote 'to produce')	a) [Parent1 χ Parent2] \ni Offspring1 b) [Parent2 χ Parent1] \ni Offspring2	{[BetterSpouse χ Parent2] \ni Offspring} C times Max where C is current Copy value of the BetterSpouse
Offspring produced	Two	Copy(BetterSpouse), minimum One

Table 6.6 : Characteristics of crossover utilized.

6.11 Next Generation

At any point of processing, we have the 'current' population and the elites. When offspring are produced, the total number of individuals in hand swells to twice of *MaxPop*. Note that elites are a subset of the parent population.

From this pool of $[2 \times \text{MaxPop}]$ individuals, we allow the best *MaxPop* number of individuals to proceed into the next generation. Five of the best individuals amongst this ‘next’ generation is copied off as elites. The criterion for ‘best’ individual is again our fitness function, [*Lower is Better (UnoSign)*].

The ‘next’ generation is now made vulnerable for possible intrusion of ‘aliens’, either as *immigrants* or as *dormant-forefathers*. Since our solution method employs Random Numbers generously (albeit in a controlled manner) there is high amount of sub-optimal and ‘noisy’ solutions. This is a situation which Sastry and Goldberg (2007) advocates crossover to be stronger, at the cost of mutation. We took this generalization a step further and avoid mutation altogether as a technique for introducing diversity of a population.

At this point let us term this generation as ‘pseudo-current’ generation. Once these intrusion phases are over, this ‘pseudo-current’ generation takes the role of ‘current’ generation.

6.12 Intrusions

To introduce sudden asymmetry in the ‘pseudo-current’ generation, we try to ‘intrude’ it by ‘alien’ individuals. This is attempted by an alien individual who might be either an ‘*immigrant*’ (freshly generated) or a ‘*dormant-forefather*’ (generated in an earlier generation). In every population we have restricted the number of attempts for intrusion at five per type.

6.12.1 Immigration

For possible immigration, we have to overcome two probabilities

- a) probability of generating an alien, and
- b) probability of intrusion into ‘pseudo-current’ generation by the alien.

Both are kept low, in the range of 0.3 (or below). These are applied as analogous to the Boolean 'AND' operator. In effect, the chance of an immigrant becoming a member of the population is a very low probability of 0.09.

If the first probability is favourable, we generate a fresh individual using the same function of our program that generated members of the initial population. It is then compared with members of the 'pseudo-current' generation. The individual whose *UnoSign* is immediately lower than *UnoSign* of the alien is targeted for replacement. If *UnoSign* of alien is lower than the weakest individual, then the weakest individual is targeted. However, intrusion by replacement is permitted only if the second probability is favourable.

A slight variation of intrusion would be to shift down all lower individuals to make space for the immigrant. In that case the 'target' individual would remain in population, at the cost of the 'weakest' one.

6.12.2 Dormant-Forefather

If allowed, attempt for infiltrating into the 'pseudo-current' generation by *Dormant-Forefather* proceeds simultaneously with possible *Immigration*. To allow this mechanism or not is controlled during each experimental run. This we did to check effectiveness of the *Dormant-Forefather* technique.

A *Dormant-Forefather* is retained from the discarded individuals of previous generation(s). Individual(s) to be retained is selected randomly from the discarded lot, and retained – if favoured by a low probability – for 'mummification'. We have used a 'mummies set' of ten *Dormant-Forefathers*.

The possibility factor for infiltration by a *Dormant-Forefather* is kept even lower than that for immigrant intrusion. It is made to depend on three probabilities, each of low value in the range of 0.3 or below:

- a) probability that some forefathers has been mummified, which are still in the ‘mummies set’, and
- b) probability of reviving a *Dormant-Forefather*, selected at random from the set of (ten) ‘mummies’, and
- c) probability that the revived individual will be allowed entry into ‘pseudo-current’ generation.

Once a *Dormant-Forefather* is finally allowed entry into a population, it moves away from the ‘mummies set’. One individual of the discarded lot of the ‘pseudo-current’ generation fills up the vacant slot.

6.13 Termination

We experimented termination using three different criterions –

- a) Fixed Generation Termination,
- b) Fitness-Deviation Termination, and
- c) Adaptive Termination.

6.13.1 Fixed Generation Termination

For the Fixed Generation Termination, we initially test feasibility of our algorithm with maximum generations (*MaxGen*) of (i) one hundred, and (ii) five hundred, keeping *MaxPop* at thirty. Next, to check conformity of our algorithm results with published results, we change *MaxPop* to fifty, and ran the Genetic Algorithm for (i) twenty, (ii) hundred, and (iii) thousand *MaxGen*. This part of the experiment is more for checking effectiveness of Termination algorithm rather than to test parameters.

6.13.2 Fitness-Deviation Termination

For experimentation with Fitness-Deviation Termination, we used the following scheme

- a) Deviation of Elite, or σ_{EL} : *UnoSign* deviation between 'best' and 'worst' of elites less than or equal to μ percent, AND
- b) Deviation of POPulation, or σ_{PO} : *UnoSign* deviation between 'best' and 'worst' of population less than or equal to δ percent, AND
- c) Deviation of Generations, or σ_{GE} : Deviation of σ_{PO} between amongst subsequent generations equal to ω percent

In some cases we had to restrict runaway processing, as the number of generations were increasing much more than generations required for convergence in other instances. Therefore we added another criterion that restricted generations to match parity with reported schedule(s) in the literature selected for comparing optimal results. Thus the last criteria of the scheme is

- d) OR, *MaxGen*, while processing the Genetic Algorithm with predefined *MaxPop*.

We summarize this Fitness-Deviation Termination criterion as

$$\begin{aligned} & [(\sigma_{EL} \leq \mu) \quad \mathbf{AND} \\ & \quad (\sigma_{PO} \leq \delta) \quad \mathbf{AND} \\ & \quad (\sigma_{GE} = \omega) \\ &] \\ & \qquad \qquad \qquad \mathbf{OR} \\ & [MaxGen * MaxPop = Max\#Schedules_{literature}] \\ & \text{where } \mu, \delta, \text{ and } \omega \text{ are the permissible limits} \end{aligned}$$

...Relationship : 6.1

6.13.3 Adaptive Termination

We use of different characteristics of the Project., viz. Project length, number of Resources under constraint, and Complexity level for devising an Adaptive Termination. In the first instance we ignore the Complexity level and use only the common and constant characteristics for the data-set. We test three variations of Adaptive Termination algorithm; each subsequent being a refinement of the previous one.

- a) To accommodate the characteristics – Project Length (T) and Resources under constraint (R), we simply take the product of the two, as *MaxGen* i.e.

$$G = T * R$$

...*Relationship : 6.2*

The Genetic Algorithm adapts to these two common characteristics of all Projects within the test data-set, and proceeds with the processing. When we switch for processing a longer Project, say change the data-set from J30 to the J60 series, we will have a deeper termination. Similar will be the case with a Project that has more number of Resources under constraint. We label this as Adaptive Termination 1, or AdapTerm1, as depicted in Relationship 6.2.

- b) Next we incorporate the Complexity level of a Project as an additional parameter to make the criteria more adaptive to Project characteristics.

We define Complexity as a measure of (or proportional to) the maximum possible schedules of a Project. We term this as *MaxSdl*. For example, *MaxSdl* for the Project in Figure 6.2 (*let us call it Project A*) would be around (3×10^4) . On the other hand, the project depicted as J3035_6.rcp in PSPLIB (*let us call it Project B*) would have in excess of (3×10^9) schedules.

Even though both projects are of same length and use the same number of constrained resources, but the difference in degree of complexity is of the order 10^5 . For the J60 data-set, the range is (1.02×10^9) of J60_01.rcp to (2.11×10^{18}) of J6041_6.rcp test file.

When we process Project A with termination criteria of, say 3000 schedules, we are exploring about 10 percent of total search space, or *MaxSdls* when we search for solution of the least complex of data-set J30. On the other hand for same termination criteria, in Project B we would be searching only about (10^{-4}) percent of the search space. If we use our previous method [i.e. $G = T * R$] with a *MaxPop* of fifty, we would be searching for optimal solution from amongst $(30 * 4 * 50 * 2 =)$ 12,000 schedules. We will be searching about 40 percent of total search space of Project A, and (4×10^{-4}) percent of Project B. Comparing both situations, the difference is of the order 10^5 . To circumvent this dichotomy, we seek to provide proportional search space sampling for every Project.

One obvious way would be simply to sample a fixed percentage of the search space – say ten percent of *MaxSdl*. In that case, we will have

$$\begin{aligned} \text{MaxGen (Project A)} &= 60, \text{ and} \\ \text{MaxGen (Project B)} &= (60 \times 10^6) \end{aligned}$$

which is again impractical considering the computational depth which would be required for Project B.

To dampen the sharp increase we test Logarithmic function, which allows a proportional change with controlled velocity.

Within the framework of Adaptive Termination algorithm as given in the previous Chapter, two versions are devised and tested. The first is a Logarithmic relationship that we term as Adaptive Termination 2, and the second is a

combination of Exponential and Logarithmic functions that we term as Adaptive Termination 3.

i)	Adaptive Termination 2		
	$G = [T * R] * INT[\log_b M]$... Relationship 6.3
ii)	Adaptive Termination 3		
		$[INT(\log_b M) - (\phi - \omega)]$	
	$G = [T * R] * \beta$, for $M \geq 10^\phi$... Relationship 6.4 (a)
	$= [T * R] * INT[\log_b M]$, for $M < 10^\phi$... Relationship 6.4 (b)
	where		
	G : MaxGen		
	T : Project length		
	R : Constrained Resources		
	M : MaxSdl,		
	β : Base of the exponential function,		
	b : Base of the logarithm used, and		
	ϕ : A limiting factor		
	ω : A small non-negative integer.		

The Projects are segregated as having 'low complexity' if $M < 10^\phi$ and as 'high complexity' if $M \geq 10^\phi$. For tuning the parameters of Relationship 6.4, we experimented with different (combinations of) values of β , b , ϕ , and ω . We term Relationship 6.3 as AdapTerm2, and Relationship 6.4 as AdapTerm3.

Relationship 6.4 has evolved in stages by incorporating additional features in previous ones. It has Relationship 6.2 and Relationship 6.3 within it.

6.14 Design of Experiment

Design of Experiment (DoE) is a structured and organized method used for determining the relationship between different factors affecting a process and the output of that process. When the results of these experiments are analyzed, they help to identify optimal conditions, the factors that most influence the results (and those that do not) as well as details such as the existence of interactions and synergies between factors. Sir Ronald A. Fisher, the renowned mathematician and geneticist first developed this method in the 1920s and 1930. Today, Fisher's methods of design and analysis are international standards in business and applied science.

Experimental design is a strategy to gather empirical knowledge, i.e. knowledge based on the analysis of experimental data and not purely on theoretical models. It can be applied whenever we intend to investigate a phenomenon in order to gain understanding or improve performance. Design of Experiments (DoE) is widely used in research and development, where a large proportion of the resources go towards solving optimization problems. The key to minimizing optimization costs is to conduct as few experiments as possible. A careful Design of Experiment result in necessitating only a small and relevant set of experiments, and thus helps to reduce costs.

In the Design Matrix, we have carefully charted out the components to be tested and their parameters (or factors) to be tuned. In our experimentation, the focus was on checking the parameters related to the 'trigger' components of Genetic Algorithm.

6.14.1 The Design Matrix

The experiments are to be conducted in two Sections. Section 1 would check if our algorithm actually runs, and that the results are measurable with published best results. Section 2 will be for testing and tuning the components and parameters of our proposed algorithm.

The Design Matrix for our experimentation is given as Table 6.7.

6.14.2 The Experimentation Plan

Based on relevant test combinations, we conduct a controlled number of experiments that would be significant for our study. One parameter or mode was tested with its different values. Then the 'best' value was carried over to the next set of experiments. This way we proceed to a next set of experiment carrying the best set of parameter values and modes.

The Experimentation Plan is given as Table 6.8. In other Chapter(s) we shall be referring to the experiment taken up for discussion by their Experiment Serial Number (ESN). The ESN contains

- a) the Section to which the experiment belongs as per Design Matrix,
- b) the Experiment# as per Design Matrix,
- c) a subset within Experiment# to indicate data-set, and
- d) a count# within subset to indicate serial number

Table 6.7 : The Design Matrix

Section A For checking <i>Effectiveness</i> of the Algorithm					Other parameters and components collated from literature
<u>Parameter/ Method</u>	<u>Description</u>	<u>Value</u>			
MaxPop	Size of (each) population	30	50		Effectiveness of the algorithm is checked with MaxPop = 30
MaxGen, G	Termination criteria	100	20		Comparison of the algorithm with published benchmark results using MaxPop = 50
		500	100		
			1000		
Section B : For Testing and Tuning parameters of the Algorithm					Change the test parameters, <i>ceteris paribus</i>
<u>Parameter / Method</u>	<u>Description</u>	<u>Value / Mode</u>			
<u>Experiment 1 : Selection</u>					
Clone Factor	Minimum clones	1	3	5	Test the impact of Clones Factor on <i>Better Spouse</i>
Selection	Selection Technique	Russian Roulette	Better Spouse		Check if <i>Better Spouse</i> produces 'better' offspring
<u>Experiment 2 : Crossover</u>					
Crossover Mode		Mid-Point	Random-Point		
<u>Experiment 3 Intrusions</u>					Compare impact of different probability values
<u>a) Immigration</u>					
	P(alien) Generating an alien	0.3	0.1		
	P(infiltration) Infiltrating the population, conversion to 'immigrant'	0.3	0.1		

(Section B continued on next page)

Table 6.7 : The Design Matrix (contd)

Section B : For Testing and Tuning parameters of the Algorithm (contd..)					
Parameter / Method	Description	Value / Mode			
b) Dormant-Forefather					
		Allow	Disallow	Check impact of proposed Dormant Forefather	
P(mummification)	Retaining a discarded soln	0 3	0 1		
P(revival)	Revival of a 'mummy'	0 3	0 1		
P(infiltration)	Infiltrating into population	0 3	0 1		
Experiment 4 Termination					
Check proposed termination algorithms					
Fixed MaxGen	(Refer Section A, and Experiments 1, 2, and 3)				
Fitness-Deviation					
	σ_{EL} Deviation of Elites	0 01	0 005		
	σ_{PO} Deviation of Population	0 10	0 050		
	σ_{GE} Deviation of Generation	0 10	0 050		
	MaxGen The fourth criteria	1000	1000		
AdapTerm1				MaxGen, $G = T * R$, keeping $C = 1$	
AdapTerm2	b	The base of Logarithm	10	20	MaxGen, $G = [T * R] * INT[\log_b M]$ ('b' is tested in AdapTermII, and used for AdapTermIII)
AdapTerm3	β	The root factor of C	2	1 5	MaxGen, $G = [T * R] * \beta$ = AdapTerm2
	$\phi - \omega$	The limiting numbers which depend on search space	5 - 1 10 - 4	6 - 4 10 - 5	For J30 data-set For J60 data-set

Table 6.8 The Experimentation Plan

#	Experiment Serial Number (ESN)	Project size (Data-set)	Population size	Selection mode	Clone factor	Crossover mode	Intrusion information	Termination	Termination Parameter values	Checking / Validating / Comparing
1	A1a1	J30	30	RR	1	MP	(0,1,0 1) D	FG	100	Feasibility of the program
2	A1b1	J60	30	BS	3	RP	(0 3, 0 3) A (0 3, 0 3, 0 3)	FG	500	
3	A2a1	J30	50	RR	1	MP	(0,1,0 1) D	FG	20	For comparing the basic algorithm with benchmark results
4	A2a2	J30	50	RR	1	MP	(0,1,0 1) D	FG	100	
5	A2a3	J30	50	RR	3	MP	(0,1,0 1) D	FG	1000	
6	A2b1	J60	50	RR	1	MP	(0,1,0 1) D	FG	20	
7	A2b2	J60	50	RR	1	MP	(0,1,0 1) D	FG	100	
8	A2b3	J60	50	RR	3	MP	(0,1,0 1) D	FG	1000	
9	B1a1	J30	50	RR	3	MP	(0,1,0 1) D	FG	100	Clones Factor
10	B1a2	J30	50	RR	5	MP	(0,1,0 1) D	FG	100	
11	B1b1	J60	50	RR	3	MP	(0,1,0 1) D	FG	100	
12	B1b2	J60	50	RR	5	MP	(0,1,0 1) D	FG	100	
13	B1a3	J30	50	BS	3	MP	(0,1,0 1) D	FG	20	Better-Spouse Selection
14	B1a4	J30	50	BS	3	MP	(0,1,0 1) D	FG	100	
15	B1a5	J30	50	BS	3	MP	(0,1,0 1) D	FG	1000	
16	B1b3	J60	50	BS	3	MP	(0,1,0 1) D	FG	20	
17	B1b4	J60	50	BS	3	MP	(0,1,0 1) D	FG	100	
18	B1b5	J60	50	BS	3	MP	(0,1,0 1) D	FG	1000	
19	B2a1	J30	50	BS	3	RP	(0,1,0 1) D	FG	100	Crossover mode
20	B2a2	J30	50	BS	3	RP	(0,1,0 1) D	FG	1000	
21	B2b1	J60	50	BS	3	RP	(0,1,0 1) D	FG	100	
22	B2b2	J60	50	BS	3	RP	(0,1,0 1) D	FG	1000	
23	B3a1	J30	50	BS	3	RP	(0,3,0 3) D	FG	1000	Intrusion parameters
24	B3a2	J30	50	BS	3	RP	(0 3, 0 3) A (0 3, 0 3, 0 3)	FG	1000	
25	B3a3	J30	50	BS	3	RP	(0 1, 0 1) A (0 1, 0 1, 0 1)	FG	1000	
26	B3a4	J30	50	BS	3	RP	(0 3, 0 3) A (0 1, 0 1, 0 1)	FG	1000	
27	B3a5	J30	50	BS	3	RP	(0 1, 0 1) A (0 3, 0 3, 0 3)	FG	1000	
28	B4a1	J30	50	BS	3	RP	(0 3, 0 3) A (0 3, 0 3, 0 3)	FD	(0 01, 0 10, 0 10)	Termination
29	B4a2	J30	50	BS	3	RP	(0 3, 0 3) A (0 3, 0 3, 0 3)	FD	(0 05, 0 50, 0 50)	
30	B4b1	J60	50	BS	3	RP	(0 3, 0 3) A (0 3, 0 3, 0 3)	FD	(0 01, 0 10, 0 10)	
31	B4b2	J60	50	BS	3	RP	(0 3, 0 3) A (0 3, 0 3, 0 3)	FD	(0 05, 0 50, 0 50)	

Table 6.8 The Experimentation Plan (Contd)

#	Experiment Serial Number (ESN)	Project size (Data-set)	Population size	Selection mode	Clone factor	Crossover mode	Intrusion information	Termination	Termination Parameter values	Checking / Validating / Comparing
32	B4a5	J30	50	BS	3	MP	(0 3, 0 3) A (0 3, 0 3, 0 3)	AT1		Termination (contd)
33	B4a6	J30	50	BS	3	RP	(0 3, 0 3) A (0 3, 0 3, 0 3)	AT1		
34	B4b5	J60	50	BS	3	MP	(0 3, 0 3) A (0 3, 0 3, 0 3)	AT1		
35	B4b6	J60	50	BS	3	RP	(0 3, 0 3) A (0 3, 0 3, 0 3)	AT1		
36	B4a07	J30	50	BS	3	MP	(0 3, 0 3) A (0 3, 0 3, 0 3)	AT2	10	
37	B4a08	J30	50	BS	3	RP	(0 3, 0 3) A (0 3, 0 3, 0 3)	AT2	10	
38	B4b07	J60	50	BS	3	MP	(0 3, 0 3) A (0 3, 0 3, 0 3)	AT2	10	
39	B4b08	J60	50	BS	3	RP	(0 3, 0 3) A (0 3, 0 3, 0 3)	AT2	10	
40	B4a09	J30	50	BS	3	MP	(0 3, 0 3) A (0 3, 0 3, 0 3)	AT2	20	
41	B4a10	J30	50	BS	3	RP	(0 3, 0 3) A (0 3, 0 3, 0 3)	AT2	20	
42	B4b09	J60	50	BS	3	MP	(0 3, 0 3) A (0 3, 0 3, 0 3)	AT2	20	
43	B4b10	J60	50	BS	3	RP	(0 3, 0 3) A (0 3, 0 3, 0 3)	AT2	20	
44	B4a11	J30	50	BS	3	MP	(0 3, 0 3) A (0 3, 0 3, 0 3)	AT3	(10, 2, 5-1)	
45	B4a12	J30	50	BS	3	RP	(0 3, 0 3) A (0 3, 0 3, 0 3)	AT3	(10, 2, 5-1)	
46	B4b11	J60	50	BS	3	MP	(0 3, 0 3) A (0 3, 0 3, 0 3)	AT3	(20, 2, 10-4)	
47	B4b12	J60	50	BS	3	RP	(0 3, 0 3) A (0 3, 0 3, 0 3)	AT3	(20, 2, 10-4)	
48	B4a13	J30	50	BS	3	MP	(0 3, 0 3) A (0 3, 0 3, 0 3)	AT3	(10, 1 5, 6-4)	
49	B4a14	J30	50	BS	3	RP	(0 3, 0 3) A (0 3, 0 3, 0 3)	AT3	(10, 1 5, 6-4)	
50	B4b13	J60	50	BS	3	MP	(0 3, 0 3) A (0 3, 0 3, 0 3)	AT3	(20, 1 5, 10-5)	
51	B4b14	J60	50	BS	3	RP	(0 3, 0 3) A (0 3, 0 3, 0 3)	AT3	(20, 1 5, 10-5)	
Legends		RR	Russian Roulette	MP	Mid-Point crossover	D	Disallow (Dormant-Forefather)			
		BS	<i>Better Spouse</i>	RP	Random Point crossover	A	Allow (Dormant-Forefather)			
		FG	Fixed Generations	FD	Fitness-Deviation	AT	Adaptive Termination			
Arrangement of ESN		First Alphabet		Second Numeral		Third Alphabet		Fourth Numeral		
		Section (A / B)		Experiment Number (1 – 4)		Data-Set (a J30 / b J60)		Subset Serial Number		
Repetition of experiments		Random Numbers are used extensively in the algorithm, hence the experiments are carried out ten times on each data-set to offset any bias								

6.15 Methodology for Result Analysis

For analyzing the results, we use common statistical tools and technique, and graphical display by transferring the result sets to spreadsheets. The analysis of result is done in different stages using simplified adaptation of statistical relationships. The Algorithm is validated from three angles – *effectiveness*, *accuracy* and *efficiency*. Finally we present our proposed Optimization Algorithm based on the analysis.

i) To compare with results of other researchers, we use Percentage Average Deviations (PAD) as devised by Kolisch and Hartmann (2006) in their methodology.

$$\text{PAD} = \frac{\sum \text{Makespan}[\text{Test}] - \sum \text{Makespan}[\text{Reference}]}{\sum \text{Makespan}[\text{Reference}]} * 100$$

...Relationship 6.5

For our experimentation, we have taken PSPLIB information as Reference. The results are considered better as PAD keeps reducing, and approaches zero.

ii) For comparing performance between (combination of) modes and/or parameters, for each run of every test instance we devise the Percentage Instance Deviation (PID) for individual instance,

$$\text{PID} = \frac{\text{Makespan}[\text{Test}] - \text{Makespan}[\text{Reference}]}{\text{Makespan}[\text{Reference}]} * 100$$

...Relationship 6.6

The results are considered better as PID approaches zero.

The effectiveness of 'this' combination under scan, as compared to some 'other' combination, is to be considered better by comparison of Average PID (APID), if

$$APID_{this} < APID_{other} \quad \dots Relationship\ 6.7(a)$$

where

$$APID_{this} = \sum PID_i / \sum i = PAD_{this}$$

for all instances, i , within a data-set, tested with 'this' combination

$$\dots Relationship\ 6.7(b)$$

...Relationship 6.7

(It may be noted that APID is same as PAD, but calculated in a roundabout way)

Performance Level Variation (PLV), and the related Average PLV, which we now define, measures a change in performance between the two combinations.

$$PLV(this, other)_i = \frac{PID_{i, other} - PID_{i, this}}{PID_{i, other}} * 100$$

...Relationship 6.8

And,

$$APLV(this, other) = \frac{PAD_{other} - PAD_{this}}{PAD_{other}} * 100$$

...Relationship 6.9

A positive value for the two implies an improvement in performance due to 'this' combination as compared to some 'other' combination. For most of our comparisons, we shall be using equivalent results of ESN A series of experiments as the 'other' experiment.

iii) To validate efficiency of a selected combination, we define and calculate an *Efficiency Index* (EI) for each test instance. This *Efficiency Index* is an indication of effectiveness of the method in reaching a level of accuracy within the sample area of the total search space. *Efficiency Index* for the i^{th} instance within a specific data-set,

$$EI_i = \text{Log}_{10} \left[\frac{\text{Makespan}[\text{Reference}]_i / \text{Makespan}[\text{Test}]_i}{\text{Sample proportion}_i} \right]$$

...Relationship 6.10(a)

where,

$$\text{Sample Proportion}_i = \frac{\text{MaxPop} * \text{MaxGen}_i \text{ [as per Termination Criteria used]}}{\text{MaxSdl}_i}$$

Relationship 6.10(b)

Efficiency of a combination is to be considered better if *Efficiency Index* is higher as compared to that of another combination for the same instance. This is an index at the instance level, and is not to be averaged out over the total data-set.

iv) To finally present the proposed algorithm, we check performance of our algorithm. The set with best (i.e. lowest) PAD, which should ideally correspond to highest APLV, is to be presented as our proposed Optimization Algorithm.

Chapter Seven

Analysis of Experimental Results

In this Chapter we present the experimental results, and analysis thereof using simple statistical measures. The Chapter is laid out mostly as sequenced in the Experimentation Plan. Towards the end we present the proposed algorithm.

7.1 Introduction

During experimentation, shifting of values of the parameters under test was done at times to tune those. That way, we proceed as per the Plan, but keep modifying and changing values in it. The Matrix and the Plan presented in the previous Chapter is the final setting, based on which we proceed to report our findings. This Chapter is laid out as per sequence of experimentation as charted in the Experimentation Plan. The complete results of the experiments as laid out in the Experimental Plan are given as Table 7.1. Next we proceed to analyze subsets of the experimental result. Finally we compare experimental results with benchmark results before presenting the proposed model in its final form.

7.2 Program Feasibility and Conformance

The first part of experimentation is an attempt to establish feasibility and conformance of the overall algorithm. Here we deal with Section A as depicted in the Design Matrix.

7.2.1 Program feasibility

Before we run our program to test concurrence with benchmark results, we check whether the algorithm will actually deliver results as expected of a Genetic Algorithm. For this we carefully make combination of components and parameters. These are noted as experiment set A1 on the Plan, corresponding to first segment of Section A.

The program ran well to deliver reasonable accuracy of results and proved its feasibility.

Table 7.1 : Experimental Results

#	ESN	Project size	Population size	Selection mode	Clone factor	Crossover mode	Intrusion information	Termination	Termination Parameter values	PAD	APLV (this, ESN A), equivalent generations	APLV (this, ESN A), more generations	Remarks / Comments
3	A2a1	J30	50	RR	1	MP	(0,1,0,1)D	FG	20	5.565			Initial Results
4	A2a2	J30	50	RR	1	MP	(0,1,0,1)D	FG	100	3.913			
5	A2a3	J30	50	RR	3	MP	(0,1,0,1)D	FG	1000	1.919			
6	A2b1	J60	50	RR	1	MP	(0,1,0,1)D	FG	20	22.967			
7	A2b2	J60	50	RR	1	MP	(0,1,0,1)D	FG	100	19.626			
8	A2b3	J60	50	RR	3	MP	(0,1,0,1)D	FG	1000	17.338			
9	B1a1	J30	50	RR	3	MP	(0,1,0,1)D	FG	100	3.282	16.1257		
10	B1a2	J30	50	RR	5	MP	(0,1,0,1)D	FG	100	3.252	16.8924		
11	B1b1	J60	50	RR	3	MP	(0,1,0,1)D	FG	100	18.439	6.0481		
12	B1b2	J60	50	RR	5	MP	(0,1,0,1)D	FG	100	18.304	6.7360		
13	B1a3	J30	50	BS	3	MP	(0,1,0,1)D	FG	20	4.931	11.3926		
14	B1a4	J30	50	BS	3	MP	(0,1,0,1)D	FG	100	2.750	29.7214		
15	B1a5	J30	50	BS	3	MP	(0,1,0,1)D	FG	1000	1.365	28.8692		
16	B1b3	J60	50	BS	3	MP	(0,1,0,1)D	FG	20	20.182	12.1261		
17	B1b4	J60	50	BS	3	MP	(0,1,0,1)D	FG	100	16.525	15.8005	4.6891	
18	B1b5	J60	50	BS	3	MP	(0,1,0,1)D	FG	1000	14.634	15.5958	15.5958	Contd

Table 7.1 Experimental Results (contd)

#	ESN	Project size	Population size	Selection mode	Clone factor	Crossover mode	Intrusion information	Termination	Termination Parameter values	PAD	APLV (this, ESN A), equivalent generations	APLV (this, ESN A), more generations	Remarks / Comments
19	B2a1	J30	50	BS	3	RP	(0,1, 0 1) D	FG	100	2.682	31.4592		
20	B2a2	J30	50	BS	3	RP	(0,1, 0 1) D	FG	1000	1.325	30.9536		
21	B2b1	J60	50	BS	3	RP	(0,1, 0 1) D	FG	100	16.341	16.7380	5.7504	
22	B2b2	J60	50	BS	3	RP	(0,1, 0 1) D	FG	1000	14.210	18.0413	18.0413	
23	B3a1	J30	50	BS	3	RP	(0,3, 0 3) D	FG	1000	1.166	39.2392		
24	B3a2	J30	50	BS	3	RP	(0 3, 0 3) A (0 3, 0 3, 0 3)	FG	1000	1.092	43.0954		
25	B3a3	J30	50	BS	3	RP	(0 1, 0 1) A (0 1, 0 1, 0 1)	FG	1000	1.149	40.1251		
26	B3a4	J30	50	BS	3	RP	(0 3, 0 3) A (0 1, 0 1, 0 1)	FG	1000	1.131	41.0631		
27	B3a5	J30	50	BS	3	RP	(0 1, 0 1) A (0 3, 0 3, 0 3)	FG	1000	1.135	40.8546		
28	B4a1	J30	50	BS	3	RP	(0 3, 0 3) A (0 3, 0 3, 0 3)	FD	(0 01, 0 10, 0 10)	3.562	8.9701		
29	B4a2	J30	50	BS	3	RP	(0 3, 0 3) A (0 3, 0 3, 0 3)	FD	(0 05, 0 50, 0 50)	3.816	2.4789		
30	B4b1	J60	50	BS	3	RP	(0 3, 0 3) A (0 3, 0 3, 0 3)	FD	(0 01, 0 10, 0 10)	17.453	11.0720		
31	B4b2	J60	50	BS	3	RP	(0 3, 0 3) A (0 3, 0 3, 0 3)	FD	(0 05, 0 50, 0 50)	19.761	-0.6879		
32	B4a5	J30	50	BS	3	MP	(0 3, 0 3) A (0 3, 0 3, 0 3)	AT1		2.417	38.2315		
33	B4a6	J30	50	BS	3	RP	(0 3, 0 3) A (0 3, 0 3, 0 3)	AT1		2.326	40.5571		
34	B4b5	J60	50	BS	3	MP	(0 3, 0 3) A (0 3, 0 3, 0 3)	AT1		16.169	17.6144		
35	B4b6	J60	50	BS	3	RP	(0 3, 0 3) A (0 3, 0 3, 0 3)	AT1		16.011	18.4194		Contd

Table 7.1 : Experimental Results (contd)

#	ESN	Project size	Population size	Selection mode	Clone factor	Crossover mode	Intrusion information	Termination	Termination Parameter values	PAD	APLV (this, ESN A), equivalent generations	APLV (this, ESN A), more generations	Remarks / Comments
36	B4a07	J30	50	BS	3	MP	(0.3, 0.3) A (0.3, 0.3, 0.3)	AT2	10	1.981	49.3739		
37	B4a08	J30	50	BS	3	RP	(0.3, 0.3) A (0.3, 0.3, 0.3)	AT2	10	1.912	51.1372	0.3648	
38	B4b07	J60	50	BS	3	MP	(0.3, 0.3) A (0.3, 0.3, 0.3)	AT2	10	15.118	22.9702	12.8050	
39	B4b08	J60	50	BS	3	RP	(0.3, 0.3) A (0.3, 0.3, 0.3)	AT2	10	14.522	26.0082	16.2439	
40	B4a09	J30	50	BS	3	MP	(0.3, 0.3) A (0.3, 0.3, 0.3)	AT2	20	2.011	48.6072		
41	B4a10	J30	50	BS	3	RP	(0.3, 0.3) A (0.3, 0.3, 0.3)	AT2	20	1.994	49.0417		
42	B4b09	J60	50	BS	3	MP	(0.3, 0.3) A (0.3, 0.3, 0.3)	AT2	20	15.232	22.3890	12.1471	
43	B4b10	J60	50	BS	3	RP	(0.3, 0.3) A (0.3, 0.3, 0.3)	AT2	20	14.823	24.4733	14.5064	
44	B4a11	J30	50	BS	3	MP	(0.3, 0.3) A (0.3, 0.3, 0.3)	AT3	(10, 2, 5-1)	0.028	99.2960	98.5646	
45	B4a12	J30	50	BS	3	RP	(0.3, 0.3) A (0.3, 0.3, 0.3)	AT3	(10, 2, 5-1)	0.007	99.8195	99.6319	Best Result, J30
46	B4b11	J60	50	BS	3	MP	(0.3, 0.3) A (0.3, 0.3, 0.3)	AT3	(20, 2, 10-4)	10.997	43.9648	36.5701	
47	B4b12	J60	50	BS	3	RP	(0.3, 0.3) A (0.3, 0.3, 0.3)	AT3	(20, 2, 10-4)	10.812	44.9079	37.6377	Best Result, J60
48	B4a13	J30	50	BS	3	MP	(0.3, 0.3) A (0.3, 0.3, 0.3)	AT3	(10, 1.5, 6-0)	0.084	97.8430	95.6016	
49	B4a14	J30	50	BS	3	RP	(0.3, 0.3) A (0.3, 0.3, 0.3)	AT3	(10, 1.5, 6-0)	0.066	98.3213	96.5770	
50	B4b13	J60	50	BS	3	MP	(0.3, 0.3) A (0.3, 0.3, 0.3)	AT3	(20, 1.5, 10-5)	12.171	37.9859	29.8023	
51	B4b14	J60	50	BS	3	RP	(0.3, 0.3) A (0.3, 0.3, 0.3)	AT3	(20, 1.5, 10-5)	11.517	41.3179	33.5739	

7.2.2 Conformance to Benchmark Literature

With the feasibility of our program in place, we next proceed to run the program in a near similar combination as set out in the benchmark result literature of Kolisch and Hartmann(2006). These experiments are depicted as Experiment Serial Number (ESN) as A2a for data-set J30 and A2b for J60.

The benchmark result literature provides the Average Deviations % (which we have termed as Percentage Average Deviation, or PAD) from optimal makespan – for J30 data-set. For J60, PAD is from Critical Path Lower Bound. For our experimental comparison, documented makespan for the test data-set are used as available in the PSPLIB directly. The published results as compared with the best of our experimental results are provided at Section 7.3 of this Chapter.

The PAD comparison of our initial setup with the Average Deviations (%) of published results is presented again in Table 7.2

ESN	Data-set	Max schedules	PAD (%)	
			Benchmark Result Range**	Experimental Results
A2a1	J30	1000	0.10 – 0.54	5.565
A2a2	J30	5000	0.03 – 0.25	3.913
A2a3	J30	50000	0.00 – 0.08	1.919
A2b1	J60	1000	11.59 – 12.77	22.967
A2b2	J60	5000	11.07 – 12.03	19.626
A2b3	J60	50000	10.64 – 11.54	17.338

Table 7.2: The initial Test Results vis-à-vis Benchmark Results Range

** Ref Table 7.13 (a and b)

As expected of a Genetic Algorithm, the deviation has decreased to increase accuracy of the algorithm under test while operating on a larger sample space.

But error factor was high for smaller search samples. To visualize this, we present summary of best, mean and worst PID results for each instance of each data set in

the Figure 7.1 for experiments ESN A2a2 and 7.A2b2 which ran 5000 schedules each.

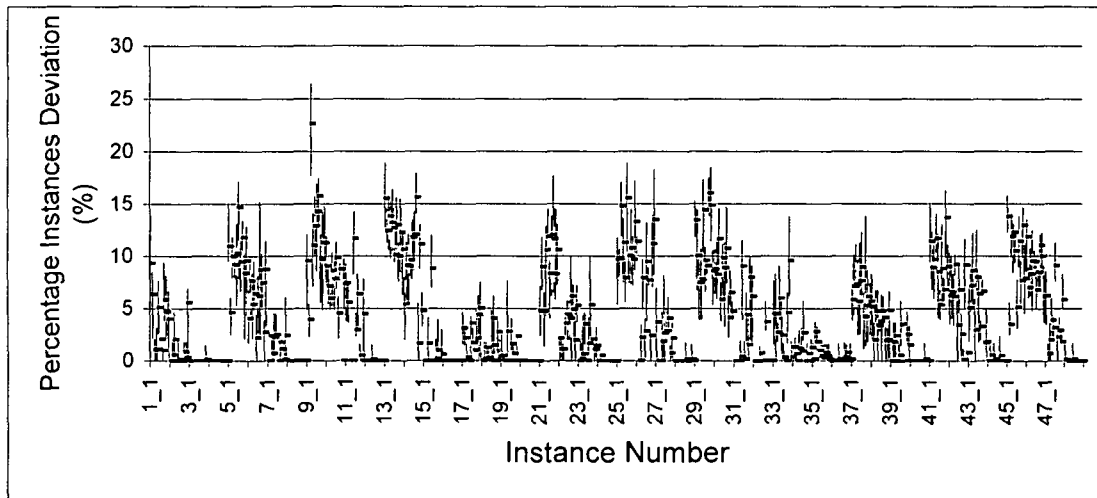


Figure 7.1(a): Summary of deviations, J30, Experiment ESN A2a2

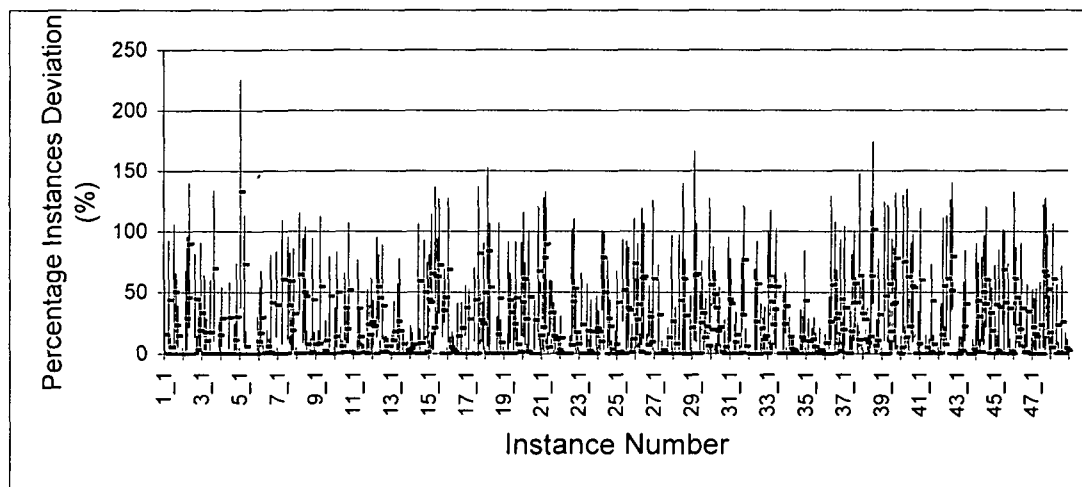


Figure 7.1(b): Summary of deviations, J60, Experiment ESN A2b2

As evident from the two Figures, the deviations are wide, and so is the range of deviation for the settings we used in our initial experiments.

Our aim henceforth would be to tune the parameters and set mode in such a way that these two features reduce, and both approach zero. We shall compare change in (improvement) status at subsequent stages and at the end, by constantly referring Table 7.1 for pertinent subset of results on two fronts, viz.

- a) improvement as compared to these initial results, using APLV, and
- b) improvement as compared to benchmark results, using PAD.

7.3 Validation of Parameters and Operator Modes

The different parameters and operator modes are now taken up for validation, either in isolation (*ceteris paribus*) or as combination of proven elements.

7.3.1 Clones Factor and *Better-Spouse* Selection

As per the Design Matrix, the first parameter we take up is the Clones Factor to check its impact on Selection of parents. In experiment set ESN A2a and A2b we have already checked Russian Roulette Selection with Clones Factor kept at one. Now we carry our experiments using three and five for Clones Factor using Russian Roulette. For each experiment here we check a total of 5000 schedules, and each data-set is run ten times.

ESN	Data-set	Clone factor	Crossover mode	Test Result PAD (%)
B1a1	J30	3	MP	3.282
B1a2	J30	5	MP	3.252
B1b1	J60	3	MP	18.439
B1b2	J60	5	MP	18.304

Table 7.3 Experimental Results: Clones Factor

We notice that there is an improvement in results if Clones Factor is increased from one to three. This may be (tentatively) attributed to increase in the number of clones of stronger individuals in the Selection pool. But when we increase Clones Factor further, there is only minimal further improvement. Performance improved by more than sixteen percent for J30, and about six percent for J60 data-set.

For further experimentation, we use Clones Factor = 3, and keep it steady there. Selection mode is now switched to *Better-Spouse* Selection mechanism, and remainder of experiments under ESN B1 is carried out.

ESN	Data-set	Population size	Selection mode	Clone factor	Crossover mode	Intrusion information	Termination	Termination Parameter values (MaxGen)	Test Result PAD (%)
B1a3	J30	50	BS	3	MP	(0,1, 0.1) D	FG	20	4.931
B1a4	J30	50	BS	3	MP	(0,1, 0.1) D	FG	100	2.750
B1a5	J30	50	BS	3	MP	(0,1, 0.1) D	FG	1000	1.365
B1b3	J60	50	BS	3	MP	(0,1, 0.1) D	FG	20	20.182
B1b4	J60	50	BS	3	MP	(0,1, 0.1) D	FG	100	16.525
B1b5	J60	50	BS	3	MP	(0,1, 0.1) D	FG	1000	14.634

Table 7.4 : Experimental Results for *Better-Spouse* Selection

As evident from Table7.4 the results are favorable. For lower number of Termination generations, we had a 11 – 12 percent improvement. Once the number of generations increased, improvement in performance shot up to nearly 15 percent for J60, and 30 percent for J30 data-set.

For checking effectiveness of the *Better-Spouse* Selection technique, PADs of these experiments are compared with the equivalent experiments that employ Russian

Roulette Selection. Then APLV is computed for *Better-Spouse* Selection over Russian Roulette, as given on the last column of Table 7.5.

Sl. No		ESN	Data-set	Selection	PAD	APLV (BS, RR)
1	a	B1a1	J30	RR	3.282	16.209
	b	B1a4		BS	2.750	
2	a	B1b1	J60	RR	18.439	10.380
	b	B1b4		BS	16.525	
3	a	A2a3	J30	RR	1.919	28.869
	b	B1a5		BS	1.365	
4	a	A2b3	J60	RR	17.338	15.596
	b	B1b5		BS	14.634	

Table 7.5 : Performance comparison between BS and RR Selection

On both data-sets, Better-Spouse Selection outperforms the Russian Roulette Selection technique for converging strongly at the optimal result. This vindicates our assumption that deliberate mating with a stronger spouse produces better offspring. For our next level of experimentation, we use BS as our default Selection mechanism.

7.2.2 Crossover Mode

Till now the Mid Point crossover mode has been the default mode. Moving ahead on the Design Matrix, we next vary Crossover mode to check impact, setting *Better-Spouse* Selection as default. The ESN B2 set is compared with equivalent experiments of ESN B1 set. As evident from the table, comparative results improved with Random-Point crossover when higher number (50,000) of samples

are tested. For making additional comment on quality of these two methods, they would be taken up alternately for checking their performance in tandem with other parameter / modes. Next we check the two intrusion mechanisms, viz. *Immigrant* and *Dormant-Forefather*.

Sl. No	ESN	Data-set	Crossover Mode	PAD	APLV (RP, MP)	
1	a	B1a4	J30	MP	2.750	2 473
	b	B2a1		RP	2.682	
2	a	B1b4	J60	MP	16.525	1.113
	b	B2b1		RP	16.341	
3	a	B1a5	J30	MP	1.365	2.930
	b	B2a2		RP	1.325	
4	a	B1b5	J60	MP	14.634	2 897
	b	B2b2		RP	14.210	

Table 7.6 : Comparison of Crossover mode

7.2.3 Intrusion Mechanisms

So far our experiments had been using *Immigrant* mechanism (with both probabilities kept at 0.1 each) as default technique for introducing diversity. *Dormant-Forefather* was disallowed (except in ESN A1b1 experiment where its feasibility was tested). The next set of experiments is marked as ESN B3, and impact of the two intrusion techniques are tested on J30 data-set. We keep *MaxGen* fixed at the longer 1000, to probe into a deeper space or age-range.

ESN	Intrusion mode	Intrusion parameters	PAD
B2a2	Immigration	(0,1, 0.1) D	1.325
B3a1		(0,3, 0 3) D	1.166
B3a2	Immigration AND Dormant-Forefather	(0.3, 0.3) A (0.3, 0 3, 0 3)	1.092
B3a3		(0 1, 0.1) A (0 1, 0.1, 0 1)	1.149
B3a4		(0.3, 0.3) A (0.1, 0.1, 0.1)	1.131
B3a5		(0 1, 0.1) A (0.3, 0 3, 0.3)	1.135

Table 7.7 : Performance comparison of Intrusion Techniques

As expected, deviation from optimal solution displays an inverse relationship with probability factor values. Performance improves further when both techniques are used in tandem. The APLV of the best combination of intrusion parameters (ESN B3a2) over the worst (ESN B2a2) is 17.585%, which we consider a potentially significant improvement.

For remainder of the experiments, we use intrusion probability combination as used for ESN B3a2. That way actual probability of *Immigration* is maintained constant at 0.09, and of *Dormant-Forefather* it is 0.027 – both being fairly low.

7.2.4 Termination

7.2.4.1 Fixed Generation Termination

All experiments till now were conducted with fixed generation termination. This implies that with a *MaxPop* of fifty for our significant experiments, the sample search space was (50 X 20 =) 1000, (50 X 100 =) 5000, and (50 X 1000 =) 50000 schedules.

However such ‘fixed’ criteria misses a crucial aspect. Once the algorithm is made a general one (to operate on other than test data, whose optimal makespan would not be known for comparison) for any Project, such criteria would be hard pressed due to possible deficiency of ideal fitness value for such field Project. To avoid any bias against unknown Projects, we decided not to advocate ‘fixed’ termination criteria, and test two other possible techniques.

The cumulative improvement until this point of experimentation is manifested by an APLV of 43.095 percent for the current (best) combination of modes and parameters (ESN B3a2) over equivalent initial combination (ESN A2a3). On instances level this is illustrated in Figure 7.2, where improvement up to 100 percent is frequently evident.

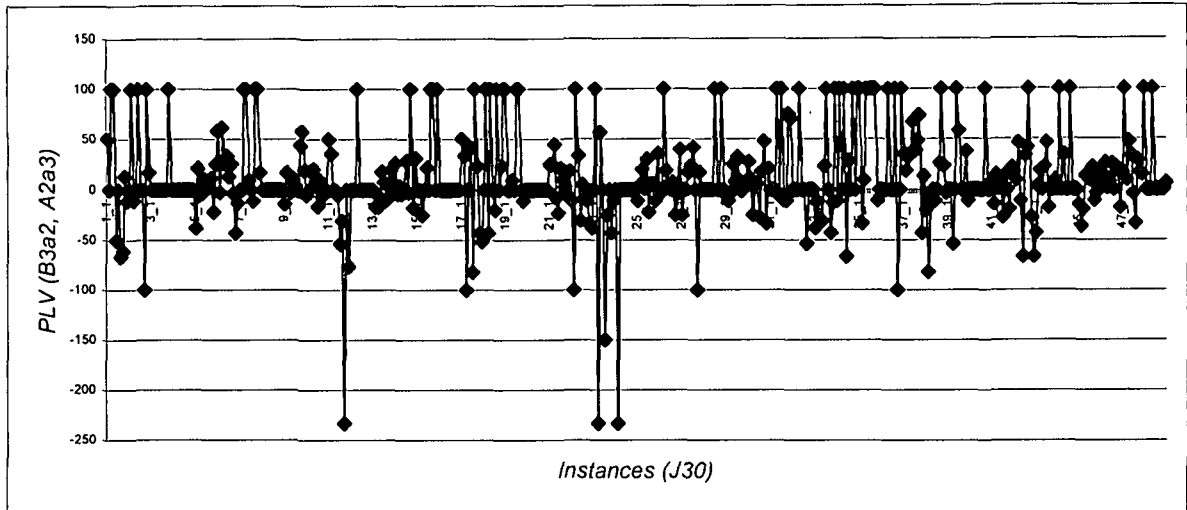


Figure 7.2: Performance comparison between initial set and current (best) set of parameters and modes

With the identified set of parameters and modes having demonstrated improvement of performance till now, we proceed to conduct the remainder of experiments viz. set ESN B4 for evaluating Termination criteria.

The first set of test is for evaluating Fitness-Deviation Termination, and then test the Adaptive Terminations. We test both Mid-Point and Random-Point crossover modes alternately for all the termination techniques.

7.2.4.2 Fitness-Deviation Termination

As per our DoE, we test Fitness-Deviation Termination (FD) with two combinations of (μ , δ and ω) as depicted in Table 7.8.

Deviations	Limits	
	Lower	Higher
σ_{EL} μ	0.01	0.05
σ_{PO} δ	0.10	0.50
σ_{GE} ω	0.10	0.50

Table 7.8: Deviation limits for Fitness-Deviation Termination

As fourth criterion, $MaxGen$ is kept at 100, so we terminate processing at 5000 schedules. The experiments are denoted as ESN B4a and B4b with serial numbers

one and two. Relevant results are displayed in Table 7.9, which also includes count of how many instances out of a total of (480 X 10 =) 4800 terminate by which part of the criteria.

ESN	Data-set	Instances terminating on		PAD	APLV (this, ESN A)
		Deviation limits	<i>MaxGen</i>		
B4a1	J30	826	3974	3 562	8 9701
B4a2	J30	3725	1075	3 816	2 4789
B4b1	J60	18	4782	17 453	11 0720
B4b2	J60	4113	687	19 761	-0 6879

Table 7.9: Termination by Fitness-Deviation criteria

Examining Table 7.9, we get mixed signals from the results of this experiment set. When Fitness-Deviation limits are kept tight for both data-sets, the instances tend to terminate in the *MaxGen*, i.e. fourth criteria. This is simply another type of Fixed Generation termination. On the other hand when Fitness-Deviation limits are kept loose the convergence is relatively fast – but on sub-optimal region. In case of ESN B4b2, the convergence led to even worse result as compared to results with our original settings.

One way of avoiding this paradox might be to keep the deviation limits tight, but increase *MaxGen* limit. Our objective of this set of experiments is to check possible pitfall of the method that has a strong tendency of false convergence.

The results justify this tendency, and we therefore seek some other conclusive method for Termination. This leads us to the next set of experimentation where we test evolution of a proposed Adaptive Termination Algorithm.

7.2.4.3 Adaptive Termination

Experiment set ESN B4x5 to B4x18 (where $x = a$ and b alternately) is the specified experiments for evolution of Adaptive Termination Algorithm.

Since we have not yet exactly pinpointed better of the two options for crossover Random-Point crossover and Mid-Point crossover, we alternately test these mode. We keep *MaxPop* constant at fifty, and use *Best-Spouse* Selection mode throughout these experiments. For diversity, we employ both intrusions, with all probabilities kept at 0.3.

Every Resource Constrained Project has three specific characteristics, viz.

1. Project Length (T),
2. Number of Resource under constraint (R), and
3. Project (precedence) Complexity (C).

We now proceed to experiment with different combinations of these three to finally evolve a Termination algorithm that adapts itself to these criteria. We term every stages of evolution as Adaptive Termination, with a serial number at each tail.

7.2.4.3.1 Adaptive Termination 1

For the first stage of evolution, we define *MaxGen* as product of Project Length (T) and number of Resource under constraint (R). For our experiments, we are using J30 (T = 30) and J60 (T = 60) data-set, each with R = 4.

Experiments for J30 data-set are labeled as ESN B4a5 and B4a6, whose processing would terminate at MaxGen = 120 generations, thereby sampling 6000 schedules. Experiments for J60 data-set are labeled as ESN B4b5 and B4b6. Termination criteria for testing these instances would 'adapt' itself and terminate at 240

generations, after sampling 12000 schedules. Results of the experiments are displayed as Table 7.10.

ESN	Data-set	Average schedules	Schedules tested	Average sample proportion % (approx)	Crossover mode	Comparisons			
						Compared with	Average sample proportion % (approx)	PAD	APLV (%)
B4a5	J30	5*10 ⁸	6000	1.2*10 ⁻³	MP	A2a2	1 0*10 ⁻³	2.417	38.2315
B4a6					RP			2.326	40.5571
B4b5	J60	7*10 ¹⁶	12000	1.7*10 ⁻¹¹	MP	A2b2	7.08*10 ⁻¹²	16.169	17.6144
B4b6					RP			16.011	18.4194

Table 7.10 : Termination by Adaptive Termination 1

With the present combinations, we get better result as compared to equivalent experiments in the initial stage, which is depicted as PAD and APLV above. Though very marginal, but there is a consistency in improvement trend when Random-Point crossover mode is utilized.

Next, we test Adaptive Termination 2 by introducing the Complexity Factor, C.

7.2.4.3.2 Adaptive Termination 2

Complexity of a Project is (proportional to) number of possible schedules (M) due to the Project's precedence constraints. This is represented as C in Adaptive Termination 2 relationship. Since M increases steeply with incorporation of additional precedence constraint, we dampen the velocity of such increase by using a logarithmic function : $G = [T * R] * INT[\log_b M]$. For a specific data-set, the variable we examine is the base of logarithm, 'b', the others being 'adapted' as dictated by the Project.

We test two values of 'b', viz. 10 and 20. For our test data-set, calculations shown by Figure 7.3(a) and Figure 7.3(b) produce discrete MaxGen levels depending on integer portion (or characteristic) of the logarithmic segment.

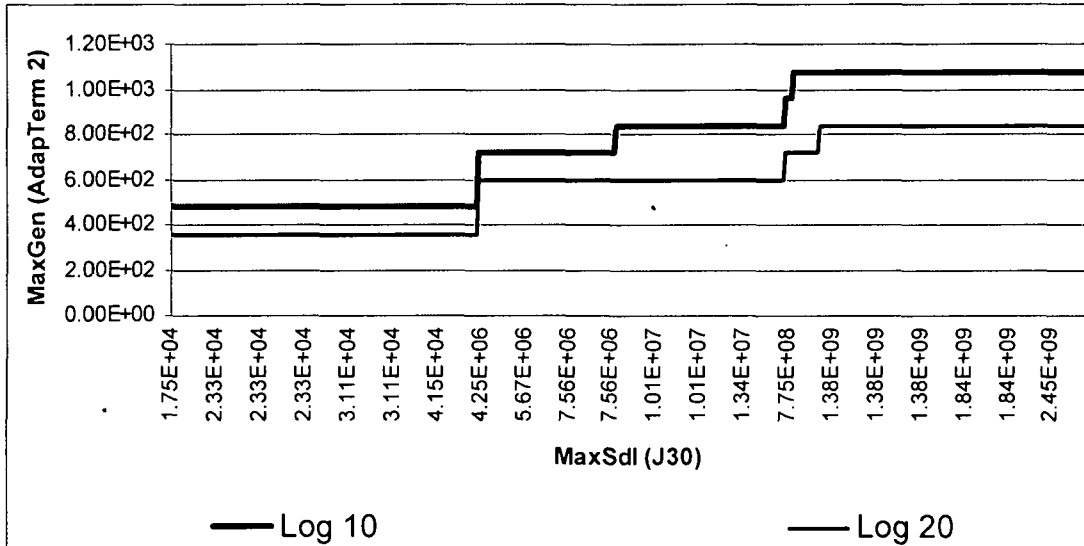


Figure 7.3 (a) : Discrete MaxGen as outcome MaxSdl, Adaptive Termination 2 (Data-set J30, sorted on *MaxSdl*)

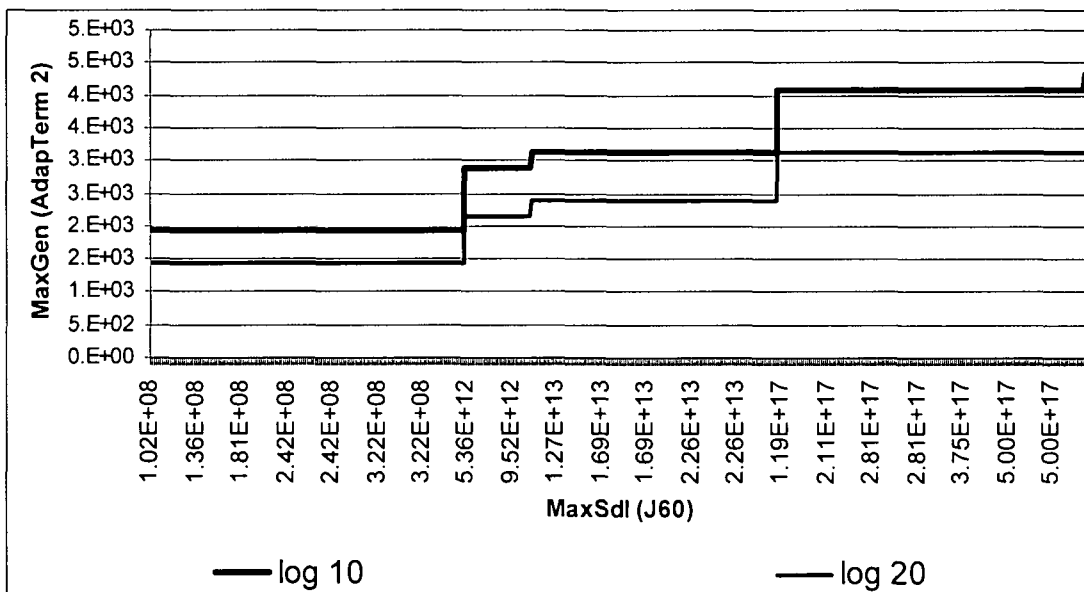


Figure 7.3 (b) : Discrete MaxGen as outcome MaxSdl, Adaptive Termination 2 (Data-set J60, sorted on *MaxSdl*)

The experiment set for *AdapTerm2* is denoted on the Experimentation Plan as ESN B4(a or b)(7 to 10). Test results are tabulated in Table 7.11.

ESN	Data Set	Crossover mode	Base of Logarithm	PAD	APLV (this, eqv of ESN A)	APLV (this, higher gen at ESN A)
B4a7	J30	MP	10	1.981	49.3739	
B4a8		RP		1.912	51.1372	0.3648
B4b7	J60	MP		15.118	22.9702	12.8050
B4b8		RP		14.522	26.0082	16.2439
B4a9	J30	MP	20	2.011	48.6072	
B4a10		RP		1.994	49.0417	
B4b9	J60	MP		15.232	22.3890	12.1471
B4b10		RP		14.823	24.4733	14.5064

Table 7.11: Termination by Adaptive Termination 2

It is obvious that a higher logarithmic base acts as a better damper. A lower logarithmic base results in higher *MaxGen*, and therefore a large sample area is tested. But when compared to the total search area (*MaxSdl*), this sample area gets proportionally smaller. This inverse relationship demonstrates efficiency of our algorithm. In tandem with better efficiency, the PAD inches up favourably on the benchmark result set.

For some of the experiments, our results have started performing better than equivalent initial results. This was to be expected since *MaxGen* have crossed over into the thousands area.

The efficiency of the adaptive algorithm is demonstrated by the following figure, where we plot the Efficiency Index of the instances of the best result, i.e. ESN B4a8.

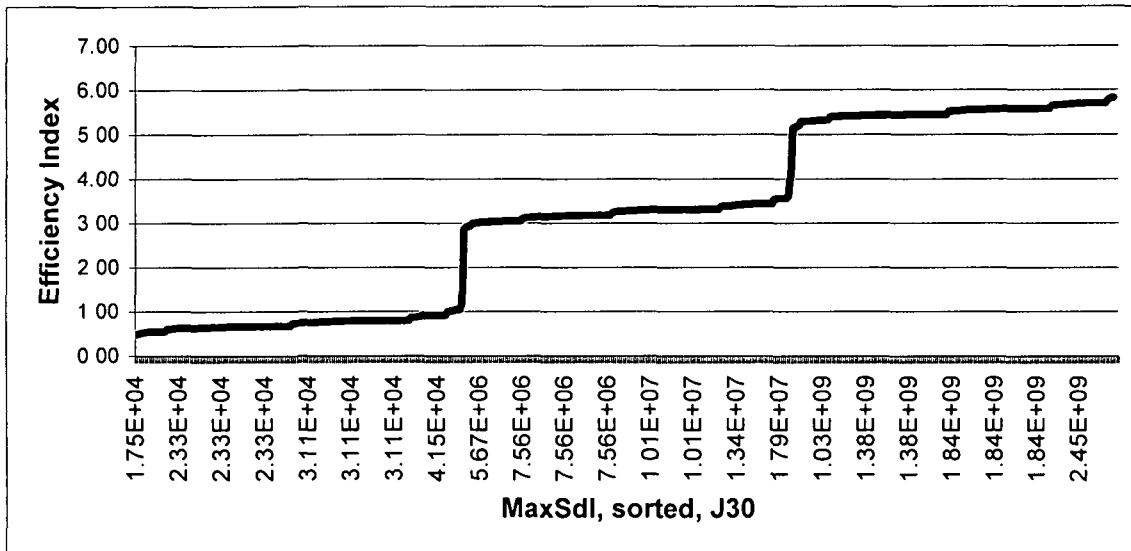


Figure 7.4 : Efficiency Index Vs Project Complexity, Adaptive Termination 2 (Data-set J30, sorted on *MaxSdl*)

Even though we test a proportionally smaller sample set, but within that we not only improve our PAD rating but also the efficiency level. In other words, as complexity of the Project increases we are able to approach optimal solution set with a proportionally increasing level of efficiency employing a relatively slower increase in *MaxGen* size.

For making a comparative analysis, we plot PLV of ESN B4a8, which displays improvement of initial solution, per instance, as Figure 7.5 Even though there are a few worsening cases, but in most of the instances the present algorithm produced hundred percent improvements, i.e. accurate makespan as provided by PSPLIB. The APLV of above 50 implies that the algorithm with this combination has evolved to provide improvement of initial results halfway through towards optimal solution.

This vindicates the assumption that a complex project should be provided with a larger sample set, which is the basis of our adaptive algorithms.

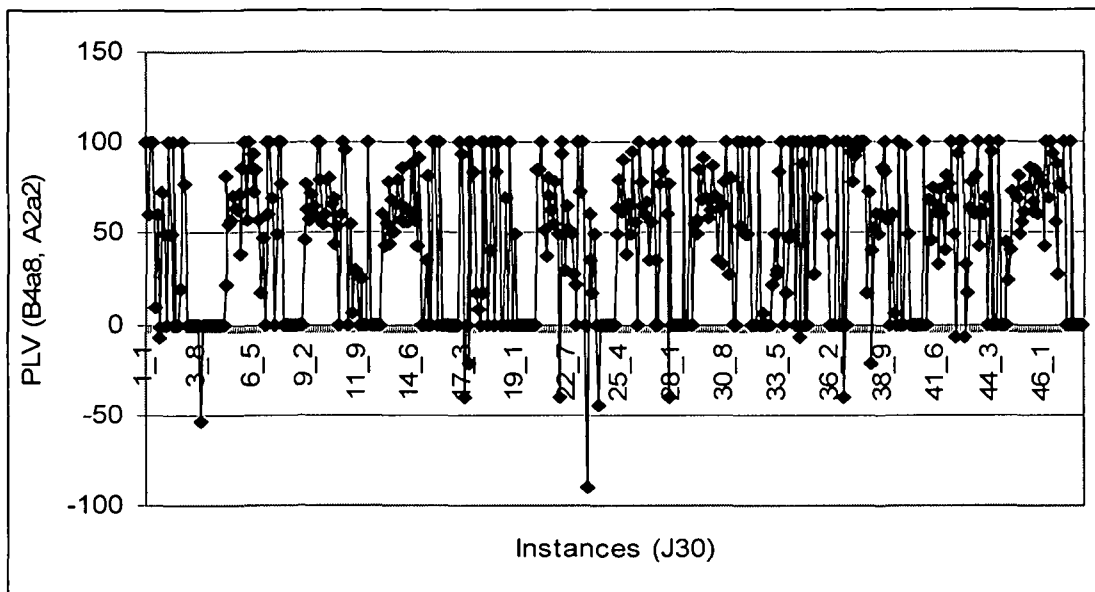


Figure 7.5 : Performance Changes, Adaptive Termination 2 (Data-set J30)

This Termination relationship acts only as a damper, and does not fully respect the exponential increase of complexity. To try to address this lacuna, we proceed to take *AdapTerm2* into another stage of evolution.

7.2.4.3.3 Adaptive Termination 3

Adaptive Termination 3 (*AdapTerm3*) is a combination of exponential and logarithmic functions. The exponential component will try to push up the resultant (i.e. *MaxGen*), but the logarithmic component would act as a damper to that rate of change of velocity. In other words, *MaxGen* would be allowed to accelerate vis-à-vis project complexity, but at a controlled rate.

We use stronger damping as complexity increases. Using Relationship 6.4 for Adaptive Termination 3, we employ 10 as the base of logarithm for J30, and for J60 we use 20. For testing ($\phi - \omega$), we use (5-1) and (6-4) for J30 data-set. Since J60 instances have a higher complexity in the range of 10^8 to 10^{18} we test higher values

of $(\phi - \omega)$, at (10-4) and (10-5) The base of the exponential function is tested at 2 and 1.5 for further damping the exponential function The discrete *MaxGen* as an outcome of use of the above combination is displayed in Figure 7.6(a) and Figure 7.6(b)

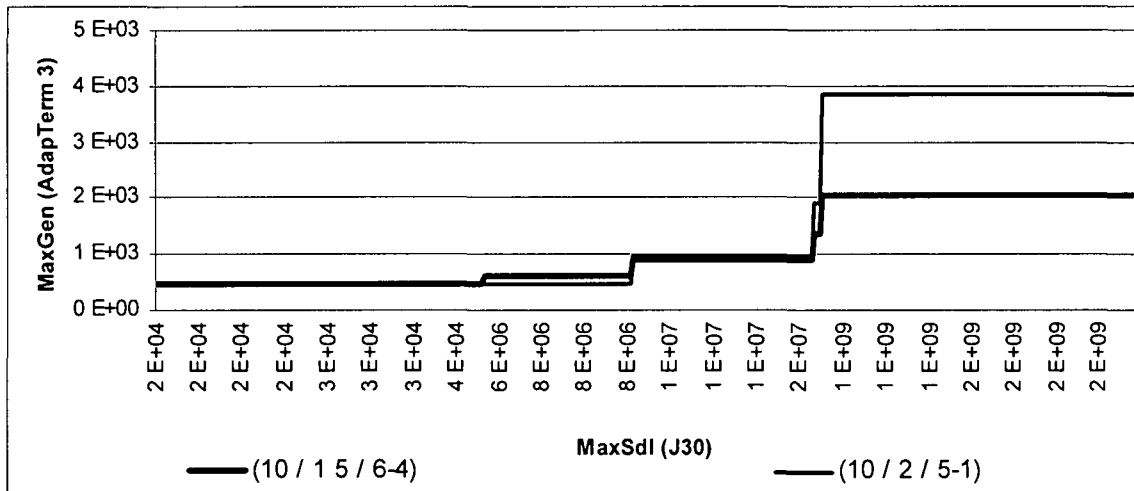


Figure 7.6 (a): Discrete MaxGen as outcome MaxSdl, Adaptive Termination 3 (Data-set J30, sorted on *MaxSdl*)

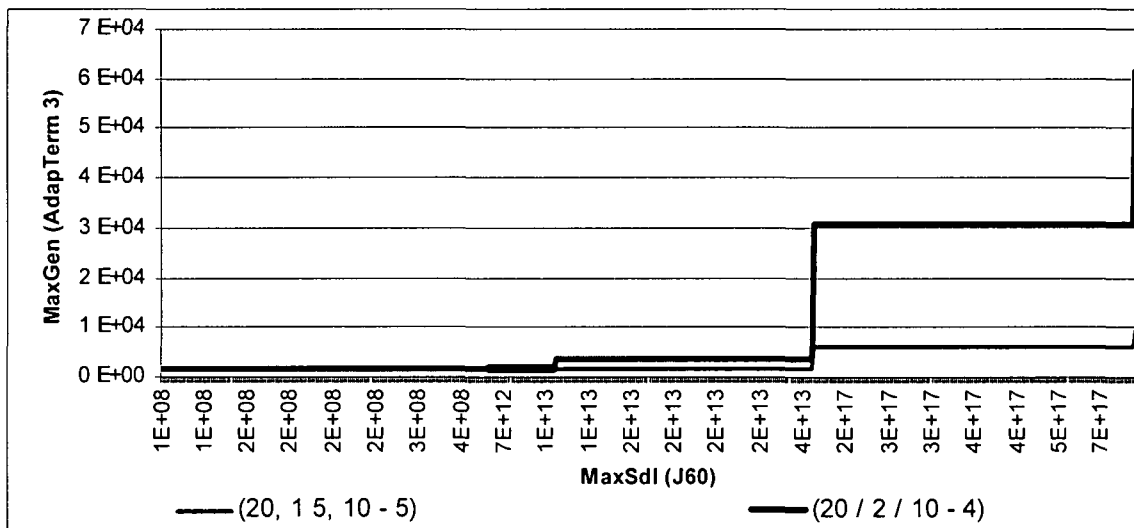


Figure 7.6 (b): Discrete MaxGen as outcome MaxSdl, Adaptive Termination 3 (Data-set J60, sorted on *MaxSdl*)

Since we have not yet made a decision on crossover mode, we alternately test with Mid-Point crossover and Random-Point crossover. We employ both Intrusion methods, setting parameters at $[(0.3, 0.3) \text{ A } (0.3, 0.3, 0.3)]$. For Selection, we employ *Better-Spouse*. *MaxPop* is kept steady at fifty. This set of experiment is labeled ESN B4a11 to 14 and B4b11 to 14, for J30 and J60 respectively.

The experimental results are displayed in Table 7.12 along with the initial results and benchmark results range.

ESN	Data-set	Crossover mode	AdapTerm 3 parameters	PAD		APLV (this, 5000)	APLV (this, 50000)
				Benchmark range	Test Results		
B4a11	J30	MP	(10, 2, 5-1)	0.00 – 2.08	0.028	99.2960	98.5646
B4a12		RP	(10, 2, 5-1)		0.007	99.8195	99.6319
B4a13		MP	(10, 1.5, 6-0)		0.084	97.8430	95.6016
B4a14		RP	(10, 1.5, 6-0)		0.066	98.3213	96.5770
B4b11	J60	MP	(20, 2, 10-4)	10.71 – 15.94	10.997	43.9648	36.5701
B4b12		RP	(20, 2, 10-4)		10.812	44.9079	37.6377
B4b13		MP	(20, 1.5, 10-5)		12.171	37.9859	29.8023
B4b14		RP	(20, 1.5, 10-5)		11.517	41.3179	33.5739

Table 7.12 : Experimental results of Adaptive Termination 3

It is observed that by employing combination as described, the algorithm has achieved the best APLV of 99.8195 for the J30 data-set over results achieved by the initial settings. The present result achieved on PAD evaluation is comparable to the best of benchmark results.

For the J60 data-set, the best APLV is 44.9079, i.e. almost halfway through towards achieving optimal results. When PAD is compared to benchmark results, the experimental results falls short of the best by about a percent.

If we compare results achieved by the two crossover modes, Random-Point crossover is providing marginally better results than Mid-Point mode.

The efficiency index for the best result is provided in Figure 7.7.

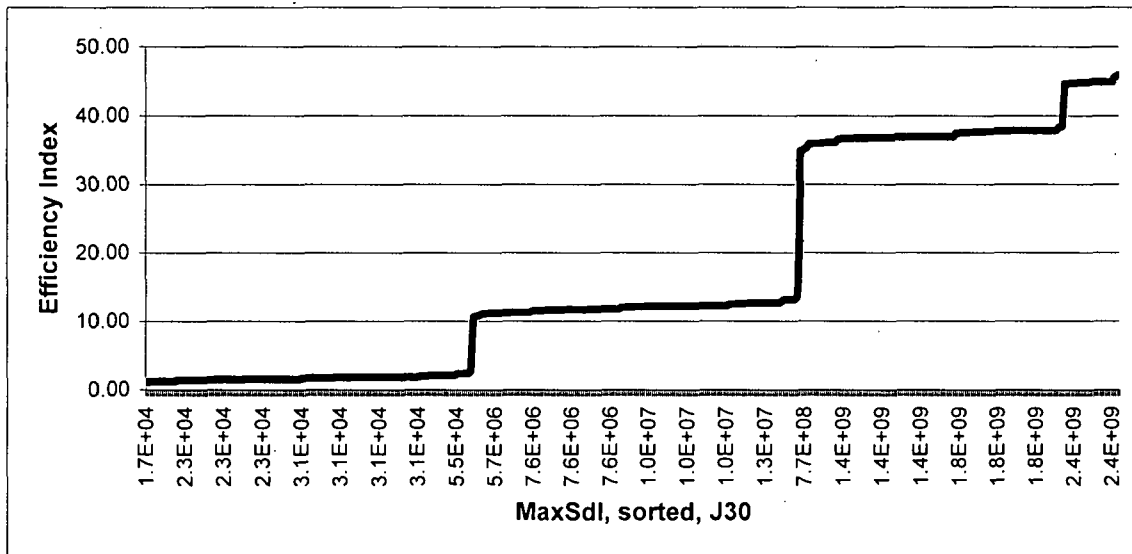


Figure 7.7: Efficiency Index Vs Project Complexity, Adaptive Termination 3 (Data-set J30, sorted on *MaxSdl*)

Compared with Figure 7.4, it becomes evident that when accuracy of result is of prime concern, an increase in sample search area produces better result with higher efficiency. This is the best justification for usage of complexity-proportional termination criteria, which we have already devised as Adaptive Termination 3.

For data set J30, we depict the PIDs of ESN B4a12 in Figure 7.8(a), for comparison with Figure 7.1(a). Similar depiction for ESN B4b12 is shown in Figure 7.8(b).

We can summarize that Adaptive Termination 3 permits exponential increase of sample search space to achieve (near) optimal results, but the permission is granted proportional to the complexity level of the Project.

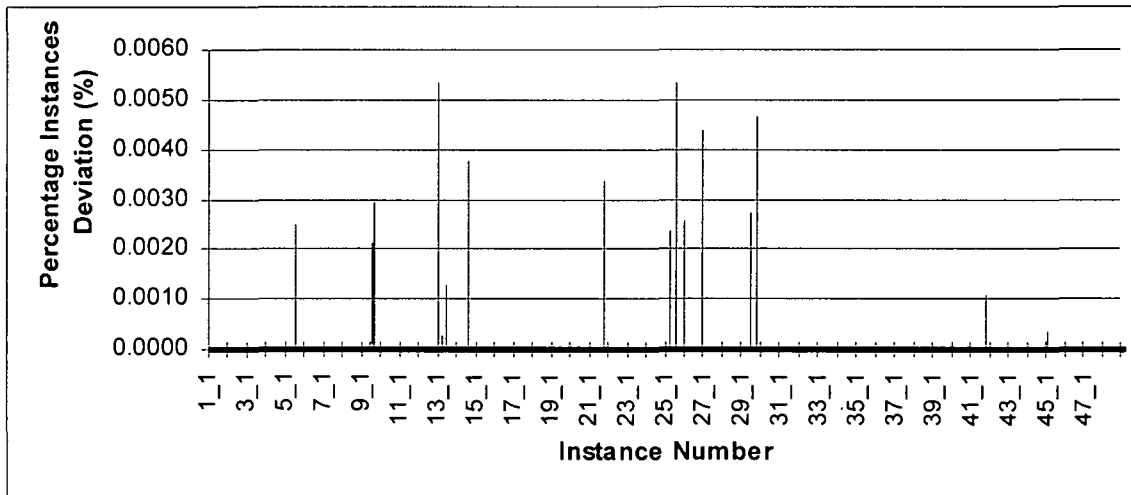


Figure 7.8(a): Summary of deviations, J30, ESN B4a12

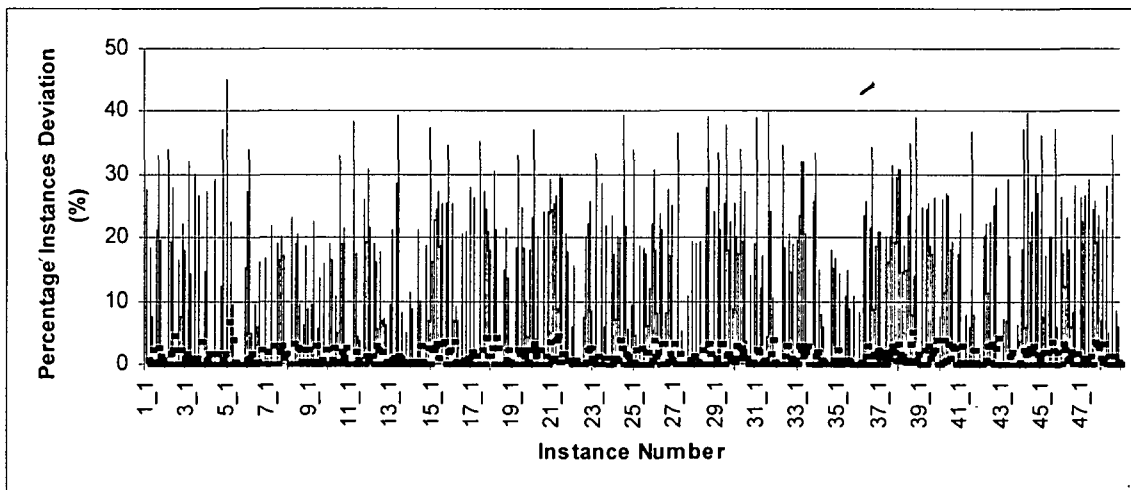


Figure 7.8(b): Summary of deviations, J60, ESN B4b12

It is distinctly clear that the present combinations of modes and parameters have achieved significant improvement as compared to the initial settings. While comparing with benchmark results, the test results shows that the current settings have managed to push the experimental algorithm quite substantially up the optimal results range.

7.3 Comparison with Published Results

In Table 7.13 (a) and (b) comparison of results are made on equitable information collated from benchmark results from published literature, notably Kolisch and Hartmann(2006). In some other literature, authors have reported results in different formats, using self-defined comparison parameters. These could not be benchmarked due to the inherent diversity, and/or are not being benchmarked by the OR community in contemporary literature. As such, those results are not considered for comparison here.

In the two tables below, we have incorporated our best result in similar tabular format as published. The tables are truncated at fifteen entries, with the topmost being the best. The initial test results of the present work are included at the bottom row of both tables.

Sl No	Author	Year	Algorithm	SGS	Average Deviations % from optimal Makespan		
					Termination criteria ##		
					1,000	5,000	50,000
1	Ranjbar, et al	2007	Scatter Search	Serial	0 10	0 03	0 00
2	Kochetov, Stolyar	2003	GA, TS	Both	0 10	0 04	0 00
3	Debels, et al	2006	Scatter search	Serial	0 10	0 04	0 00
4	Present Work	2008	GA (BATGA)	Serial	0 007 (Adaptive Termination)		
5	Kemmoe, et al	2007	PSO		0 26	0 21	--
6	Debels, et al	2004	Scatter search	Serial	0 27	0 11	0 01
7	Valls, et al	2008	GA, hybrid	Serial	0 27	0 06	0 02
8	Valls, et al	2004	GA	Serial	0 34	0 20	0 02
9	Alcaraz et al	2004	GA, FB	Both	0 25	0 06	0 03
10	Alcaraz, Moroto	2001	GA, FB	Serial	0 33	0 12	--
11	Tormos, Lova	2003	Sampling, LFT	Both	0 25	0 13	0 05
12	Nonobe, Ibaraki	2002	TS, activity list	Serial	0 46	0 16	0 05
13	Tormos, Lova	2001	Sampling, LFT	Both	0 30	0 16	0 07
14	Hartmann	2002	GA, self adapting	Both	0 38	0 22	0 08
15	Hartmann	1998	GA, activity list	Serial	0 54	0 25	0 08
Initial Test Results of Present Work without AdapTerm, etc					5 565	3 913	1 919

Table 7.13 (a): Comparison with benchmark results : J30

Termination Criteria Max Schedules // Arranged as sorted on last column

Sl No	Author	Year	Algorithm	SGS	Average Deviations % from optimal Makespan		
					Termination criteria ##		
					1,000	5,000	50,000
1	Ranjbar, et al	2007	Scatter Search, FBI	Serial	11 59	11 07	10 64
2	Debels, et al	2006	Scatter search, FBI	Serial	11 73	11 10	10 71
3	Valls, et al	2008	GA, hybrid, FBI	Serial	11 56	11 10	10 73
4	Kochetov, Stolyar	2003	GA, TS	Both	11 71	11 17	10 74
5	Valls, et al	2004	GA, FBI	Serial	12 21	11 27	10 74
6	Present Work	2008	GA (BATGA)	Serial	10 812 (Adaptive Termination)		
7	Alcaraz et al	2004	GA, FB, FBI	Both	11 89	11 19	10 84
8	Hartmann	2002	GA, self adapting	Both	12 21	11 70	11 21
9	Hartmann	1998	GA, activity list	Serial	12 68	11 89	11 23
10	Tormos, Lova	2003	Sampling, LFT, FBI	Both	11 88	11 62	11 36
11	Tormos, Lova	2003	Sampling, LFT, FBI	Both	12 14	11 82	11 47
12	Alcaraz, Moroto	2001	GA, FB	Serial	12 57	11 86	--
13	Tormos, Lova	2003	Sampling, LFT, FBI	Both	12 18	11 87	11 54
14	Bouleimen, Lecocq	2003	GA, activity list	Serial	12 75	11 90	--
15	Klein	2000	TS, activity list	Serial	12 77	12 03	--
Initial Test Results of Present Work without AdapTerm, etc					22 967	19 626	17 338

Table 7.13 (b): Comparison with benchmark results : J60

Termination Criteria Max Schedules // Arranged as sorted on last column

For reporting the best results of the present work, we have not segregated it according to the tabled termination criteria. The reason being that our termination criteria is Adaptive Termination 3, which is made flexible proportional to complexity level of the project instance – the very foundation of the GA model presented in this work.

Moreover, the entry of the best results of present work is kept at a modest distance from the top, by displaying precision of results upto third place after decimal. As evident, we have compared our results only with 50,000 termination criteria entries of other authors. The reason being that Adaptive Termination 3 analyses a proportionally higher number of solutions for complex projects.

We have restricted analysis of results for present work only to simple tests as depicted in a previous Chapter. As mentioned there, these tests are modeled in line with the widely utilized test parameters of Kolisch and Hartmann(2006). The reason being that it allowed us compare results from a wider pool of published results.

7.4 Presentation of the Algorithm

In several stages with mix-and-match of different combination of modes and tuning of parameters, the algorithm is now in a position to be proposed as outcome of the present work.

From the ESN B4 set, the experiments with the best results are run again multiple times for checking repeatability. The combination corresponding to ESN B4a12 for data-set J30 and ESN B4b12 for data-set J60 has the overall best performance.

Based on these findings and related analysis, we present the proposed algorithm for optimizing resource scheduling and allocation of projects summarized as follows :

- A) Sequencing : Serial, forward sequencing
- B) Schedule Generation Scheme : *Sweep-Creep* (serial, forward scheduling)
- C) Optimization Approach : Genetic Algorithm, with its components –
 - i) Fitness : Sequence and makespan based *UnoSign*
 - ii) Selection : *Better-Spouse* Selection, based on *UnoSign*
 - iii) Crossover : Precedence-Set crossover, Random point
 - iv) Diversity :
 - a) Immigrant, with suggested probabilities (0.3 * 0.3)
 - b) Dormant-Forefather, with suggested probabilities (0.3 * 0.3 * 0.3)
 - v) Termination : Adaptive Termination 3, for maximum generations, viz.

$$G = \lceil T * R \rceil * \beta^{\lfloor \text{INT}(\log_b M) - (\phi - \omega) \rfloor}, \text{ for } M \geq 10^\phi$$

$$= \lceil T * R \rceil * \text{INT}[\log_b M], \text{ for } M < 10^\phi$$

where

G : *MaxGen*

T : Project length

R : Constrained Resources

M : *MaxSdl*,

β : Base of the exponential function,

b : Base of the logarithm used, and

ϕ : A limiting factor

ω : A small non-negative integer.

Because of two distinctive features, viz.

- a) The *Better-Spouse* Selection operator, and
- b) Adaptive Termination

we propose to term proposed Genetic Algorithm as “*Better-Spouse Selection and Adaptive Termination Based Genetic Algorithm*”, or BATGA.

This adaptation of Genetic Algorithm, or BATGA, has facilitated optimization of the RCPSP up to a level at par with comparable contemporary results. However the algorithm falls short of matching the best of published results, which is a fair indication that further tuning of components and parameters of BATGA has to be carried out.

Chapter Eight

Conclusions

In this concluding Chapter we provide a summary of work carried out, including that of our initial background study. After that, a short discussion about strength and shortcomings of the proposed algorithm is presented. Based on the work and experience gained, we thereafter hint at possible contribution to the knowledge domain.

A work of this nature can never be conclusive. Keeping that in mind, we chart out a few area and directions, towards which we shift focus where further studies might be carried out.

8.1 Summary of Research Issues

In the total course of our research, our focus was primarily on an attempt to provide a as much possible a realistic solution to a problem area that is universal in its proliferation. Within the area of work, we took up an approach that was then modified for its applicability. Within the demarcated boundary, our extent of work is of computational experimentation of the proposed algorithm. With appropriate modesty we acknowledge that contributions from other researchers have been the foundation of our work. On such strong foundation, this work is just a modest attempt at furthering research in the chosen field.

Project Management is a mission involving high degree of adverse probabilities. Conforming to Murphy's Law, – *if anything can go wrong, it will* – completion of a Project within budgeted time and cost frame is a Herculean task.

The Project Manager is in the unenviable position of trying to manage the limited available resources to deliver the Project within its framework. But adversities work overtime to make sure that his efforts gets pulled to the extremes, and even after that results are achieved at a heavy price. One area where most of the Project Manager 's time and energy goes in is reworking of Project schedules. Once a schedule is in place, theoretically the work should proceed as per plan. But factors, usually exogenous, disrupt the plan, and it requires reworking. The only ally with the Project Manager in such a situation is the software that aids scheduling.

But software is as good as the algorithm on which it runs. Therefore constant upgradation of the mechanics of the algorithm is one way of assisting the Project Manager in his work.

Scheduling of Projects with constrained resource has been studied from multiple fronts since the advent of mathematical 'optimization'. With the advent of newer optimization techniques, Resource Scheduling is one area where it is applied to arrive at better solution. For Project Management, the problem that deals with

constrained resource scheduling is termed as the Resource Constrained Project Scheduling Problem, or RCPSP. Since the 1950s numerous researchers using contemporary optimization techniques have attempted to arrive at optimal algorithm for tackling this problem. Being a NP-Hard problem, the optima is elusive, and welcomes any betterment of existing algorithm.

With the advent of heuristic techniques, Operations Research community has discovered a totally new dimension of solution approach. One such approach, Genetic Algorithm, has been used for RCPSP with multiple adaptations of the original concept. Ever-new metaheuristics are evolving to design algorithm for taking RCPSP as near as possible towards the optima. This sequence of background study helped us identify the exact problem, and the approaches made for optimizing it.

In this work, we present adaptations of proven and robust Genetic Algorithm components, and have incorporated some innovative measures. For this, we studied the mechanism of Genetic Algorithm and operational intricacies of the different components. Having identified short but critical gaps where the methodology may be adapted and modified, we proceed to do so – first as an algorithm, and thereafter implementing it for experimentation. During algorithm design and development, we kept options open for evolving it to the final state.

For validation of our algorithm, we proceed to run the program on internationally accepted benchmark test data-set. As compared to initial settings, the algorithm is made to evolve to its presentable stage. At this stage, the results achieved healthy improvement as compared to initial ones. The final results nearly matches the leading results available in literature. Analytically, the algorithm proved its worthiness on three fronts – effectiveness, accuracy and efficiency.

Acknowledging two distinct characteristics of the algorithm, the proposed model is termed as “*Better-Spouse Selection and Adaptive Termination Based Genetic Algorithm*”, or BATGA.

8.2 Strength and Shortcomings of the Algorithm

Experimentation has shown that BATGA is quite effective in its objective to provide effective, accurate and efficient convergence, almost at par with the best set of results published. The adaptive nature of the algorithm is its best strongpoint.

Selection is an important component of Genetic Algorithm. BATGA incorporates a technique where it deliberately selects a *Better-Spouse* for mating. This technique has proved its worthiness by providing very strong positive results.

With correct setting of program parameters, the algorithm has proved its ability to test different data-set. Limitations crept in due to computational constraints, and on a stronger platform, this algorithm has a capability to process complex Projects with even tighter constraints.

In contrast to a few other proposed algorithms in literature, BATGA minimizes ‘wastage’ of computational effort. The results are deliberately made to be feasible, and thereby eligible to be evaluated. Of course, keeping in view the requirement of a Genetic Algorithm, ‘unfit’ results are discarded. A feature of BATGA allows random probability based recall of such previously discarded results, thereby possibly reducing computational effort even further.

In spite of the positive vibes, BATGA is not devoid of shortcomings. We question our own claim regarding ‘uniqueness’ of *UnoSign*. We have claimed that *UnoSign* provided actionable uniqueness of schedules – but that is purely based on program output. The uniqueness was demonstrated when we calculated *UnoSign* by

independently generating all possible schedules of some of less complex instances. But that was only a limited empirical outcome.

What we assume to be the strongpoint of BATGA is also its weakest defense. Conceptually, an adaptive algorithm would provide results with increasing efficiency. But the adaptive function itself has to prove its robustness in face of adversity.

8.3 Contribution to Knowledge Base

The RCPSP, by its very nature, invites ever-new angle for optimization. Genetic Algorithm, the approach which we have selected, encourages constant modification and adaptation of its basic structure. The present work has attempted a few specific contributions to these knowledge domains – both apparently interlaced.

8.3.1 *Sweep-Creep* SGS

In BATGA, a modified serial, forward scheduling SGS is put forward and termed as *Sweep-Creep* mechanism. While scheduling, this mechanism constantly matches resource requirement of activity being scheduled with current position of resource availability.

8.3.2 *UnoSign* based Fitness Value

In many literatures, duplicate solutions were encountered amongst schedules of similar makespan. Since makespan is the decisive fitness check for the RCPSP, this duplication led to discarding of feasible solution and/or devising complex fitness functions.

BATGA employs a unique signature – *UnoSign* – for each solution generated within, which is derived from activity sequence and makespan of each schedule. Derived only once, *UnoSign* reduces calculations of other factors later on, thereby lessening demand on computational resources. *UnoSign* is exploited in many locations of BATGA, viz. cloning of parents, selection for mating, skimming off of elites (or solution set), and survivor selection for next generation.

8.3.3 *Better-Spouse Selection*

A modified version of Russian Roulette Selection is put forward as component of BATGA. We call this *Better-Spouse Selection*, where the algorithm deliberately forces the selection process to chose a stronger mate for the first one. Therefore, crossover can never take place between parents of equal strength – a drawback that would otherwise waste processing effort of the algorithm.

8.3.4 *Adaptive Termination, AdapTerm3*

Most of Genetic Algorithm adaptation for RCPSp runs with fixed number of generations, overlooking the complexity level of the Project, etc. For example, consider a Genetic Algorithm with fifty as population size and run it for a thousand generations. Some (Project) instances of J30 data-set with a maximum of about eighteen thousand feasible schedules makes use of such Genetic Algorithm meaningless. On the other end of the scale would be instances that would be as complex as having about 10^8 schedules.

A feature of BATGA addresses this uneven distribution of sample space. We introduce an adaptive termination for the Genetic Algorithm, which is dependent on three crucial Project characteristics – the number of activities, the number of resources under constraint, and complexity level of the Project. Using a combination of Exponential and Logarithmic functions, an algorithm is presented that allows the Genetic Algorithm to adapt to Project parameters. This we term as

AdapTerm3, which assists BATGA to decide by itself the maximum generations the process should run. *AdapTerm3* mechanism runs an accelerating-damping combination to change maximum generations in proportion to input information of all the three project characteristics.

8.4 Directions for Further Research

Despite promise of wide scope, the present work is only a limited effort within the field of study. We can view this wide picture from three perspectives – assistance for the Project Manager on the macro level, the RCPSP in the micro level, and Genetic Algorithm as the approach.

On all of these fronts, a number of issues remain untouched by our work, and many of those touched has ample possibility for future research for improvement on results achieved. Our experience shows that in spite of extensive amount of reported work, there is still scope for improvement and contribution.

8.4.1 The Macro Perspective

Software usage for Project Management can go only one way – up. A number of possibilities exist where studies can be carried out in Business Informatics for application in this field. We have focused only on (constrained) resource scheduling, which aids the Project Manager in deciding on delivery date by optimal management of resources. Each item in the list of Project Management Knowledge Area of the PMBOK would trigger other area of study. As evident from Table 1.1 and Table 1.2, the present study has operated in a small niche of a section of that list.

The present work is on managing a single project, since every instance of the test data-set represents a project. Or for that matter, a study whose result would improve decision making by the Project Manager for the Project under

jurisdiction. When we view the picture from a higher level, an organization (be it the Government or a Corporate house) has a number of Projects under its fold. Each with its own characteristics, maybe on opposite side of the spectrum. This bouquet is Project Portfolio of the organization. We invite attention towards pioneering conceptual work initiated globally in the area of Project Portfolio Management. A further extension is in the area of Program Management, which is in a relatively nascent stage. Here the Project is considered as a module of a Program, i.e. we get a even wider field. These are the macro avenues where exploration has to be made to seek incorporation of the present work as a component.

8.4.2 The Micro Perspective

Because of its NP-hard nature, the RCPS \bar{P} was and would be subjected to research for years. Right from the days of exact methodologies till the newest adaptation of Ant-Colony Optimization, an algorithm to achieve full optimality has eluded the RCPS \bar{P} . Different methodologies for optimization of the components have been tried, but the 'whole' seems to be far away.

Schedule Generation Schemes are theoretically limited in generic number. But variation within the set would always be a potential hunting ground. We have implemented a serial, forward scheduling mechanism (the *Sweep-Creep*). It is obviously a limited use, and other variations would have to be experimented upon.

At every point of scheduling, the *Sweep-Creep* mechanism keeps checking current availability of resources. This is one feature we tried to exploit for making the program respond to dynamic change in resource availability. The concept is presented as Appendix III. We expect to carry forward the present work in that direction, as one its extensions.

8.4.3 The Approach

For approaching optimality of RCPSP we have employed Genetic Algorithm. Operations Research brethren are employing hybrids for achieving better results. Hybrid Genetic Algorithm of Valls, et al and Genetic Algorithm – Tabu Search combination of Kochetov and Stolyar (2003) have proved this possibility. We invite attention for hybridizing BATGA in toto or components thereof for bettering the present set of reported RCPSP results. One suggested way to speed up the search would be by employing Ant-Colony Optimization (ACO) in tandem with BATGA. This hybridization should yield good results especially when the search area is large, and multiple sub-optimal peaks exist. Once the ACO hints at the feasible zone, BATGA would take over for homing in on the solution. This way we can avoid the redundancies of Genetic Algorithm, i.e. generating and discarding sub-optimal solutions.

Akin to Fuzzy Genetic Algorithm of Pan and Yeh (2003), a hint of using Neural Networks for making the algorithm ‘learn’ has been made in the concept paper at Appendix II. The *AdapTerm* 1, 2, and 3 of BATGA displays an interesting characteristic of the Natural world, that they seem to act as neurons but in its primitive stages of evolution. If proved so, and thereby improved upon, we feel that a hybrid system of Neural Networks with BATGA would deliver a potent combination for practical application of the proposed algorithm for aiding the Project Manager.

The mathematical relationships of BATGA have been formulated by experimental tuning of parameters. The structure of the relationships, especially that of the fitness value, *UnoSign*, and termination criteria, *AdapTerms* 2 and 3 begs for mathematical validation.

For *AdapTerm3*, we have utilized a Logarithmic damper on an Exponential accelerator function. To establish credible design and utility of adaptive

termination, it would be worth experimentation using other combinations. A possible combination is a Polynomial damper on an Exponential accelerator.

For correlation between parameters and sensitivity study of impact of those, we have used very simple relationships for comparison with benchmark results. Detailed statistical analysis, specifically ANOVA and Sensitivity Analysis, can be carried out for further validation of the algorithm. This approach of analysis would provide further integrity of performance of BATGA.

Due to lack of plausible proof, we have questioned our own claim of uniqueness of *UnoSign*. Similarly, *AdapTerm* parameters are provided within limited exposure. Theoretical validation of the parameters of these two critical components is an area where we invite attention. Additional study can be made only for empirical verification of the structure and values of variables thereof.

Finally, field-testing of the algorithm using real data would prove its worthiness for practical usage in Project Management software. Since our study was initiated by exploring at the macro level, we do not wish to remain complacent now that our algorithm has provided good results. Future work for validating robustness of BATGA is envisioned by putting it to process real data of Projects – on both inline as well as off-line mode.

8.5 Practical Utilization of BATGA

The commercial market has a number of Project Management software. Notable amongst them (*in their latest individual versions*) would be MS Project, Primavera, Scitor Project Scheduler, etc. In a seminar paper at the Johannes Kepler Universität Linz, a comparison of seven Project Management software was made by Mühlbauer, et.al(2007), although not benchmarked. On our last access, Wikipedia listed above a hundred Project Management software in their page

http://en.wikipedia.org/wiki/List_of_project_management_software. According to Kim (2007), the demand of project scheduling software has continued to grow at an annual rate of almost 20%.

As mentioned elsewhere in this document, software would be as fast as the algorithm it uses. Mellentien and Trautmann (2001) by their tests on commercial project management software – for performance, revealed a considerable performance gap between the implemented method and state-of-the-art project scheduling algorithms. When matched with research results of those days, they concluded that there was still a significant potential for improving solutions to resource allocation problems in practice. To maintain parity with majority research, these tests were carried out on the PSPLIB instances. An indication of the gap is provided in Table 8.1 by comparison of their findings with contemporary literature results at that time.

Author / Software	Year / Version	Mean Deviation % (J30)
Alcaraz, Moroto	2001	0.12
Tormos, Lova	2001	0.07
Hartmann	1998	0.08
Acos Plus.1	8.2	3.87
CA SuperProject	5.0a 002	5.39
CS Project	3.0	3.50
MS Project	2000	5.18
Scitor Project	8.0.1	4.85

Table 8.1: Gap between literature result and commercial implementation (2001)

In the concluding remarks, Smith (2004) observes that the best measure of the effectiveness of a scheduling algorithm should come from the industry. In light of this, the outcomes and addition to the body of knowledge by the present work is now dependent on incorporating them into PMIS, for use by Project Managers of the real world. With all humbleness, the algorithm proposed in the present work – BATGA – is put forward for possible inclusion in Project Management software.

The present work have implemented an adaptation of Genetic Algorithm for approaching optima of the RCPSP as an attempt to formulate an optimized algorithm for resource scheduling and allocation in Projects. Within its boundary limitations, the approach achieved respectable results when compared with benchmark results. While this might sound good, there is a considerable amount of work yet to be done for making the BATGA robust and universally acceptable.

We rest with the likelihood that whatever little contribution the present work made into the knowledge domain to try mitigating uncertainty of Project Management would invite attention for scrutiny and further study.

Appendix I

Glossary

We present a short glossary of terms specifically coined and/or used in this work.

AdapTerm	Adaptive Termination; using features of input information; of the input project instance.
AdapTerm 1	Adaptive Termination using number of project activities, and resources under constraint
AdapTerm 2	Inclusion of project complexity information and damping function to AdapTerm 1
AdapTerm 3	Inclusion of accelerator function to AdapTerm 2
Alien	A freshly generated individual which would try to infiltrate the population (except the initial population)
APID	Average Percentage Instance Deviation for individual instance; comparative measure
APLV	Average Performance Level Variation; comparative measure between different parameter settings; for same instance
Better-Spouse	Selection for mating as used in BATGA; the second 'spouse' will always have better fitness value than the first.
Copy	Number of times a parent is eligible to be selected for mating; also number of maximum Offspring the 'better-spouse' can have if BATGA use Random-Point crossover.
Dormant-Forefather	Unary operator; mechanism for introducing diversity into a population; analogous to the <i>Egyptian Mummy</i>

Egyptian Mummy	A retained solution which was to be otherwise discarded, for possible revival later on; analogous concept of
EI	Efficiency Index; accuracy for deviation as compared to proportion of schedules sampled in an instance
Immigrant	The alien, after it receives favourable response to first of the two intrusion probabilities
Infiltrate	Movement of an alien or a Dormant Forefather into a population; after overcoming probability barrier.
MaxPop	Population size in a generation, including that of Initial population.
MaxSdl	Maximum number of schedules possible for a specific instance; indicates Complexity level of the Project
Mummify, Mummification	Conversion of a possibly weak, about to be discarded solution, into a Dormant Forefather; retaining the Dormant Forefather (for future revival)
PAD	Percentage Average Deviation; comparative measure; comparison with benchmarks
PID	Percentage Instance Deviation for individual instance; comparative measure
PLV	Performance Level Variation; comparative measure between different parameter settings; for same instance
Sweep-Creep	Scheduling mechanism
UnoSign	Fitness Value of schedules; unique signature; calculation based on sequence of activity and constrained makespan

Appendix II

Published Paper

This paper was proposed to be a backgrounder for developing an algorithm to streamline project planning and implementation strategies by discussing possible adaptation of tools and techniques used by the world of Information Technology.

The present work is an extension of this paper, and has addressed a portion of the total road-map.

Towards an Optimized Algorithm for Scheduling and Resource Allocation of Infrastructure Projects.

(A Background Study)

Tridib R. Sarma

Abstract :

The field of Project Management (PM), because of its visible impact, has shown an extraordinary growth internationally, especially in the developed nations. Project Management tools and techniques, and the related Information System - developed especially for Engineering Projects, forms an integral part of modern Project Management. But in the Developing and Underdeveloped nations, the emphasis on professional PM is relatively less. This has resulted in massive over-expenditures and resource loss due to Time & Cost overruns. Even the developed nations are not totally free from such mismanagement. In most cases the improper management and execution of schedules is the major culprit. It becomes very much evident that project overruns shows its presence and impact on the macro level. However its impact on many fronts can be gauged, controlled and remedied to a great extent at the micro level. The most important aspect for this impact control and management is the Project Monitoring System. Operating this in tandem with professionally laid down project schedules and correct resource allocation for the scheduled jobs in many cases leads to a higher level of streamlining of Project Implementation.

Keywords : *Project Management, Algorithm, Scheduling, Resource Scheduling, Resource Constraints*

1. Infrastructure Management

Businesses, industries, organizations and nations of every size and focus are counting on professionally managed project management skills to make them succeed in the ever more competitive global marketplace. Infrastructure Management and Development, along with the related aspects of Engineering Project Management is an area that has a direct bearing on the National as well as International scenario for sustaining the economic as well as Industrial growth of the present day. The importance of this area can be gauged from the fact many countries have bodies to monitor works of Infrastructural importance. Both Government as well as the Industry establishes such bodies. India has a full-fledged Union Ministry with the name of Ministry of Project Implementation (MPI) to monitor macro projects. On the international front, the leading body is the UNIDO. The professionals coming under the purview of this field have their own body with the self-explaining name Project Management Institute (PMI), whose headquarters is in Pennsylvania (USA), with Chapters all over the world.

Project Management tools and techniques, and the related Information System - developed especially for especially for Engineering Projects, forms an integral part of modern Project Management. Because of its visible impact, the field of Project Management has shown an extraordinary growth internationally, especially in the developed nations. Individuals skilled in the field of Engineering and Management are gradually gravitating towards the development and management of Infrastructural Projects.

2. Infrastructure (mis)management in developing countries

However in the developing and under-developed nations, the importance attached to this area, especially by the professionals and powers to be, is relatively low. This is especially true in cases of Government/Semi-Government and PSU Projects. Even a small-scale project cannot be completed within the budgeted Time/Cost frame. This results in massive cost overruns - running into thousands of crores of taxpayer money. And high time overruns - running into years, which in turn is translated into cost overruns. To stay on top, new projects and business development must be completed quickly, in time and within cost budget. Failure on any of these fronts would result in

massive overruns of the two most important resources - time and cost. This gets negatively reflected on the business of a firm, position of the related industry and the economy of the nation as a whole.

In case of India, a report of 1998 cites the fact that till then, the country has lost over Rs. 45,000 crores due to cost and time overruns in executing major and mega projects - in the Public Sector. This report is based on facts collated and released by the Government as well as FICCI. The major contributors to this dubious distinction are the power, railways and steel sectors - which accounts for around three-fourths of this cost overrun. Citing specific examples of Indian context, even in short would produce volumes, and is left out of the scope of this report. However a few example of neighboring nations are provided.

At this juncture, it would be pertinent to state a paradoxical fact that this data on cost overrun doesn't fully capture the extent of the problem. Paradoxical in the sense that there are certain sectors where the time overrun doesn't get converted to cost overruns. In areas like power and petrochemicals, equipment prices have declined dramatically over the years. So we land up in a peculiar situation where we have time overrun without cost overruns. In such a situation we have to fall back on alternative course of action for converting the time overrun to cost parameters. One convenient and plausible way is to use the concept of Opportunity cost in terms of the profits foregone and extra costs in other ways. For example, take IOC's Panipat Refinery. It had a time overrun of 14 months (at the time of the said report), but no cost overruns, with costs frozen at around Rs. 3,600 crores (including Rs. 800 crores for an associated pipeline project). Yet this delay has meant that the country had to import more petrochemical products for that period. Also, since IOC would have been entitled to a 12% post tax return for the immediate period, when the APM dismantling began, the delay meant that IOC actually had lost out on an additional profit of around Rs. 250 crores, or around 15% of its total profits for the year.

To cite a few examples of project overruns, let us turn our attention towards Myanmar (formerly Burma). A government report in 1999 proudly stated that foreign investments were still flowing for two Hotel projects. One had an original estimate of US\$50M but had till then cost the investors US\$85M. And the other, also estimated at

US\$50M, had cost going up beyond US\$90M. Till the time of reporting, none of the two were completed. Another example of infrastructural project going haywire in Myanmar was yet another foreign investment sponsored natural gas pipeline to Thailand which was experiencing 70-80% cost overruns. It is interesting to note that in this report, the blame is on the inept and inefficient military junta who has 'zero training in economics, finance and investment management'

One example from Pakistan of gross mismanagement of engineering project is the Sandak Project - a mining and smelting endeavor. This project was identified and proposed way back in 1974. It was delayed by two decades due to various operational and administrative hurdles. At one time a Chinese firm entered into the picture and offered its technical skills and expertise to build the project on manufacturer's credit. According to the joint schedule, the project should have been completed by June 1995. It did start its test operations in 1995, but for only 45 days. Due to the lack of a refinery, the blister copper had to be transported to China or Iran to make it marketable. This proved to be its nemesis. The Chinese firm at that point agreed to build a refinery again on deferred payment schedule. But nothing came of it, and after that test run, not a single ton of production have taken place. Experts believe that the Sandak Project, which 'had been grossly mismanaged in the past, can be revived with professional help, adding, it must be saved in the best national interest'.

Even the developed nations are not free from such mismanagement. However the focus of this author's work is on the developing and underdeveloped nations. One can go on citing such examples from the poorer nations.

In most cases the improper management and execution of schedules is the major culprit. But should one blame only those who executes the schedules? The body, which had originally formulated the schedules, does in most cases lay down schedules that are too far away from achievable reality. This is a raw truth as in most cases the person who would finally control the execution of the plans – the Project Manager – is not made a party of the planning phase. Or even if he is made so, during implementation phases, situation crops up which would call for changes in the plans – either subtle or drastic. Couple that with conflicting resource allocation and inadequate monitoring, and one can very well imagine the resulting chaos. A project is an open

system, fully interacting with the environment - both for input(s) as well as output(s). The control and feedback mechanism installed for the project is the only tool that attempts to keep the project within track to proceed towards its logical and physical conclusion.

It is very much evident that project overruns shows its presence and impact on the macro level. However its impact on many fronts can be gauged, controlled and remedied to a great extent at the micro level. For identifying the points where these checks can be incorporated, one needs to go into more details into the relevant aspects within the field of Project Management.

3. Project Management Knowledge Areas

PMI have identified nine areas of Project Management Knowledge. (It is worthwhile to note here that PMI discourages use of the term 'functions' in this context, as the term 'function' has been frequently misunderstood to mean an element of a functional organization.) The nine Project Management Knowledge Areas are :

- i. Project Integration Management,
- ii. Project Scope Management
- iii. Project Time Management
- iv. Project Cost Management
- v. Project Quality Management
- vi. Project Human Resource Management
- vii. Project Communication Management
- viii. Project Risk Management
- ix. Project Procurement Management

Within the scope of this paper, we proceed further into the 3rd and 4th items of the above list, i.e. into Project Time Management and Project Cost Management.

Project Time Management knowledge area identifies five major processes :

1. **Activity Definition** – identifying the specific activities that must be performed to produce the various project deliverables,

2. **Activity sequencing** – identifying and documenting interactivity dependencies,
3. **Activity duration estimation** – estimating the number of work periods which will be needed to complete individual activities,
4. **Schedule development** – analyzing activity sequences, activity durations, and resource requirements to create the project schedule, and,
5. **Schedule control** – controlling changes to the project schedule.

In Fig 1.0 one can identify the specific micro-level position within Project Time Management Knowledge area, understanding and controlling of which ultimately controls the macro impacts.

Figure 1.0 : Project Time Management Overview			
<i>(Source PMBOK)</i>			
Major Processes	1.1.1 Inputs	1.1.2 Tools and Techniques	1.1.3 Outputs
1.1 Activity Definition	<ul style="list-style-type: none"> .1 Work Breakdown Structures .2 Scope Statement .3 Historical Information .4 Constraints .5 Assumptions 	<ul style="list-style-type: none"> .1 Decomposition .2 Templates 	<ul style="list-style-type: none"> .1 Activity list .2 Supporting details .3 Work Breakdown Structure updates
1.2 Activity Sequencing	<ul style="list-style-type: none"> .1 Activity list .2 Product Description .3 Mandatory dependencies .4 Discretionary dependencies .5 External dependencies .6 Constraints .7 Assumptions 	<ul style="list-style-type: none"> .1 Precedence diagramming method .2 Arrow diagramming method .3 Conditional diagramming method .4 Network templates 	<ul style="list-style-type: none"> .1 Project network diagram .2 Activity list updates
1.3 Activity Duration Estimating	<ul style="list-style-type: none"> .1 Activity list .2 Constraints .3 Assumptions .4 Resource requirements .5 Resource capabilities .6 Historical information 	<ul style="list-style-type: none"> .1 Expert judgement .2 Analogous estimation .3 Simulation 	<ul style="list-style-type: none"> .1 Activity duration estimates .2 Basis of estimates .3 Activity list updates
1.4 Schedule Development	<ul style="list-style-type: none"> .1 Project network diagram .2 Activity duration estimates .3 Resource requirements .4 Resource pool description .5 Calendars .6 Constraints .7 Assumptions .8 Leads and lags 	<ul style="list-style-type: none"> .1 Mathematical analysis .2 Duration compression .3 Simulation .4 Resource leveling heuristics .5 Project management software 	<ul style="list-style-type: none"> .1 Project schedule .2 Supporting details .3 Schedule management plan .4 Resource requirement updates
1.5 Schedule Control	<ul style="list-style-type: none"> .1 Project schedule .2 Performance reports .3 Change requests .4 Schedule management plan 	<ul style="list-style-type: none"> .1 Schedule change control system .2 Performance measurement .3 Additional planning .4 Project management software 	<ul style="list-style-type: none"> .1 Schedule updates .2 Corrective action .3 Lessons learned

Similarly, we can identify the micro level aspects of Project Cost Management, and pinpoint the control areas under the scope of specific studies. This can be done in Figure 2.0. Project Cost Management includes the processes required to ensure that the project is completed within the approved budget. As evident from figure 2.0, it involves 4 major processes, viz.

1. **Resource Planning** – determining what resources and what quantities of each should be used to perform the project activities,
2. **Cost estimation** – developing an approximation of the costs of the resources needed to complete the project activities,
3. **Cost Budgeting** - Allocating the overall cost estimate to individual work items, and,
4. **Cost control** – controlling changes to the project budget.

Figure 2.0 : Project Cost Management Overview
(Source PMBOK)

Major Processes	2.1.1 Inputs	2.1.2 Tools and Techniques	2.1.3 Outputs
2.1 Resource Planning	.1 Work breakdown structure .2 Historical information .3 Scope statement .4 Resource pool description .5 Organizational policies	.1 Expert judgement .2 Alternatives identification	.1 Resource requirements
2.2 Cost Estimating	.1 Work breakdown structure .2 Resource requirements .3 Resource rate .4 Activity duration estimate .5 Historical information .6 Chart of accounts	.1 Analogous estimating .2 Parametric modeling .3 Bottom-up estimating .4 Computerized tool	.1 Cost estimates .2 Support detail .3 Cost management plan
2.3 Cost Budgeting	.1 Cost estimates .2 Work breakdown structure .3 Project schedule	.1 Cost estimating tools and techniques	.1 Cost baseline
2.4 Cost Control	.1 Cost baseline .2 Performance reports .3 Change requests .4 Cost management plan	.1 Cost change control system .2 Performance measurement .3 Additional planning .4 Computerized tools	.1 Revised cost estimates .2 Budget updates .3 Corrective actions .4 Estimate at completion .5 Lessons learned

Project cost management is primarily concerned with the cost of the resources needed to complete project activities. From an business and economic point of view, it is always the cost figures that are displayed and makes all the difference. It may be noted that any overrun on the Time factor would invariably get reflected on the Cost

factor due to obvious reasons. Thus it is imperative that these two factors are the prime factors to be controlled.

The most important tool in the hand of the Project Manager for this impact control and management is the Project Monitoring System. This has to operate in tandem with professionally laid down project schedules and correct resource allocation for the scheduled jobs. A plan is never static, especially for Infrastructure Projects. As soon as a plan is finalized, something endogenous and/or exogenous mandates a change in it. This instability goes on to upset the subsequent stages, most perceptibly the resource allocations. It thus becomes imperative that the resources, which are finite in numbers and quantity, be reallocated. For a relatively small project, manual reshuffling might be possible, but for Infrastructure Projects, this invariably demands a fast and intelligent computer software, with necessary heuristics built in. (The RCPS problem, to be discussed shortly, is a special case of resource leveling where the heuristic involved is a limitation on the quantity of resource available. The software incorporating this set of heuristics would be as fast as the algorithm it follows, *ceteris paribus*). Based on the outcomes, the project activities may be rescheduled or resequenced. These are invariably inter related, and demands maximum efficiency from the Project Manager. His only ally in this battle against Time-Cost overrun is the Computer, and the Project Management softwares which forms the Project Monitoring System, which earlier was referred to as the control and feedback mechanism. This in essence is the Project Monitoring System, which shall be henceforth referred as ProMonS.

4. Project monitoring systems

The most important component of ProMonS is the Project Management Information System (PMIS). The periodicity of reference and reporting by ProMonS is a crucial factor, and aims at constant checking and monitoring of the present status of the project. Additionally it can be suitably upgraded to predict the future parameters of the project based on historical and other factors. This can be based on any planning technique used in the enterprise. Measurement, evaluation, trouble-shooting and

improvement of performance are the prime objectives of the PMIS. Improvement comes through decision-making, which is based on information.

A Project Management Information System generally consists of three modules –

PMIS/T :: Time Management Information;

PMIS/C :: Cost Management Information; and

PMIS/R :: Resources Management Information.

In addition to these, the Project Monitoring System would consist of a module to handle information on quality and quality standards. Similarly there would be a module to provide information on combined exception report on the total performance.

The system works on certain algorithms, which are based on pertinent assumptions, rules and constraints. To improve on the working and performance of the software, the algorithms needs to be studied, and newer and latest technological innovations needs to be incorporated.

In this paper, attempt is made to highlight one such area where improvement can be made by incorporating the latest tools and techniques.

5. Resource allocation for infrastructure projects

Infrastructure projects depend to a great extent on machinery and equipment. These are usually limited in numbers and scarce to get. In many cases delay occurs in project completion due to the non-availability of the same machinery and equipment for possible parallel jobs. The problem created due to inadequate resource allocation spills over to create a time-overrun problem, which in turn creates problem in the cost management aspect. Procuring additional machinery and equipment could mitigate this problem. But it is a rule rather than exception that procurement of such additional machinery and equipment fails to be justified from the economical point of view. Therefore there is a paradoxical relationship between resources requirement and availability, and their allocation. To study and analyse this situation, this problem has been brought into focus worldwide by the name of Resource-Constrained Scheduling Problem, or RCS Problem.

6. The Resource-Constrained Scheduling Problem; and Resource Allocation under Constraints

The RCS problem, and the related resource allocation problem, has been evolved over a period of time. It has been studied from a number of angles, for varied applications. It initially started with job sequencing in the Shop-Floor, and allocating finite number of machines, operators, etc. Presently this problem is also being studied for application in the area of hardware resource allocation, in the field of computing. And over time, this problem got spilled over into the area of Project Management.

For application into the area of Project Management, the problem was rechristened as the Resource-Constrained Project Scheduling Problem (RCPS). A RCPS consists of a set of tasks, and a set of finite capacity resources. Each task puts some demand on the resources. A partial ordering of these tasks is also given specifying that some tasks must precede others. Generally the goal is to minimize makespan without violating the precedence constraints, or over-utilizing the resources. The prime focus of this problem is the formulation of the sequence of jobs (events and activities) for optimal utilization of the resources (usu. reusable) keeping into account the temporal restrictions. Thus there are resource constraints as well as sequence rules.

Formally, the RCPS problem can be depicted as follows :

Given :

- a set of tasks, T ,
- a set of resources, R ,
- a capacity function, $C : R \rightarrow N$,
- a duration function, $D : T \rightarrow N$,
- a utilization function, $U : T \times R \rightarrow N$,
- a partial order, P on T , and
- a deadline, d .

Find : An assignment of start times $S : T \rightarrow N$, satisfying the following :

- a) Precedence constraints : If t_1 precedes t_2 in the partial order P , then $S(t_1) + D(t_1) \leq S(t_2)$
- b) Resource constraints : For any time x , let $running(x) = \{t | S(t) \leq x < S(t) + D(t)\}$. Then for all times x , and all $r \in R$,

$$\sum_{t \in running(x)} U(t,r) \leq C(r).$$
- c) Deadline : For all tasks $t : S(t) \geq 0$ and $S(t) + D(t) < d$.

One needs to understand this problem and its intricacies, the assumptions, etc. applied till date alongwith the tools and techniques used to solve it. This leads to identification of niches within it where studies can be carried out for improvement

Formulating the total job sequence right from start of the project till its end (completion, or abandonment), with possible parallel paths, and simultaneously allocating resources has been attempted with the help of many methods and algorithms, classically with Network Analysis. With time, contemporary techniques - mostly evolving in the area of Operations Research, were used to take the RCPSP towards its optimal solution. A study of recent works in this area reveals that the latest trend is the application of techniques like Genetic Algorithm (GA), Neural Networks (NN), Machine Learning (ML), etc amongst others

7. Literature review

A number of works have been carried out for obtaining the near optimal solution of the RCS problem for application in Production Scheduling. The application of resource-constrained scheduling and allocation algorithms, and related topics as applicable to the field of Infrastructure Projects is however relatively less. In the following paragraphs, an attempt is made to capture the summary of a few of the works done, especially in the later area in recent times.

One of the pioneering work done specifically in this field was by Balas E. (1971) who laid down a structural approach that involves a generalization of both the disjunctive graph method in job shop scheduling and the order theoretic methods for precedence constrained scheduling.

In 1982, Lawler E.L. et al provided a conceptual summary of works and developments done in the area of deterministic scheduling and scheduling.

Bartush M. and co-workers provides a number of pioneering and referential works in the area of algorithm generation for scheduling problems in construction industry. They have also produced literatures for integrating computers with project scheduling.

Lenstra et al provided a treatment for solving complexities of scheduling project networks with precedence constraints.

During the mid '80s, a number of algorithmic works on project networks scheduling and resource allocation were carried out by Mohring R.H. as well as Radermacher F.J., both jointly as well as independently.

Crawford, J.M. approached the RCPS problem by a combination of limited discrepancy search (LDS) with a novel optimization technique. He arrived at a near final form heuristics for problems of realistic size and character. This work have been run on a series of problems made available by Barry Fox of McDonnell Douglas and Mark Ringer of Honeywell, serving as Benchmarks Secretary in the AAAI Special Interests Group in Manufacturing and in the AIAA Artificial Intelligence Technical Committee respectively.

Patterson J. et al applied the Integer Programming algorithm to project networks for solving the RCS problem.

Presently works in this area is being actively carried out at a number of Universities and Institutions of international repute, especially at Monash University, Australia, Christian-Albrechts-Universitat zu Kiel, Germany, Technische Universitat, Berlin, etc., where application potential of newer tools and techniques (of Artificial Intelligence and OR) have been studied.

To understand the mechanisms of a dynamic system, the use of System Dynamics is one good tool. Love, P.E.D, et al aptly captured this, in a paper in 2001, where they used System Dynamics to understand change and reworks in the construction projects. This powerful tool can be adapted to generate scenarios, and synthesize data sets.

Project Management was provided with a radically new methodology - SYDPIM, which integrates the use of System Dynamics simulation models with the traditional PERT/CPM network models. This was the out come of a massive UK military software project under the stewardship of Rodrigues, A.G. at the University of Strathclyde.

A related but not exactly the same field of study is process resequencing. Here the whole work sequence is altered, without violating rules and precedence, and alternative sequences are developed. Then these new sequences are subjected to study for identifying a (set of) better possible alternative to the original sequence. Attempts to apply GA to this were carried out by Altus S.S. et al and Rogers J L.

The unveiling of a number of techniques used for solving Artificial Intelligence (AI) problems have thrown open the possibility of further easing the RCS and related problems for Project Management.

Holland H.J. in 1975 provided the pioneering work in adapting Natural Systems' processes for Artificial Systems. He laid the foundations for AI techniques.

A number of works were produced in the area of incorporating AI techniques in solving the RCPSP. Hartmann S. and Kolisch R. are the two major contributors in this area. Most of their works were carried out in the late '90s till date.

Alvarez-Valdes R. et al described a heuristic algorithm based on empirical analysis for the RCPSP in the late '80s. Similar work was done by Botor F. in early '90s. Drexel A. et al gave new modelling concepts and studied their impacts on the RCS problem. Neumann and Franck provided some structural questions and priority-rule methods for the RCPSP with time windows. In a few other works, Neumann was assisted by Zimmermann J.

In 1995, Rabelo L. et al attempted a hybrid approach for real time sequencing and scheduling problems. They used NN, GA, Simulation and Machine Learning for their study.

Stork, F. in a paper in 2000 studied the problem, and came out with a Branch and Bound algorithm for a stochastic setup of the RCPSP.

Walker, E.D attempted a DBR (Drum-Buffer-Rope)-based approach to scheduling resource-constrained multiple projects, and compared his proposed heuristic with more conventional methods, esp. PERT based ones.

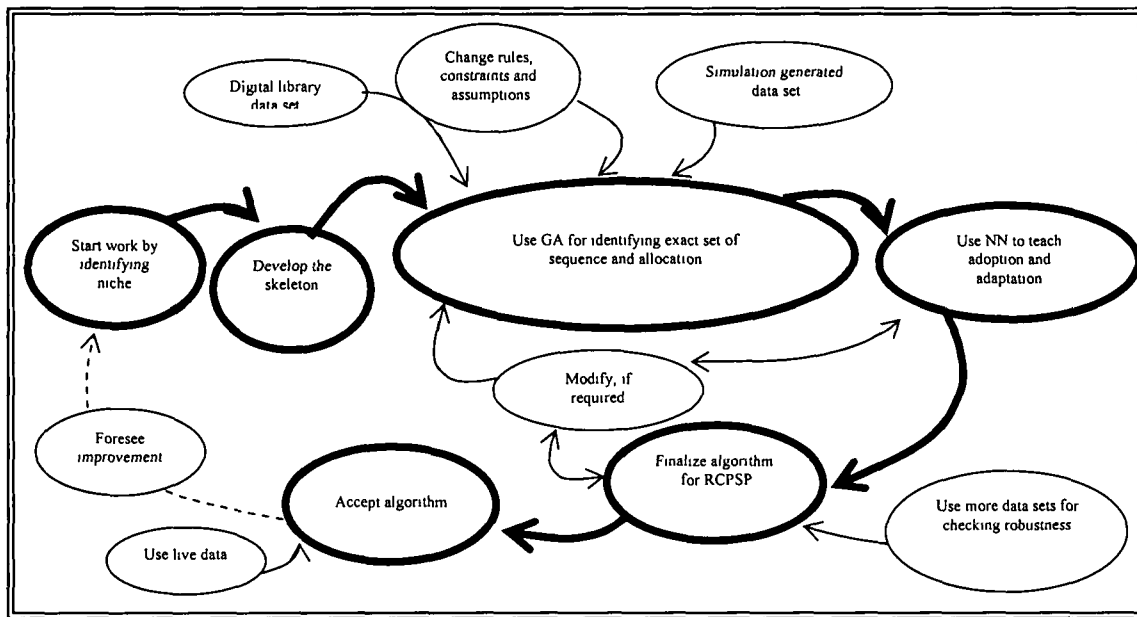
Upon examining these recent works, it was readily apparent that there are many assumptions, most common being that once a “proper plan” is created there is little need for control. Then there are works that have used certain “rules”, but which in practice is not feasible. And a few of them have operated on static project environment. These and other shortcomings point out the possible niche where new works can be carried out.

8. Problem formulation and methodology for solution

By systematically changing the rules, by consciously discarding and/or modifying the assumptions and by modifying constraints on artificial scenarios and data sets, one can depict a project as dynamic. The resultant changes due to these dynamics are next attempted to be taken care of by shifting the resource allocations. This in essence is the methodology to be followed for taking the RCPS problem towards an optimal solution.

Amongst the various research paradigms, the methodology proposed to be utilized shall be one used universally for algorithm design, i.e. Design-Development-Validation. Since this paper is prepared as a background study, the details of the methodology and other aspects is kept beyond its scope. But in essence, the RCPS problem is presently being experimented with application of latest tools and techniques used for solving AI problems (based on adaptation processes of the Biological World) for arriving at an optimal solution. Genetic algorithms are being heavily used in this type of situations where certain patterns exist, and shifts occur within the patterns. Or otherwise, externalities exist or are incorporated that triggers a change in the patterns. Another powerful technique is the use of Neural Networks. This is to be used for imparting intelligence to the algorithm, so that to a great degree the system itself can predict or be able to control changes, and lead to adoption and adaptation. The algorithms so developed are based on synthetic data sets, available from digital libraries of different Universities and Institutions. Alternately, data sets and scenarios are generated by different simulation methods. This pathway is illustrated as Figure 3.0.

Figure 3.0 : Pathway for developing the Algorithm



9. Conclusion

This paper has given only the background of the topic. Because of its inherent nature, Infrastructure Project Management is a field of dynamic changes. The changes, both endogenous as well as exogenous, affect the plans and the related matters. Which in most cases is translated to time overrun, or cost overrun, or both. For assisting the Project Manager, the PM software has to be incorporated with necessary fast and efficient algorithms. For developing and improving such algorithm, it is necessary to understand the dynamics of the changes involved, their impacts as well as the reactions required to control the variation(s) due to these changes.

This paper highlights one micro level problem, understanding of which has a great impact on the macro level. Unless the Project Manager is fully equipped with the latest tools, incorporating the best and most efficient techniques, Time-Cost overruns would continue to plague the industry, and the economy as a whole. Thus an attempt shall be made to attack the Resource-Constrained Project Scheduling problem with a set of new methods. Studies in this direction have started only recently, and this author expects to further the studies. Subsequent write-ups shall highlight the outcomes of works by this author

References *(Partial and Selective List)*

a) Books

1. Chandra, P.; Projects: Planning, Analysis, Selection, Implementation, and Review, 4th Edition, 1995; Tata McGraw-Hill Publishing Company Ltd
2. Duncan, W.R.; A guide to Project Management Body of Knowledge, 1996, Project Management Institute, PA, USA.
3. Goldberg, D E; Genetic Algorithms in Search, Optimization and Machine Learning, 1989, Addison-Wesley.
4. Hoffman, A.; Paradigms of Artificial Intelligence : A Methodological & Computational Analysis, 1998; Springer.
5. Iyer, P.P., Engineering Project Management, with Case Studies, 1996; Wheeler Publishing
6. Joy, P.K.; Total Project Management – The Indian Context, 1993; Macmillan India Ltd

b) Published articles

7. Alvarez-Valdez Olaguybel, R, J. M. Tamarit Goerlich 1989, Heuristic algorithms for resource-constrained project scheduling: A review and an empirical analysis
8. Balas, E., 1971, Project Scheduling with resource constraints.
9. Bartusch M., 1983, An algorithm for generating all maximal independent subsets of a project.
10. Bartusch M., Mohring R.H, Radermacher F.J 1988, A conceptual outline of a DSS for scheduling problems in the building industry.
11. Bartusch M., Mohring R.H, Radermacher F.J. 1988, Scheduling project networks with resource constraints & time windows.
12. Botor, F.F,1994, A new and efficient heuristic for scheduling of projects with resource restrictions and multiple execution modes,.
13. Botor, F.F, 1994, An adaptation of the simulated annealing algorithm for solving resource-constrained project scheduling problem
14. Crawford, J.M., An Approach to Resource-Constrained Project Scheduling
15. Davis, E.W, Heidorn, 1971, An algorithm for optimal project scheduling under multiple resource constraints.
16. Franck B, Neumann K, Schwindt C, 2001, Truncated branch-and-bound, schedule-construction, and schedule-improvement procedures for resource-constrained project scheduling.
17. Franck B, Neumann K, Schwindt C,2001, Project scheduling with calendars.
18. Hartmann, S, 1998, A Competitive Genetic Algorithm for Resource-Constrained Project Scheduling.
19. Holland H J., 1975, Adaptation in Natural and Artificial Systems
20. Kolisch, R., Project Scheduling under Resource Constraints, 1995
21. Lawler E.L, Lenstra J.K, Rinnooy Kan A.H G, 1982, Recent developments in deterministic scheduling and sequencing, a survey, in.
22. Lenstra E.L, Rinnooy J K, Kan A.H.G, 1978, Complexity of scheduling under precedents constraint.
23. Love, P.E.D , et al, 2001, Using systems dynamics to better understand change and rework in construction project management systems.
24. Mohring R.H, Radermacher F J , 1984, Scheduling problemn with resource duration interaction, methods of.
25. Mohring R.H, Schulz, A S , Stork, F., Uetz, M , 2001, On project scheduling with irregular starting time costs.
26. Mohring R H, Schulz, A.S , Stork, F , Uetz, M., 2002, Solving project scheduling problems by minimum cut computation.
27. Neumann K., Schwindt, C , Zimmermann J, 2001, Recent results on resource-constrained project scheduling with time windows: Models, solution methods and applications.
28. Neumann K, Zhan J, 1995, Heuristics for the minimum project-duration problem with minimal and maximal time-lags under fixed resource constraints.

29. Neumann K, Zimmermann J, 1999 Resource levelling for projects with schedule-dependent time windows
30. Neumann K, Zimmermann J, 1999, Methods for resource-constrained project scheduling with regular and nonregular objective functions and schedule-dependent time windows.
31. Patterson J.H, Slowinski R., Weglarz J., 1982, An algorithm for a general class of precedence and RCS problems
32. Patterson J.H, Talbot F B, 1978, An efficient integer programming algorithm with network cuts for solving RCS problem.
33. Radermacher F J., 1986, Schedule induced posets
34. Radermacher F.J., 1986, Scheduling of project networks
35. Robelo, L, Jones, A., Yih, Y., 1995, A Hybrid approach using Neural Networks, Simulation, Genetic Algorithm and Machine Learning for real time sequencing and scheduling problem.
36. Rodrigues, Alexandre J G P, SYDPIM – A System Dynamics-based Project Management Integrated Methodology – Integrating System Dynamics Project Models with PERT/CPM
37. Rogers J.L , McCulley and Bloebum, Integrating a GA into a Knowledge-Based system for ordering Complex Design Processes.
38. Schürmer, A , 1999, Adaptive control schemes applied to project scheduling with partially renewable resources.
39. Stermann, J.D , 1992, System Dynamics modeling for project management
40. Stork, F, 2000, Branch-and-bound algorithms for stochastic resource-constrained project scheduling.
41. Tavares, L.V., Weglarz, J., 1990, Project Management and Scheduling: A Permanent Challenge for OR.
42. Syswerda, G , 1991, Schedule Optimization using Genetic Algorithms
43. Walker, E.D , 2001, Towards a more effective method of scheduling resource-constrained multiple projects.
44. Wall, M.B., 1991, A Genetic Algorithm for resource-constrained scheduling.
45. Williams, T.M , 1999, The need for new paradigms for complex projects

c) Newspaper references

46. The Burmanet News, Nov 1, 1996, More Pain Not Gain In Cost Over-Runs. Ken & Visakha Kawasaki.
47. The Indian Express, May 12, 1998, Project Delays Cost Nation A Whopping 45,000Cr Sunil Jain.
48. The Age, Nov 11, 1999, Cost Over-Runs Hit Big Projects : Ewin Hannan, State Editor
49. The Dawn 9 Feb, 2000, Probe Into Sandak Cost Over-Runs
50. The Hindu, May30, 2000, Public Projects – The Need For Tighter Monitoring: P K.Joy

d) Web Sources

51. Fox, B., Ringer, M., Planning & Scheduling Benchmarks, www.neosoft.com/~benchmrx/
52. Wall, M.B, A Genetic Algorithm for Resource-Constrained Scheduling, www.lancet.mit.edu/pub/mbwall/
53. Various job-shop problems and other OR problems, www.mscmg.ams.ic.ac.uk/info.html

Appendix III

Extension of BATGA

A feature of the scheduling mechanism used for BATGA demonstrates an ability to respond to dynamic interference of input information. This feature is tentatively tested for extension of BATGA. Being in its formative stages the formal report is held back.

The *Sweep-Creep* procedure developed for scheduling in BATGA demonstrates an ability to respond to dynamic interference of input information. This feature is tentatively tested for extension of BATGA. Being in its formative stages the formal report is held back.

The *Sweep-Creep* algorithm is reproduced here,

For scheduling the 'mth' task in the sequence,

Start from beginning of the stub

(1) Check (*Sweep* for) availability of all resources at time slot t

If available for total duration of the activity,

Place the task at that time slot

Else shift(*Creep* up) one time-step ahead and *SweepCreep* from(1) all over,

Until

a) either the task is placed,

b) or end of the stub is reached

If end of stub is reached

Place the task at this end position

Pick up next task of the sequence for scheduling.

The *Sweep-Creep* Algorithm

At point (1), the algorithm checks status of resource availability in time slot, 't'. Every time the algorithm arrives at this position, a check is made. This feature was exploited for testing response of our algorithm to external interference.

After the data-input segment of BATGA reads input information, the same is placed in memory for subsequent processing. BATGA is now allowed to process the project information, as is normally expected of a Genetic Algorithm for project scheduling.

We place a function within the program that randomly interferes with pertinent project information currently in store. Changing input information of one project with that of another project does this. For doing so, based on a random function, the data-input segment reads another project. The resource requirement of the current project is then replaced by those of the new project – but only of those activities yet to be scheduled in the current project. This, we decided, reflects a situation where the resource requirements of some activities change midway of processing a project for an optimum schedule.

We had no comparative information / result to check accuracy and efficiency of BATGA's response. But the effectiveness was amply demonstrated. Because it is a Genetic Algorithm, BATGA proceeds with its processing irrespective of any change in problem data. The resultant is a change in makespan from the optimal, but which is otherwise expected – as a reaction to change in exogenous interference.

However, this feature is yet to be fully experimented with. Being in its formative stage, we refrain from making a formal report, and have therefore appended rather than made it a feature of the main text.

References and Bibliography

We include the digital sources (data-set, results, etc) and literature sources (printed journals, CD-ROM versions, internet downloads, etc) – both which are referred to and/or mentioned in the text as well as those that were consulted with and advice received during all stages of work.

Communication was sent to all possible authors / moderators for usage, and response were received from a majority of them. With due regards, we acknowledge all copyrights, trademarks as well as other reservations on all sources that has been mentioned in this text.

Digital Source References (major)

- A. IlliGAL : The Illionis Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign, USA
- B. KanGAL : The Kanpur Genetic Algorithms Laboratory, Indian Institute of Technology, Kanpur, India
- C. PSPLIB : The Project Scheduling Problems Library, Christian-Albrechts-Universitaet zu Kiel, Germany
- D. EvoNET / EvoWEB : A repository of information and linkages for Evolutionary Computing.

Literature References and Bibliography

1. [Aarts et al(1988)] Aarts, E., P. Laarhoven. Job Shop Scheduling by Simulated Annealing. *Technical Report OS-R8809 Centre for Mathematics and Computer Science(Amsterdam)*. 1988
2. [Adams et al(1988)] Adams, J., Balas E. and Zawack D. The shifting Bottleneck Procedure For Job Shop Scheduling. *Management Science* 34, 391-401. (1988).
3. [Alcaraz and Maroto (2001)] Alcaraz J. and Maroto C. A Robust Genetic Algorithm For Resource Allocation In Project Scheduling. *Annals of Operations Research*, 102:83–109, 2001.
4. [Alcaraz and Maroto (2004)] Alcaraz J, Maroto C. and Ruiz R. Improving The Performance Of Genetic Algorithms For The RCPS Problem. *Proceedings of the Ninth International Workshop on Project Management and Scheduling, Nancy* : 40–43, 2004.
5. [Altus et al(1996)] Altus, S. S.; Kroo, I. M.; and Gage, P. J. A Genetic Algorithm for Scheduling and Decomposition of Multidisciplinary Design Problems, *ASME Paper, Journal of Mechanical Design*, 118:486-489. December 1996
6. [Alvarez-Valdez et al(1989)] Alvarez-Valdez O, Tamarit R., J. M. Goerlich, Heuristic Algorithms For Resource-Constrained Project Scheduling: A Review And An Empirical Analysis. *Advances in Project Scheduling, Slowinski, R., Weglarz, J. (Eds.), Elsevier, Amsterdam*. pp. 113-134. 1989

7. [Aporntewan (2001)] Aporntewan C, Chongstitvatana P. A Hardware Implementation of the Compact Genetic Algorithm. *Proceedings of the 2001 IEEE Congress on Evolutionary Computation, Seoul, Korea*. May 2001
8. [Applegate and Cook (1991)] Applegate, D., and W. Cook. A computational study of the job-shop scheduling instance, *ORSA Journal on Computing* 3, 149-156. 1991
9. [Artigues et al (2003)] Artigues C., Michelon P., Reusser S.. Insertion Techniques For Static And Dynamic Resource–Constrained Project Scheduling. *European Journal of Operational Research*, 149:249–267, 2003.
10. [Artigues et al (2004)] Palpant M., Artigues C., Michelon P. LSSPER: Solving The Resource Constrained Project Scheduling Problem With Large Neighbourhood Search. *Annals of Operations Research*, 131:237–257, 2004
11. [Balas (1967)] Balas, E. Finding a Minimaximal Path in a Disjunctive PERT Network. *Theory of Graphs International Symposium*, 1967
12. [Balas (1971)] Balas, E. Project Scheduling With Resource Constraints, *Application of Mathematical Programming Techniques*, Ed Beale E. M. L. 1971
13. [Bartusch (1983)] Bartusch M., An Algorithm For Generating All Maximal Independent Subsets Of A Proset.
14. [Bartusch et al(1988)] Bartusch M., Mohring RH., Radermacher. FJ. Scheduling Project Networks With Resource Constraints And Time Windows. *Annals of Operations Research*, 16:201–240, 1988.
15. [Berman (2004)] Berman AM. *Data Structures via C++: Objects by Evolution*. OUP. 2004

16. [Bhaskar et al(2004)] Bhaskar T., Pal R. and Pal M.N., Resource Time Ratio Exponent Technique (RETIREXT):An Efficient Non-recursive Heuristic for Project Scheduling under Multiple Resource Constraints. *Ninth International Workshop on Project Management and Scheduling (PMS 2004)*:92-95. 2004
17. [Bidot (2005)] Bidot, J. *A General Framework Integrating Techniques for Scheduling under Uncertainty*, PhD Thesis, Specialty: Industrial Systems Institut National Polytechnique de Toulouse, France. November 2005
18. [Blazewicz et al (1983)] Blazewicz J, Lenstra J. K., Rinnooy Kan A. H. G. Scheduling Subject To Resource Constraints: Classification and Complexity, *Discrete Applied Mathematics*, 5:11–24. 1983
19. [Blickle (1996)1] Blickle T, Thiele L. A Comparison Of Selection Schemes Used In Evolutionary Algorithms. *Evolutionary Computation*, 4(4) December 1996
20. [Blickle (1996)2] Blickle T, Thiele L. A Mathematical Analysis of Tournament Selection. *Proceedings of the 6th International Conference on Genetic Algorithms*. July 1995
21. [Boctor (1990)] Boctor, FF. Some Efficient Multi-Heuristic Procedures For Resource-Constrained Project Scheduling. *European Journal of Operational Research*, , 49(1): 3-13, November. 1990
22. [Boctor (1993)] Boctor, F. F. Heuristics For Scheduling Projects With Resource Restrictions And Several Resource-Duration Modes. *International Journal of Production Research* 31(11): 2547. (1993).
23. [Burjorjee (2008)] Burjorjee K. The Fundamental Problem with the Building Block Hypothesis. *Working Paper. Computer Science Department. Brandeis University Waltham, MA 02454-9110, USA*. October 2008

24. [Can et al(2004)] Can VT, Jacques A, Ferland, Anh NH. Genetic Algorithm for the Resource-Constrained Project Scheduling Problem Using Encoding with Scheduling Mode. December 6, 2004
25. [Coelho and Tavares (2003)] Coelho J. and Tavares L. Comparative Analysis Of Meta-Heuristics For The Resource Constrained Project Scheduling Problem. *Technical report, Department of Civil Engineering, Instituto Superior Tecnico, Portugal, 2003.*
26. [CONDOR Report] Committee on the Next Decade in Operations Research (CONDOR). Operations Research : The Next Decade. *Operations Research* 36:619-637
27. [Crawford (1996)] Crawford, J M. An Approach to Resource Constrained Project Scheduling. *Proceedings of the 1996 Artificial Intelligence and Manufacturing Research Planning Workshop. Albuquerque, NM., The AAAI Press. 1996*
28. [David (2002)] David, EG. Representations for Genetic and Evolutionary Algorithms. *Verlag-Springer. 2002*
29. [Davis (1985)] Davis, L. Job Shop Scheduling with Genetic Algorithms. *Proceedings of International Conference on Genetic Algorithms and their Applications, Pittsburgh, 1985*
30. [Davis and Patterson (1975)] Davis, E. W. and Patterson J. H. A Comparison of Heuristic and Optimum Solutions in Resource-Constrained Project Scheduling. *Management Science* 21(8): 944-955. 1975
31. [Davis and Heidorn (1971)]. Davis, E. W. and Heidorn G. E. An Algorithm for Optimal Project Scheduling under Multiple Resource Constraints. *Management Science* 17(12): B-803-b817. 1971

32. [Davison (1998)] Davison BD. Classification Using an Online Genetic Algorithm. *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI)*, : 1189. July, 1998
33. [De Reyck et al (1998)] De Reyck B, Herroelen, WS. An Optimal Procedure For The Resource Constrained Project Scheduling Problem With Discounted Cash Flows And Generalized Precedence Relations. *Computers & Operations Research*, 25:1-17, 1998
34. [Deb 2001]] Deb. K. *Multi-Objective Optimization Using Evolutionary Algorithms*. John Wiley & Sons, Inc., New York, NY, USA, 2001.
35. [Deb et al (2002)] Deb K, Pratap A, Agarwal S, Meyarivan T. A Fast And Elitist Multiobjective Genetic Algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*. 2002
36. [Debels and Vanhoucke (2005)] Debels D and Vanhoucke M. A Decomposition-Based Heuristic For The Resource-Constrained Project Scheduling Problem. *Technical Report 2005/293, Faculty of Economics and Business Administration, University of Ghent, Ghent, Belgium*, 2005.
37. [Debels and Vanhoucke (2005)] Debels D and Vanhoucke M. A Bi-Population Based Genetic Algorithm For The Resource-Constrained Project Scheduling Problem. *VLG Working Paper 2005/8, VLG Management School, University of Ghent, Ghent, Belgium*, 2005.
38. [Debels and Vanhoucke (2006)] Debels D and Vanhoucke M. Meta-Heuristic Resource-Constrained Project Scheduling : Solution Space Restrictions and Neighbourhood Extensions. *VLG Working Paper 2006/18, VLG Management School, University of Ghent, Ghent, Belgium*, 2006.
39. [Debels et al(2006)] Debels D., De Reyck, B., Leus, R. and Vanhoucke M. A Hybrid Scatter Search/Electromagnetism Metaheuristic for Project

Scheduling. *European Journal of Operational Research*, 169(2), pp 638-653, 2006

40. [Deblaere et al(2007)] Deblaere F, Demeulemeester E, Herroelen W, Van de Vonder S. Robust Resource Allocation Decisions in Resource-Constrained Projects. *Decision Sciences*, 38(1):5-37 February 2007
41. [DellAmico and Trobian (1993)] DellAmico M; Trobian M. Applying Tabu Search to the Job Shop Scheduling Problem. *Annals of Operations Research*, 41:231-252. 1993
42. [Demeulemeester (2002)] Demeulemeester EL, Herroelen W. *Project Scheduling: A Research Handbook*. 447. 2002
43. [Demeulemeester and Herroelen (1992)] Demeulemeester E. and Herroelen W., A Branch-And-Bound Procedure For The Multiple Resource-Constrained Project Scheduling Problem, *Management Science*,. 38, .: 1083–1818, 1992.
44. [Evaristo and van Fenema (1999)]Evaristo R and van Fenema PC. A Typology of Project Management: emergence and evolution of new forms. *International Journal of Project Management*. 17(5):275-281. 1999
45. [Falkenauer (1998)] Falkenauer E, *Genetic Algorithms and Grouping Problems*, ISBN 978-0-471-97150-4. England: John Wiley & Sons Ltd. 1998
46. [Fleszar et al(2004)]Fleszar K. and Hindi K. Solving The Resource-Constrained Project Scheduling Problem By A Variable Neighbourhood Search. *European Journal of Operational Research*, 155:402–413, 2004.

47. [Fleurent and Ferland (1994)] Fleurent, C., and Ferland, J., Genetic Hybrids For The Quadratic Assignment Problem, DIMACS. *Series in Mathematical Theoretical Computer Science*, 16, 190–206.
48. [Fogel (1966)] Fogel IJ, Owens AJ, Walsh MJ. *Artificial Intelligence Through Simulated Evolution*. John Wiley, New York. 1966
49. [Fogel (2000)] Fogel DB, Michalewicz Z. *How To Solve It: Modern Heuristics* Springer-Verlag New York, Inc. 2000
50. [Fogel (2006)] Fogel DB. *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*. ISBN 0471749206, 9780471749202. Wiley-IEEE, 2006
51. [Fogel and Perlman (2006)] Fogel BL and Perlman S. Novel Mutations In The Senataxin DNA/RNA Helicase Domain In Ataxia With Oculomotor Apraxia 2. *Neurology*, 67(11):2083-2084, 2006
52. [Franck et al (2001)] Franck B, et al. Truncated Branch-And-Bound, Schedule-Construction, And Schedule-Improvement Procedures For Resource-Constrained Project Scheduling. 2001
53. [Garey et al., 1978] Garey M.R., Graham R.L., and Johnson D.S.. Performance Guarantees For Scheduling Algorithms. *Operations Research*, 26(1):3–21, January–February 1978.
54. [Glover (1989)] Glover, F. Tabu Search - Part I, *ORSA Journal on Computing*. 1(3) :190-206. 1989
55. [Glover (1990)] Glover, F. Tabu Search - Part II, *ORSA Journal on Computing*, Vol 2, No. 1, 4-32(1990)

56. [Glover (2007)] Glover, F. Tabu Search – Uncharted Domains, *Annals of Operations Research*, Vol. 149, No. 1, pp. 89-98. 2007
57. [Glover and Greenberg (1989)] Glover, F and Greenberg H.J. New Approaches for Heuristic Search: A Bilateral Linkage with Artificial Intelligence, *European Journal of Operational Research*. 39(2).pp.119-130. 1989
58. [Glover and Laguna (1997)] Glover F and Laguna M. *Tabu Search*, Kluwer Academic Publishers. ISBN 0 792 39965 X. 1997
59. [Godley et al (2007)] Godley PM., Cairns DE. and Cowie J. Directed Intervention Crossover Applied To Bio-Control Scheduling. *IEEE: Proceedings of IEEE Congress On Evolutionary Computation, CEC 2007*. 2007
60. [Goldberg, 1989)] Goldberg, D. E. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading, MA: Addison-Wesley). 1989
61. [Goncalves and Mendes (2003)] Goncalves J. and Mendes J. A Random Key Based Genetic Algorithm For The Resource – Constrained Project Scheduling Problem. (*Technical report, AT&T Labs, TD-6DUK2C, 2005*)
Technical Report, Departamento de Engenharia, Universidade do Porto, 2003
62. [Goncalves and Mendes (2005)] Refer to previous entry
63. [Goodpasture (2004)] Goodpasture, JC. *Quantitative Methods in Project Management*. ISBN 1-932159-15-0. J Ro ss
Publishing Inc. FL. 2004
64. [Gruninger (1996)] Gruninger T. Multimodal Optimization using Genetic Algorithms. *Technical Paper, Stuttgart University*. 1996

65. [Hartmann (1998)] Hartmann, S. A Competitive Genetic Algorithm for Resource-Constrained Project Scheduling. *Naval Research Logistics*. 45 : 733-750. 1998
66. [Hartmann (2002)] Hartmann S. A Self Adapting Genetic Algorithm For Project Scheduling Under Resource Constraints. *Naval Research Logistics*. 49 : 433-448. 2002
67. [Hartmann and Kolisch (2000)] Hartmann, S, Kolisch, R. Experimental Evaluation Of State-Of-The-Art Heuristics For The Resource Constrained Project Scheduling Problem. *European Journal Of Operational Research* 127 :394-407. 2000
68. [Herroelen (1972)] Herroelen WS. Resource-Constrained Project Scheduling - The State of the Art. *Operational Research Quarterly*, 23(3):261-275. Sep., 1972
69. [Hildum (1994)] Hildum DW. *Flexibility In A Knowledge-Based System For Solving Dynamic Resource-Constrained Scheduling Problems* PhD Thesis, Department of Computer Science, University of Massachusetts Amherst. September 1994
70. [Hill (1999)] Hill RR. A Monte Carlo Study Of Genetic Algorithm Initial Population Generation Methods. *Proceedings Winter Simulation Conference*. Farrington, Nembhard, Sturrock, and Evans, eds. 1999
71. [Hindelang and Muth (1979)] Hindelang, TJ, Muth, JF. A Dynamic Programming Algorithm For Decision CPM Networks. *Operations Research*, 27(2):225-241, Mar-Apr 1979
72. [Hindi (2002)] Hindi K. S., Yang H., Fleszar K.. An Evolutionary Algorithm For Resource Constrained Project Scheduling. *IEEE Transactions on Evolutionary Computation*, 6:512–518, 2002.

73. [Hoffman (1998)] Hoffman, A.; 1998, *Paradigms Of Artificial Intelligence : A Methodological & Computational Analysis*,; Springer.
74. [Holland (1975)] Holland JH, *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor. 1975
75. [Iyer (1996)] Iyer, P.P. *Engineering Project Management, With Case Studies*, Wheeler. 1996
76. [Jean (1996)] Jean, YP. Genetic Algorithms for the Travelling Salesman Problem, *Annals of Operations Research*, 63: 339 – 370. 1996
77. [Jones (1995)] Jones T., Crossover, Macromutation, And Population-Based Search, *Proceedings of the Sixth International Conference on Genetic Algorithms*, 1995
78. [Jones(1979)] Jones WT and Chiaraviglio, L Is Science An Adaptive System? *Behavioral Science* 24(5), 325-333.1979
79. [Joy (1993)] Joy, P.K.; *Total Project Management – The Indian Context*, Macmillan. 1993
80. [Józefowska and Zimniak (2004)] Józefowska J, Zimniak A. A Multiple Criteria Genetic Algorithm Operating on a Reduced Search Space. *Proceedings of 10th IEEE Conference on Methods and Models in Automation and Robotics*, Domek S., Kaszyński R. (Eds.): 1379-1384. 2004
81. [Kemmo, et al(2007)] Kemmo, S.T, Gourgand, M, Quilliot, A., Solving Resource Constrained Project Scheduling Problem with Particle Swarm Optimization, *Proceedings of the 3rd Multidisciplinary International Conference on Scheduling : Theory and Applications, MISTA'07*, pp 251-258. August 2007

82. [Khamooshi (1999)] Khamooshi H. Dynamic Priority-Dynamic Programming Scheduling Method (DP)2SM: A Dynamic Approach To Resource Constraint Project Scheduling *International Journal of Project Management*, 17:383-391(9). December 1999
83. [Kim (2007)] Kim JL. Permutation Based Elitist Genetic Algorithm Using Serial Scheme For Large-Sized Resource-Constrained Project Scheduling. *Proceedings of the 2007 Winter Simulation Conference*, 2007
84. [Kirkpatrick et al (1983)] Kirkpatrick, S., C. D. Gelatt, et al. Optimization by Simulated Annealing. *Management Science* 220:671-680. 1983
85. [Klein (1999)] Klein, R. *Scheduling Of Resource-Constrained Projects*. Kluwer Academic Publishers, Dordrecht. 1999
86. [Klein (2000)] Klein, R.. Bidirectional planning: improving priority rule based heuristics for scheduling resource–constrained projects. *European Journal of Operational Research*. 127:619–638, 2000.
87. [Kolisch (1995)] Kolisch, R. *Project Scheduling under Resource Constraints*. Heidelberg, Physica-Verlag. 1995
88. [Kolisch (1996)/1] Kolisch R. Efficient Priority Rules for the Resource-Constrained Project Scheduling Problem. *Journal of Operations Management* 14(3):179-192. 1996
89. [Kolisch (1996)/2] Kolisch, R. Serial and Parallel Resource-Constrained Project Scheduling Methods Revisited: Theory and Computation. *European Journal of Operational Research* 90:320-333. 1996
90. [Kolisch and Hartmann (1998)] Kolisch R., Hartmann S. Heuristic Algorithms For Solving The Resource Constrained Project Scheduling

Problem: Classification And Computational Analysis. *Handbook on Recent Advances in Project Scheduling*, Kluwer, Boston, 1998.

91. [Kolisch and Hartmann (2000)] Kolisch R and Hartmann S. Heuristic Algorithms For The Resource Constrained Project Scheduling Problem: Classification And Computational Analysis. *Project Scheduling: Recent Models, Algorithms and Applications*, ed. Jan Weglarz, pp 147–178, Boston, Kluwer Academic Publishers. 2000
92. [Kolisch and Hartmann (2006)] Kolisch R. and Hartmann S. Experimental Investigation Of Heuristics For Resource-Constrained Project Scheduling: An Update. *European Journal of Operational Research* 174(1): 23-37. 2006
93. [Kolisch and Padman (1997)] Kolisch, R. and Padman, R. An Integrated Survey Of Deterministic Project Scheduling. *Technical Report 463, Manuscripte aus den Instituten fur Betriebswirtschaftslehre der Universitat Kiel*. 1997
94. [Kolisch and Sprecher (1996)] Kolisch R and Sprecher A. PSPLIB – A Project Scheduling Library. *European Journal of Operations Research*. : 205 – 216. 1996
95. [Kolisch et al (1992)] Kolisch R., Sprecher A., and Drexl A. Characterization and Generation of a General Class of Resource-Constrained Project Scheduling Problems. *Technical Paper Universitat zu Kiel*. 1992
96. [Kurtulus and Davis (1982)] Kurtulus I. and Davis E.W. Multi-Project Scheduling: Categorization of Heuristic Rules Performance, *Management Science* 28(2):161-172. 1982

97. [Ladd (1996)] Ladd SR. *Genetic Algorithms in C++*. ISBN 1558514597. M&T Books, 1996
98. [Lawler et al(1982)] Lawler EL, Lenstra JK, and Rinnooy Kan AHG Recent Developments in Deterministic Sequencing and Scheduling: A Survey. *Deterministic and Stochastic Scheduling*. : 35-74, 1982
99. [Lawrence and Morton, 1993] Lawrence SR and Morton TE. Resource Constrained Multi-Project Scheduling With Tardy Costs: Comparing Myopic, Bottleneck, And Resource Pricing Heuristics. *European Journal of Operational Research*, 64(2):168–187, January 1993.
100. [Lenstra et al(1978)] Lenstra J., Rinnooy K., Complexity of Scheduling under Precedence Constraints. *Operations Research* 26(1):22-35. 1978
101. [Lenstra, et al(1978)] Lenstra E.L, Rinnooy J.K, Kan A.H.G, Complexity Of Scheduling Under Precedents Constraint. 1978
102. [Love (2001)] Love, P.E.D. The Influence Of Project Type And Procurement Method On Rework Costs In Construction Projects, *ASCE Journal of Construction Engineering and Management*. 2001.
103. [Love et al(1999)] Love, PED, Mandal, P, and Li, H. Determining The Causal Structure Of Rework Influences In Construction. *Construction Management and Economics*. 17 :505-517. (1999)
104. [Love and Holt(2000)] Love, P.E.D.and Holt, G.D. (2000), Construction Business Performance Measurement: The SPM Alternative, *Business Process Management Journal*, 6(5):.408-16. 2000
105. [Love et al(2000)] Love, P.E.D., Mandal, P., Smith, J., Li, H. Modelling The Dynamics Of Design Error Induced Rework In Construction, *Construction Management and Economics*, 18(5):575-86. 2000

106. [Love et al(2002)] Love, PED, Holt, GD, Shen, LY, Li, H, Irani, Z. Using Systems Dynamics To Better Understand Change And Rework In Construction Project Management Systems. *International Journal of Project Management*. 20:425-436.2002
107. [Mallikarjun and Chugan (2004)] *Managing Trade, Technology and Environment* ISBN:81-7446-363-x, Ed. Mallikarjun M and Chugan PK. Excel Books, 2004
108. [Mellentien and Trautmann(2001)] Mellentien, C, Trautmann, N. Resource Allocation With Project Management Software. *OR Spektrum* 23:383-394, 2001
109. [Merkle et al(2002)]Merkle D, Middendorf M, Schmeck H. Ant Colony Optimization For Resource-Constrained Project Scheduling. *IEEE Transactions on Evolutionary Computation* 6(4): 333-346. 2002
110. [Michalewicz (1996)] Michalewicz Z. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer, ISBN 3540606769, 9783540606765. 1996
111. [Mohsenin and Ali (2008)] Mohsenin Y and Ali HH. A New Genetic Algorithm for Resource Constrained Project Scheduling. *Proceedings of Artificial Intelligence and Applications*. 2008
112. [Mohsenin et al(2008)] Mohsenin Y., Ali H.H. A New Genetic Algorithm for Resource Constrained Project Scheduling. *Proceeding of Artificial Intelligence and Applications*. 2008
113. [Mühlbauer, et. al(2007)] Mühlbauer, R, Klambauer, T, Ruprechtsberger, W, Schaitl, J, Comparison of Project Management-Software, *Seminar Projektorganisation, Johannes Kepler Universitat Linz, Institut fur Bioinformatik, Arbeitsgruppe Informations Systeme (IFS), Linz, 5. December, 2007*

114. [Muth and Thompson (1963)] Muth, JF, Thompson, GL. *Industrial Scheduling*. Prentice Hall, 1963.
115. [Neumann (1990)] Neumann, K. *Stochastic Project Networks - Temporal Analysis, Scheduling, and Cost Minimization.*, Springer-Verlag. Berlin. 1990.
116. [Neumann et al(1995)] Neumann K, Zhan J. Heuristics For The Minimum Project-Duration Problem With Minimal And Maximal Time-Lags Under Fixed Resource Constraints. 1995
117. [Neumann et al(1999)/1] Neumann K, Zimmermann J, Resource Levelling For Projects With Schedule-Dependent Time Windows. 1999
118. [Neumann et al(1999)/2] Neumann K, Zimmermann J, Methods For Resource-Constrained Project Scheduling With Regular And Nonregular Objective Functions And Schedule-Dependent Time Windows. 1999
119. [Neumann et al(2001)] Neumann K., Schwindt, C., Zimmermann J, Recent Results On Resource-Constrained Project Scheduling With Time Windows: Models, Solution Methods And Applications 2001
120. [Nonobe and Ibaraki (2002)] Nonobe K. and Ibaraki T. Formulation And Tabu Search Algorithm For The Resource Constrained Project Scheduling Problem. *Essays and Surveys in Metaheuristics ed Ribeiro and Hansen* :557–588. 2002.
121. [Nowicki and Smutnicki (1996)] Nowicki BA, Smutnicki. A Fast Tabu Search Algorithm for the Job Shop Problem. *Management Science*. 42(6):797-813. 1996
122. [Ogino (2002)] Ogino D, Mori S, Nose M and Sawada H. Pedigree Analysis Programme GTree 1.0. *Genome Informatics* 13:246-247. 2002

123. [Pan and Yeh, 2003] Pan H and Yeh CH, Fuzzy Project Scheduling, *12th IEEE International Conference on Fuzzy Systems, 2003. FUZZ '03.* 1:755-760. May 2003
124. [Panwalkar and Iskander, (1977)] Panwalkar S.S. and Iskander W. A Survey Of Scheduling Rules. *Operations Research*, 25(1):45–61, January 1977.
125. [Patterson (1982)] Patterson J.H, Slowinski R., Weglarz J., An algorithm for a general class of precedence and RCS problems. 1982
126. [Patterson (1984)] Patterson, JH. A Comparison Of Exact Approaches For Solving The Multiple Constrained Resource Project Scheduling Problem. *Management Science* 30(7):854. 1984
127. [Patterson (1990)] Patterson JH, Talbot BF, Slowinski R, and Weglarz J, Computational Experience With A Backtracking Algorithm For Solving A General Class Of Precedence And Resource-Constrained Scheduling Problems. *European Journal of Operational Research.* vol. 49, pp. 68–79, 1990.
128. [Patterson et al (1978)] Patterson J.H, Talbot F.B, An Efficient Integer Programming Algorithm With Network Cuts For Solving RCS Problem. 1978
129. [Peteghem and Vanhoucke (2008)] Peteghem V. and Vanhoucke M. Hybridized Scatter Search Heuristic For The Multi-Mode RCPSP. *11th International Workshop on Project Management and Scheduling (PMS), Istanbul, Turkey.* April 2008
130. [Pinson et al(1994)] Pinson, E., Prins, C., and Rullier, F. Using Tabu Search For Solving The Resource-Constrained Project Scheduling Problem. *Proceedings of the 4 International Workshop On Project Management And Scheduling*, Leuven :. 102-106. 1994

131. [PMBOK (1996)] Duncan, W.R. et al ed. *A Guide to the Project Management Body of Knowledge (PMBOK® Guide)* 1996 Edition, ANSI.PMI 1996
132. [PMBOK (2000)] Belev G. et al ed. *A Guide to the Project Management Body of Knowledge (PMBOK® Guide)* Second Edition, ANSI.PMI 2000
133. [PMBOK (2004)] Bolles D. et al ed. *A Guide to the Project Management Body of Knowledge (PMBOK® Guide)* Third Edition ANSI.PMI 2004
134. [Pupong et al (2008)] Pupong P, Khadwilard A, and Klakankhai A. Multi-matrix Real-coded Genetic Algorithm for Minimising Total Costs in Logistics Chain Network. *International Journal of Intelligent Systems and Technologies* Winter 2008
135. [Ranjbar et al(2007)] Ranjbar, M and Kianfar, F. A Hybrid Scatter Search for the RCPSP (Accepted for publication in *Transaction E: Industrial Engineering* Vol 16, No 1) 2007
136. [Robelo et al(1995)] Robelo L., Jones, A., Yih, Y. A Hybrid Approach Using Neural Networks, Simulation, Genetic Algorithm And Machine Learning For Real Time Sequencing And Scheduling Problem. 1995
137. [Rodrigues (1999)] Rodrigues AJGP. SYDPIM Integration of SD and PERT/CPM Tools: Assessing Fagan Analysis in a Large-Scale Software Project. *Management Science*. 1999
138. [Rogers (1996)] Rogers J. L., McCulley C. M. and Bloebaum C. L., Integrating a Genetic Algorithm Into a Knowledge-Based System for Ordering Complex Design Processes, *NASA TM 110247*, April 1996
139. [Sastry and Goldberg (2001)] Sastry K. and Goldberg D.E. Modeling Tournament Selection With Replacement Using Apparent Added Noise.

Proceedings of the Genetic and Evolutionary Computation Conference 11 (2001) :
129 – 134. 2001

140. [Sastry and Goldberg (2007)] Sastry, K. and Goldberg, D.E. Lets Get Ready To Rumble Redux: Crossover Versus Mutation Head To Head On Exponentially Scaled Problems. *Proceedings of Proceedings of the Genetic and Evolutionary Computation Conference'07, London, England.* July 2007
141. [Schirmer (1999)] Schirmer, A., Adaptive Control Schemes Applied To Project Scheduling With Partially Renewable Resources. 1999
142. [Shtub, et al. (1994)] Shtub A, Bard JF, Globerson, S. *Project Management: Engineering, Technology and Implementation.* ISBN 0-13-556458-1. Prentice Hall, Englewood Cliffs, NJ 07632. 1994
143. [Shukla et al(2008)] Shukla, S, Şon, Y, Tiwari, Fuzzy-based adaptive sample-sort simulated annealing for resource-constrained project scheduling. *The International Journal of Advanced Manufacturing Technology*, 36: 982-995(14). April 2008
144. [Slowinski et al(1989)] Slowinski R., Weglarz J., Eds. *Advances in Project Scheduling.* Amsterdam, Elsevier. 1989
145. [Smith (2004)] Smith T. *Window Based Project Scheduling Algorithms* PhD Thesis, Department of Computer and Information Science, University of Oregon, June 2004
146. [Sprecher et al (1995)] Sprecher A., Drexl A., Characterization and generation of a general class of resource-constrained project scheduling problems, *Management Science.*, 41:1693–1703. 1995.
147. [Sterman (1992)] Sterman, J.D., System Dynamics Modeling For Project Management. 1992

148. [Stinson et al., 1978] Stinson JP, Edward W. Davis, and Basheer M. Khumawala BM. Multiple resource-constrained scheduling using branch and bound. *AIIE Transactions*, 10(3):252–259, September 1978.
149. [Stork (2000)] Stork, F, Branch-And-Bound Algorithms For Stochastic Resource-Constrained Project Scheduling. 2000
150. [Syswerda (1989)] Syswerda, G. Uniform Crossover In Genetic Algorithms *Proceedings of the Third International Conference on Genetic Algorithms*, Morgan Kaufmann. 1989
151. [Syswerda (1990)] Syswerda, G. The Application of Genetic Algorithms to Resource Scheduling. *Proceedings of Fourth International Conference on Genetic Algorithms* : 502-508. 1990
152. [Syswerda (1991)] Syswerda, G. Schedule Optimization Using Genetic Algorithms. *Handbook of Genetic Algorithms*, New York, Van Nostrand Reinhold. 1991
153. [Thomas and Salhi (1998)] Thomas PR, Salhi S A Tabu Search Approach for the Resource Constrained Project Scheduling Problem. *Journal of Heuristics*, 4(2):.123-139. 1998
154. [Tinnirello (1999)] Ed Tinnirello. *Project Management Handbook, Best Practices Series*. P, ISBN 0 8493 9998 X. Auerbach. 1999
155. [Tormos and Lova (2001)] Tormos P. and Lova A. A Competitive Heuristic Solution Technique For Resource Constrained Project Scheduling. *Annals of Operations Research*, 102:65–81, 2001.
156. [Tormos and Lova (2003)] Tormos P. and Lova A. An Efficient Multi-Pass Heuristic For Project Scheduling With Constrained Resources. *International Journal of Production Research*, 41(5):1071– 1086, 2003.

157. [Valls et al(2003)] Valls V., Ballestin F., and Quintanilla M. S. A Hybrid Genetic Algorithm For The RCPSP. *Technical report, Department of Statistics and Operations Research, University of Valencia*, 2003. Published in *European Journal of Operational Research*, 185(2), pp 495-508, 2008
158. [Valls et al(2008)] *Refer to previous entry*
159. [Valls et al(2004)] Valls V., Ballestin F., and. Quintanilla MS. A Population-Based Approach To The Resource-Constrained Project Scheduling Problem. *Annals of Operations Research*, 131:305–324, 2004.
160. [Vaziri et al(2005)] Vaziri K, Nozick LK, Turnquist MA. Resource Allocation And Planning For Program Management. *Proceedings of the 37th conference on Winter Simulation Orlando, Florida* :2127 – 2135. 2005
161. [Vose and Wright (1998)] Vose MD and Wright AH. The Simple Genetic Algorithm and the Walsh Transform: Part I, Theory. *Evolutionary Computation*, 6(3):253 (1998).
162. [Walker (2001)] Walker, E.D., Towards A More Effective Method Of Scheduling Resource-Constrained Multiple Projects. 2001
163. [Wall (1996)] Wall, M.B. *A Genetic Algorithm For Resource-Constrained Scheduling*. PhD Thesis, Department of Mechanical Engineering, Massachusetts Institute of Technology, USA. June 1996
164. [Watson and Jansen (2007)] Watson RA and Jansen T. A Building-Block Royal Road Where Crossover is Provably Essential. *Proceedings of the Genetic and Evolutionary Computation Conference*: 1452-1459. 2007
165. [Williams (1999)] Williams, T.M., The Need For New Paradigms For Complex Projects. *International Journal of Project Management*. 17(5):269-273. 1999

166. [Wolpert and Macready (1997)] Wolpert, D.H., Macready, W.G. No Free Lunch Theorems for Optimization. *IEEE Transactions on Evolutionary Computation* 1, 67. 1997
167. [Wright, et al(2003)] Wright AH, Vose MD, Rowe JE. Implicit Parallelism. *Proceedings of Genetic and Evolutionary Computation Conference*, 2003.
168. [Xu et al)2007)] Xu N, Bernstein O, Nozick L, Jones D. Stochastic Rollout And Justification To Solve The Resource-Constrained Project Scheduling Problem. *Proceedings of the 2007 IEEE Winter Simulation Conference* Ed Henderson, Biller, Hsieh, Shortle, Tew, and. Barton, pp 1820 1827. 2007
169. [Yassine et al(2007)] Yassine AA, Meier C, Browning. Multi-Project Scheduling using Competent Genetic Algorithms. *Technical Report University of Illinois Department of Industrial & Enterprise Systems Engineering (IESE)* Feb. 2007
170. [Zwikael, et al (2004)] Zwikael, O, Globerson S. Evaluating the quality of Project Planning: A Model and Field Results. *International Journal of Production Research*, 42(8):1545-46. 2004
171. [Zwikael, et al (2006)] Zwikael, O, Cohen, Y, Sadeh, A. Non-Delay Scheduling as a Managerial Approach for Managing Projects. *International Journal of Project Management* 24:330-336. 2006