

Summer 2003

# Optimization-Based Architecture for Managing Complex Integrated Product Development Projects

Hisham Mohamed El-Sayed AbdelSalam  
*Old Dominion University*

Follow this and additional works at: [https://digitalcommons.odu.edu/mae\\_etds](https://digitalcommons.odu.edu/mae_etds)

 Part of the [Industrial Engineering Commons](#), [Mechanical Engineering Commons](#), and the [Operational Research Commons](#)

---

## Recommended Citation

AbdelSalam, Hisham M.. "Optimization-Based Architecture for Managing Complex Integrated Product Development Projects" (2003). Doctor of Philosophy (PhD), dissertation, Mechanical Engineering, Old Dominion University, DOI: 10.25777/fxr5-j946 [https://digitalcommons.odu.edu/mae\\_etds/104](https://digitalcommons.odu.edu/mae_etds/104)

This Dissertation is brought to you for free and open access by the Mechanical & Aerospace Engineering at ODU Digital Commons. It has been accepted for inclusion in Mechanical & Aerospace Engineering Theses & Dissertations by an authorized administrator of ODU Digital Commons. For more information, please contact [digitalcommons@odu.edu](mailto:digitalcommons@odu.edu).

**OPTIMIZATION-BASED ARCHITECTURE FOR MANAGING  
COMPLEX INTEGRATED PRODUCT DEVELOPMENT PROJECTS**

by

Hisham Mohamed El-Sayed AbdelSalam  
B.S. July 1996, Cairo University, Cairo, Egypt  
M.S. May 2000, Old Dominion University, Norfolk, USA

A Dissertation Submitted to the Faculty of  
Old Dominion University in Partial Fulfillment of the  
Requirement for the Degree of

DOCTOR OF PHILOSOPHY

MECHANICAL ENGINEERING

OLD DOMINION UNIVERSITY

August 2003

Approved by

Han P. Bao (Director)

Sebastian Bawab (Member)

Keith Williamson (Member)

Charles Keating (Member)

## ABSTRACT

### OPTIMIZATION-BASED ARCHITECTURE FOR MANAGING COMPLEX INTEGRATED PRODUCT DEVELOPMENT PROJECTS

Hisham M. E. AbdelSalam  
Old Dominion University, 2003  
Director: Dr. Han. P. Bao

By the mid-1990's, the importance of early introduction of new products to both market share and profitability became fully understood. Thus, reducing product time-to-market became an essential requirement for continuous competition. Integrated Product Development (IPD) is a holistic approach that helps to overcome problems that arise in a complex product development project. IPD emphasis is to provide a framework for an effective planning and managing of engineering projects. Coupled with the fact that about 70% of the life cycle cost of a product is committed at early design phases, the motivation for developing and implementing more effective methodologies for managing the design process of IPD projects became very strong.

The main objective of this dissertation is to develop an optimization-based architecture that helps guiding the project manager efforts for managing the design process of complex integrated product development projects. The proposed architecture consists of three major phases: system decomposition, process re-engineering, and project scheduling and time-cost trade-off analysis. The presented research contributes to five areas of research:

1. Improving system performance through efficient re-engineering of its structure. The Dependency Structure Matrix (DSM) provides an effective tool for system structure understanding. An optimization algorithm called Simulated Annealing (SA) was implemented to find an optimal activity sequence of the DSM representing a design project.

2. A simulation-based optimization framework that integrates simulated annealing with a commercial risk analysis software called Crystal Ball<sup>TM</sup> was developed to optimally re-sequence the DSM activities given stochastic activity data.
3. Since SA was originally developed to handle deterministic objective functions, a modified SA algorithm able to handle stochastic objective functions was presented.
4. A methodology for the conversion of the optimally sequenced DSM into an equivalent DSM, and then into a project schedule was proposed.
5. Finally, a new hybrid time-cost trade-off model based on the trade-off of resources for project networks was presented.

These areas of research were further implemented through a developed excel add-in called “**optDSM**”. The tool was developed by the author using Visual Basic for Application (VBA) programming language.

And your Lord has ordained that you do not worship anyone except Him, and treat your parents with kindness; if either of them or both reach old age in your presence, do not say "Uff"\* to them and do not rebuff them, and speak to them with the utmost respect. And lower your wing humbly for them, with mercy, and pray, "My Lord! Have mercy on them both, the way they nursed me when I was young." [Holy Quran 17:23-24] (\* *Any expression of disgust.*)

This dissertation is dedicated to my mother, Prof. Ragaa Osman Osman and to my father,  
Prof. Mohamed El-Sayed AbdelSalam.

## ACKNOWLEDGMENTS

The author owes many thanks to a number of people who contributed in a variety of ways to the successful completion of this dissertation.

My first debt of gratitude must go to my research advisor Dr. Han P. Bao. Over the past four years, Dr. Bao provided the vision, encouragement and advise necessary for me to proceed through both Masters and Doctoral programs in Mechanical Engineering.

Special thanks go to my committee members: Dr. Sebastian Bawab, Dr. Keith Williamson, and Dr. Charles Keating for their support, guidance and helpful suggestions.

I am also grateful to Dr. Sushil Chaturvedi chairman of the Department of Mechanical Engineering for the financial assistance he made available to support the completion of this dissertation.

This dissertation owes a great deal to my older brothers Dr. Omar and Dr. Tarek who have encouraged and helped me for as long as I can remember.

The last thank you is saved for my dear wife Rasha. She sacrificed a career back home, moved with me to the US one week after we got married, and stood by my side throughout the completion of my research.

## TABLE OF CONTENTS

	Page
<b>LIST OF TABLES</b> .....	xi
<b>LIST OF FIGURES</b> .....	xiii
Chapter	
<b>I. INTRODUCTION</b> .....	1
1. INTRODUCTION TO THE DOMAIN: RESEARCH MOTIVATION .....	1
2. INTRODUCTION TO THE CONTEXT: PROBLEM FRAMING .....	5
3. INTRODUCTION TO THE RESEARCH: RESEARCH OBJECTIVES .....	8
4. INTRODUCTION TO THE TECHNOLOGY: ANALYTICAL STRATEGY AND SOLUTION POTENTIAL .....	10
5. INTRODUCTION TO THE DISSERTATION: READER'S GUIDE .....	12
<b>II. THE DEPENDENCY STRUCTURE MATRIX (DSM)</b> .....	14
1. DSM METHODOLOGY .....	14
2. DSM MODEL .....	16
3. DSM ANALYSIS .....	19
4. RELATED WORK (LITERATURE REVIEW) .....	20
4.1 <i>Design Manager's Aid for Intelligent Decomposition with a Genetic Algorithm         (DeMAID/GA)</i> .....	22
4.2 <i>Problem Solving Matrix (PSM32)</i> .....	23
4.3 <i>The Analytical Design Planning Technique (ADePT)</i> .....	23
4.4 <i>A Genetic Algorithm for Decomposition of Analyses (AGENDA)</i> .....	24
4.5 <i>DSM@MIT</i> .....	25
4.6 <i>Other Models</i> .....	25
5. CRITIQUE .....	26
6. SUMMARY .....	29
<b>III. SIMULATED ANNEALING (SA)</b> .....	30
1. OPTIMIZATION .....	30
2. META-HEURISTICS .....	32
3. METROPOLIS ALGORITHM .....	33
4. THE SIMULATED ANNEALING ALGORITHM .....	33
4.1 <i>Background</i> .....	33
4.2 <i>Physical Annealing</i> .....	34
4.3 <i>The Algorithm</i> .....	35
A. <i>Generation Mechanism (Neighborhood Moves)</i> .....	36
B. <i>Objective Function Evaluation</i> .....	38
C. <i>Cooling (Annealing) Schedule</i> .....	38
5. SA: PROS AND CONS .....	41
5.1 <i>Simulated Annealing vs. Local Search</i> .....	41
6. SUMMARY .....	43

<b>IV. SIMULATION-BASED OPTIMIZATION .....</b>	<b>44</b>
1. SIMULATION .....	44
2. ADVANTAGES AND DISADVANTAGES .....	45
3. UNCERTAINTY ANALYSIS .....	46
4. MONTE CARLO SIMULATION .....	47
5. INTERFACING OPTIMIZATION METHODS WITH SIMULATION .....	49
6. RELATED WORK .....	50
7. SUMMARY.....	54
<b>V. TIME-COST TRADE-OFF IN PROJECT NETWORKS.....</b>	<b>55</b>
1. PROJECT MANAGEMENT .....	55
2. PROJECT MANAGEMENT IN MANUFACTURING.....	56
3. PROJECT CRASHING AND TIME-COST TRADE-OFF.....	58
4. TIME-COST RELATIONSHIP MODELS.....	60
5. CRITICISMS OF CURRENT PRACTICES (CONSTANT COST-SLOPE CONCEPT) .....	63
6. PROJECT CRASHING WITH MATHEMATICAL PROGRAMMING.....	63
7. THE TCTO PROBLEM IN LITERATURE .....	64
7.1 <i>Mathematical Programming</i> .....	66
7.2 <i>Heuristic Algorithms</i> .....	67
7.3 <i>Simulation</i> .....	67
7.4 <i>Artificial Intelligence</i> .....	68
8. SUMMARY.....	68
<b>VI. DSM OPTIMIZATION AND ANALYSIS .....</b>	<b>69</b>
1. OBJECTIVE .....	69
<i>The Concept of Load</i> .....	70
2. ASSUMPTIONS, AND LIMITATIONS .....	71
3. MATHEMATICAL MODELING OF FEEDBACKS .....	74
4. LOGICAL CONSTRAINTS .....	75
5. ITERATION CONSIDERATIONS.....	76
6. NUMERICAL DSM.....	77
7. COMPUTING LOAD .....	78
8. OPTIMIZATION WITH SIMULATED ANNEALING.....	81
8.1 <i>A Two-Stage Simulated Annealing Algorithm</i> .....	81
8.2 <i>Decision Variables</i> .....	81
8.3 <i>Generation of The Initial Solution Configuration</i> .....	82
8.4 <i>Generation of a Feasible Neighboring Solution Configuration</i> .....	82
8.5 <i>Object Function Evaluation</i> .....	84
8.6 <i>Cooling Schedule</i> .....	85
8.7 <i>Stopping Criterion</i> .....	86
9. HANDLING STOCHASTIC ACTIVITIES LOAD.....	86
9.1 <i>Uncertainty in Activity Load Estimation</i> .....	86
9.2 <i>Min-Mean-Max (M3) Method</i> .....	87
9.3 <i>Utility Function (UF) Method</i> .....	93
10. DSM CONVERSION INTO A PROJECT SCHEDULE .....	96
10.1 <i>Related Practice</i> .....	96



10.2 <i>The Proposed Conversion Procedure</i> .....	99
A. Patterns Recognition .....	100
B. Collapsing.....	101
C. Tearing .....	102
D. Rules of Conversion.....	106
11. SUMMARY.....	108
<b>VII. TCTO HYBRID MODEL .....</b>	<b>109</b>
1. INTRODUCTION .....	109
2. BASIC CONCEPTS .....	109
2.1 <i>Illustration of the Basic Concept</i> .....	111
3. THE PROPOSED MODEL.....	117
3.1 <i>Problem Statement</i> .....	117
3.2 <i>Assumptions and Limitations</i> .....	118
3.3 <i>Notations</i> .....	118
3.4 <i>Inputs</i> .....	119
3.5 <i>Process</i> .....	120
3.6 <i>Decision Variables</i> .....	120
3.7 <i>Constraints</i> .....	120
A. Network Constraints .....	120
B. Assignment Feasibility.....	121
C. Resource Availability Constraints.....	121
D. Project Completion Constraint.....	121
E. Activity Duration Constraints.....	121
F. Non-negativity Constraints.....	122
3.8 <i>Objective Function</i> .....	122
4. THE MODEL: HOW IS IT DIFFERENT FROM THE CALSSICAL MODEL?.....	122
5. SOLUTION: OPTIMIZATION METHODS IMPLEMENTED .....	124
5.1 <i>Simulated Annealing</i> .....	127
A. Decision Variables and Solution Representation.....	127
B. Generation of the Initial Solution Configuration .....	128
C. Generation of a Feasible Neighboring Solution Configuration .....	128
D. Objective Function Evaluation.....	128
E. Cooling Schedule.....	128
F. Stopping Criterion .....	128
6. SUMMARY.....	129
<b>VIII. ARCHITECTURE AND PRODUCT .....</b>	<b>130</b>
1. ARCHITECTURE OVERVIEW.....	130
1.1 <i>Modeling</i> .....	132
1.2 <i>Optimization (Re-Sequencing)</i> .....	132
1.3 <i>Structuring</i> .....	133
1.4 <i>Conversion to a Project Schedule</i> .....	133
1.5 <i>Scheduling</i> .....	134
1.6 <i>Crashing</i> .....	134
2. PRODUCT .....	135
2.1 <i>DSM Optimization and Analysis</i> .....	135

2.2 <i>Time-Cost Trade-Off</i> .....	138
4. SUMMARY.....	139
<b>IX. CASE STUDIES AND RESULTS .....</b>	<b>142</b>
1. PROJECT 1: BENCHMARKING .....	142
2. PROJECT 2: A CONCEPTUAL DESIGN PROJECT – DETERMINISTIC CASE.....	145
2.1 <i>Case (1)</i> .....	148
2.2 <i>Case (2)</i> .....	151
2.3 <i>Case 3</i> .....	154
3. PROJECT 3: A CONCEPTUAL DESIGN PROJECT – STOCHASTIC CASE.....	159
3.1 <i>Min-Mean-Max (M3) Results</i> .....	161
3.2 <i>Utility Function Method Results</i> .....	166
4. PROJECT 4: ILLUSTRATIVE PROJECT .....	169
4.1 <i>Optimization</i> .....	170
4.2 <i>Conversion to a Project Schedule</i> .....	172
4.3 <i>Time-Cost Trade-Off</i> .....	176
5. SUMMARY.....	180
<b>X. SUMMARY AND FUTURE WORK.....</b>	<b>181</b>
1. THE DEPENDENCY STRUCTURE MATRIX.....	182
2. SIMULATED ANNEALING.....	183
3. SIMULATION-BASED OPTIMIZATION.....	184
4. HANDLING STOCHASTIC OBJECTIVE FUNCTION.....	185
5. DSM CONVERSION TO A PROGRAM.....	186
6. TCTO HYBRID MODEL.....	187
7. OPTDSM.....	188
9. LESSONS LEARNED.....	189
<b>REFERENCES.....</b>	<b>191</b>
<b>APPENDIX A: OPTDSM.....</b>	<b>204</b>
1. OVERVIEW .....	204
2. FUNCTIONS .....	205
2.1 <i>New: DSM</i> .....	205
2.2 <i>Create DSM</i> .....	208
2.3 <i>Optimization: Settings</i> .....	208
2.4 <i>Optimization: Start</i> .....	211
2.5 <i>Optimization: Final DSM</i> .....	213
2.6 <i>Post Optimization: Collapsed DSM</i> .....	213
2.7 <i>Project</i> .....	213
3. FUTURE MODIFICATIONS .....	214

## LIST OF TABLES

Table	Page
1. Summary of DSM Characteristics .....	16
2. Comparison of Different DSM analysis tools.....	28
3. Summary of various simulation-based optimization research.....	54
4.1 Iteration Factor Values.....	77
5. Utility Function indices.....	95
6. I-Pattern Conversion Rules.....	104
7. C-Pattern Conversion Rules.....	104
8. L-Pattern Conversion Rules.....	105
9. Resource-Levels.....	113
10. Cost Calculations for the Initial Configuration (Assignment).....	114
11. Cost calculations for the Final Configuration (Assignment).....	116
12. Summary of Architecture Functions.....	140
13. Summary of Architecture Options.....	141
14. optDSM Sequence vs. AGENDA Sequence.....	144
15. Project 2: Activities List .....	147
16. Project 2: Couplings List .....	147
17. Case (1): Results .....	148
18. Case (2): Results .....	151
19. Case (3): Results .....	155
20. Project 3: Activities List .....	160
21. M3: Results.....	162
22. Utility Function Method: Results.....	166
23. Project 4: Activities List .....	169
24. Project 4: Couplings List .....	169
25. Project 4: Logical Constraints.....	169
26. Conversion to a Project Schedule .....	172
27. Activities List of the Equivalent Schedule.....	176
28. Resource Data.....	177

Table	Page
29. Resource Assignment for 56 hours .....	178
30. Resource Usage.....	178
31. Case Projects Summary .....	180

## LIST OF FIGURES

Figure	Page
1. Cost Impact .....	4
2. Research Motivation Mind Map .....	4
3. Rich Picture Diagram Representing the Problematic Situation.....	7
4. Layout of the Dissertation at a Glance.....	13
5. A Typical Dependency Structure Matrix (DSM).....	18
6. Feedback vs. Feed forward Couplings.....	18
7. DSM Analysis.....	21
8. Metropolis Algorithm .....	34
9. Generic Simulated Annealing Algorithm .....	36
10. Flowchart of the Standard (Naïve) SA Algorithm.....	37
11. Black Box Optimization .....	38
12. Local Search.....	42
13. Simulated Annealing.....	43
14. System Experimentation with Simulation .....	45
15. Monte Carlo Simulation.....	48
16. Linear Time-Cost relationship .....	61
17. Different Time-Cost Models.....	62
18. Schematic Representations of DSM Analysis Steps.....	70
19. Assumption: System Decomposition.....	73
20. Assumption: Activity Start .....	73
21. Assumption: Activity Output.....	73
22. Coupling (General Form).....	74
23. Feed forward vs. Feedback Coupling .....	75
24. Hard Constraint.....	76
25. Load Computations Heuristic .....	80
26. Load computations example .....	81
27. Two-Stage Simulated Annealing .....	83
28. Solution Representation.....	84

Figure	Page
29. Generating a Neighboring Solution .....	84
30. Cases with $\mu_p < \mu_c$ (Schematic) .....	89
31. First Set of Comparison Rules .....	90
32. Cases with $\mu_p = \mu_c$ (Schematic) .....	91
33. Second Set of Comparison Rules .....	92
34. Robust vs. Optimal solution .....	94
35. DSM of a Hypothetical Project .....	97
36. Converting the Final DSM to a Project Schedule .....	98
37. Different patterns .....	103
38. Equivalent DSM for C4-Pattern .....	105
39. Equivalent DSM for S-Coupling .....	106
40. Equivalent Schedule for the Comparison Case According to the Proposed Methodology .....	107
41. Resource Levels .....	110
42. Different Assignments Result in Different Activity Durations .....	111
43. Project Network .....	113
44. Project Schedule Corresponding to Initial Assignment .....	113
45. Gantt Chart Showing the Three Paths .....	114
46. Activities' Slacks .....	115
47. Crashing of Activity E .....	115
48. Relaxing of Activity D .....	116
49. Models comparison .....	124
50. TCTO Optimization Macro Flowchart .....	126
51. Solution Representation .....	127
52. The Proposed Architecture .....	131
53. Optimization Cases .....	134
54. Simulation-based Optimization Framework .....	137
55. TCTO Module .....	139
56. Solution Obtained by AGENDA .....	143
57. Solution Obtained by optDSM .....	143

Figure	Page
58. Process Flowchart .....	146
59. Initial DSM Sequence .....	146
60. Case (1): DSM with Minimum Number of Feedbacks .....	149
61. Case (1): Meta-stabel Objective Functions at Different Temperatures .....	150
62. Case (2): DSM with Minimum Total Load.....	152
63. Case (2): Meta-stabel Objective Functions at Different Temperatures .....	153
64. Case (3): DSM with Minimum Total Time and Cost .....	156
65. Case (3): Meta-stabel Objective Functions at Different Temperatures .....	157
66. Case (3): Accepted Solutions.....	158
67. Probability Distributions' Parameters.....	159
68. M3: DSM Corresponding to the Optimal Solution.....	163
69. M3: Probability Distribution of the Initial Solution .....	163
70. M3: Probability Distribution of the Optimal Solution.....	164
71. M3: Meta-stable Objective Function Values .....	164
72. M3: Solution Robustness .....	165
73. M3: Accepted Solutions.....	165
74. Utility Function Method: DSM.....	167
75. Utility Function Method: Meta-stable Objective Function Values.....	167
76. Utility Function Method: Solution Robustness.....	168
77. Project 4: Initial DSM.....	170
78. Project 4: DSM with Minimum Number of Feedbacks .....	171
79. Project 4: DSM with Minimum Total Load.....	171
80. Conversion to a Project Schedule: Step 1 .....	173
81. Conversion to a Project Schedule: Step 2 .....	173
82. Conversion to a Project Schedule: Steps 3 and 4.....	174
83. Conversion to a Project Schedule: Step 5 .....	174
84. Conversion to a Project Schedule: Step 6 .....	175
85. Conversion to a Project Schedule: Steps 7, 8, and 9.....	175
86. TCTO Curve .....	177
87. Gantt Chart (for 56 hours Project Duration).....	179

Figure	Page
88. Meta-stable Objective Function; Cost (for 57 hours Project Duration).....	179
89. optDSM Main Menu.....	205
90. New DSM.....	206
91. Data Entry Tables.....	207
92. Data Entry Tables Filled.....	207
93. Optimization Sub-menu.....	208
94. Optimization Options/Settings window.....	210
95. Feedback Calculations Sheet.....	211
96. Screen Shot: Deterministic Optimization.....	212
97. Screen Shot: Stochastic Optimization.....	212
98. Post Optimization Sub-menu.....	213



# CHAPTER I

## INTRODUCTION

### 1.1 Introduction to the Domain: Research Motivation

Manufacturing firms in the United States have almost universally recognized the need to reconsider traditional methods of product development and introduction [1]. In order for a product to be competitive, it needs to be introduced quickly without compromising product performance [2]. This is so because products that meet the needs of customers faster than competitors grow at a rapid pace, both in terms of market share and profitability [3]. Thus, reduction in product development cycle time has become an essential goal [1]. The significance of time-to-market is further demonstrated by [4-6].

In a 1991 pamphlet issued by the National Research Council, [7], four requirements for using design as a source of competitive advantage were cited: (1) committing to continuous improvement both of products and of design and production processes, (2) establishing a corporate Product Realization Process (PRP) supported by top management, (3) developing and/or adopting and integrating advanced design practices into the PRP, and (4) creating a supportive design environment. Moreover, incorporating the following steps was defined as effective PRP practice: (1) defining customer needs and product performance requirements, (2) planning for product evolution beyond the current design, (3) planning concurrently for design and manufacturing, (4) designing the product and its manufacturing processes with full consideration of the entire product life cycle, and (5) producing the product and monitor product and processes. In this spirit, the term 'Integrated Product Development' (IPD) was coined to describe a process that has been adopted by most progressive manufacturing firms, even though firms may have different names for this process [8].

---

The format of this thesis is based on "The American Society of Mechanical Engineers Transactions Journals"

The basic disciplines for making progress in manufacturing belong not only to mechanical engineering, but also to industrial engineering, mathematics, management science, and computer science. These separate disciplines are individually supported by their research, methods, and software. There is a lack of focused attention on how to integrate knowledge from many disciplines into knowledge that furthers manufacturing goals. Moreover, at the same time that this lack of strategy is apparent, all dimensions of manufacturing (e.g. products, processes, markets) are becoming more complex and diverse. Complex new products based on massive information content and information-dominated design and manufacturing methods already require us to deal with an entirely new scale of complexity. Providing tools to facilitate and manage the complexity of this information and computation intensive activities plays an important role in supporting and even enabling the complex practice of manufacturing.

The difficulties in designing complex engineering products do not arise simply from their technical complexity but rather from the managerial complexity necessary to manage the interactions between the different engineering disciplines, which imposes additional challenges on the design process [9]. As a result, a systems level solution must be determined and deployed. Integrated product development is a holistic approach that helps to overcome problems that arise in complex product development environments. Integrated product development was defined by Fiksel [1] as:

“a process whereby all functional groups (e.g. engineering, manufacturing, marketing, etc.) that are involved in a product life cycle participate as a team in the early understanding and resolution of key product development issues including quality, manufacturability, reliability, maintainability, environment, and safety.”

IPD is based on Concurrent Engineering (CE), but goes beyond CE with regard to the level of integration. In the scope of IPD, designers, assembly planners and production planners, as well as persons responsible for quality or testing not only consult themselves while they are working simultaneously on their tasks, but exchange interconnected intermediate results in a continuous interplay [10,11].

The motivation for adopting IPD can be further understood when the economics of product development are considered; where between 60 and 80 percent of the overall

product costs are committed between the concept and preliminary design phases of the program [12]. And since only a small cumulative expenditure of funding is committed during early phases in the classical serial approach, the cost of design change increases exponentially as the development process advances as shown in Fig.1. For example, in the automotive and electronics industry, it has been shown that up to 80% of product life-cycle costs are committed during the concept and preliminary design stages, and that the cost of design changes steeply increases as a product proceeds into full-scale development and prototyping [7, 8, 13]. Another study included in [14] showed that about 70% of the life cycle cost of a product is determined at the conceptual design stage. Furthermore, O'Grady et al. [15] showed that design of products determines their quality and 70% to 80% of the final production cost.

Thus, the motivation for current research can be summarized as follows:

1. There is a need to reduce both product time-to-market and product development cost.
2. Dealing with complex products adds more difficulty to the management of the design process within product development projects.
3. As a result, a more effective methodology – that is IPD - has to be implemented.
4. Since about 80% of cost is committed at early development phases, the current research focuses on improving the product design process.

Figure 2 is a simple mind map of research motivation.

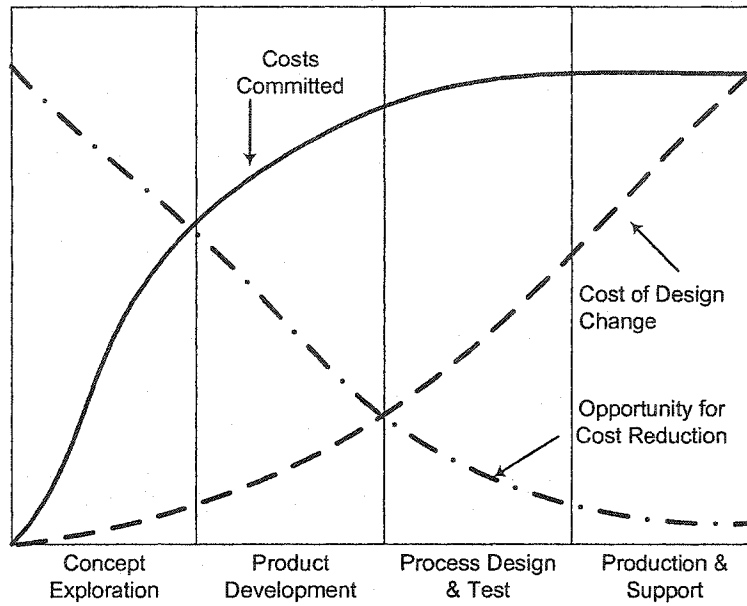


Figure 1. Cost Impact (Reference: [12]).

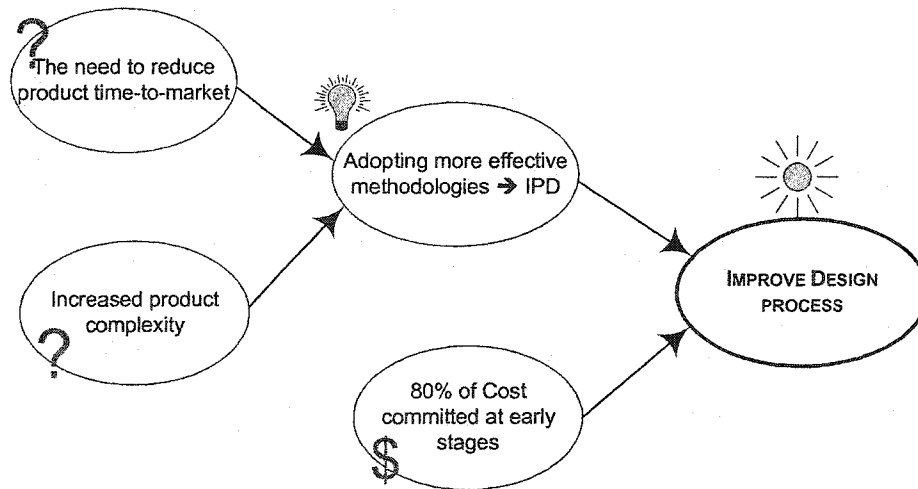


Figure 2. Research Motivation Mind Map.

## 1.2 Introduction to the Context: Problem Framing

Integrated product development is a general procedure for product development with focus on integration between market, product and production aspects when creating new business [16]. The concept of IPD was developed independently by two sources. From 1984 to 1988, IPD was introduced into Danish industry (mechatronic products) in a country-wide campaign [17, 18]. In the late 1980's, IPD was introduced by the U.S. defense industry [12].

An IPD emphasis is to provide a framework for effective planning and managing of engineering projects. To clarify the presented problematic situation, the *rich picture diagram*<sup>1</sup> shown in Fig.3 is used to explore the connections and interdependencies among the different components of the IPD approach to present its complexity, on one hand, and to help defining both the WSOI (wider system of interest) and NSOI (narrower system of interest) on the other hand.

The seven phases of the IPD approach, shown in the rich picture diagram, represent the WSOI. The NSOI, which is the focus of the current research, consists of the first four phases together or, in general, the *design process*. The design process itself is typically a complex system. The main approach to handling such a system is to build a model that imitates the real system (or desired system in our case). Typically, this includes: (1) defining the system of interest, (2) defining the system boundary, (3) decomposing the system into sub-systems and further into smaller components, and (4) defining the relationships among these components. Following these steps, the system will be decomposed into possibly several hundred activities (components) and thousands of variable interchanges among these activities. The sequence of performing these activities strongly affects the time (and hence the cost) needed to realize the whole project. Furthermore, a common characteristic of such projects is the involvement of teams from different disciplines, working simultaneously on different aspects of the project

---

<sup>1</sup> A rich picture is “a cartoon-like summary of everything (or almost everything!) the observer knows about the situation studied [19].”

associated with their analyses tools and software. So, a tool is needed for arranging (sequencing) the activities of the project for efficient execution, and moreover, to decompose the project into sub-projects (circuits) such that each can be performed at a certain discipline by a certain team. Successful project management requires the effective control of the design teams and the exchange of information between them for successful design management [20].

Furthermore, project managers, in addition to scheduling projects, are frequently confronted with the problem of having to reduce the scheduled completion time indicated by the Critical Path Method (CPM) in order to meet a pre-specified deadline. Project duration reduction, or project crashing, can be achieved by assigning more resources (labor, material, equipment, etc.). However, additional resources cost money and, hence, increase the overall project cost. Thus, the decision to reduce the project duration, and by how much, must be based on an analysis of the trade-off between time desired project duration to be reduced and the extra cost needed.

Thus, the complexity of the presented problem arises from the following sources:

1. Large number of components.
2. Complex interactions scheme.
3. Uncertainty in components duration and cost.
4. Hard precedence constraints.
5. Resource limitations.

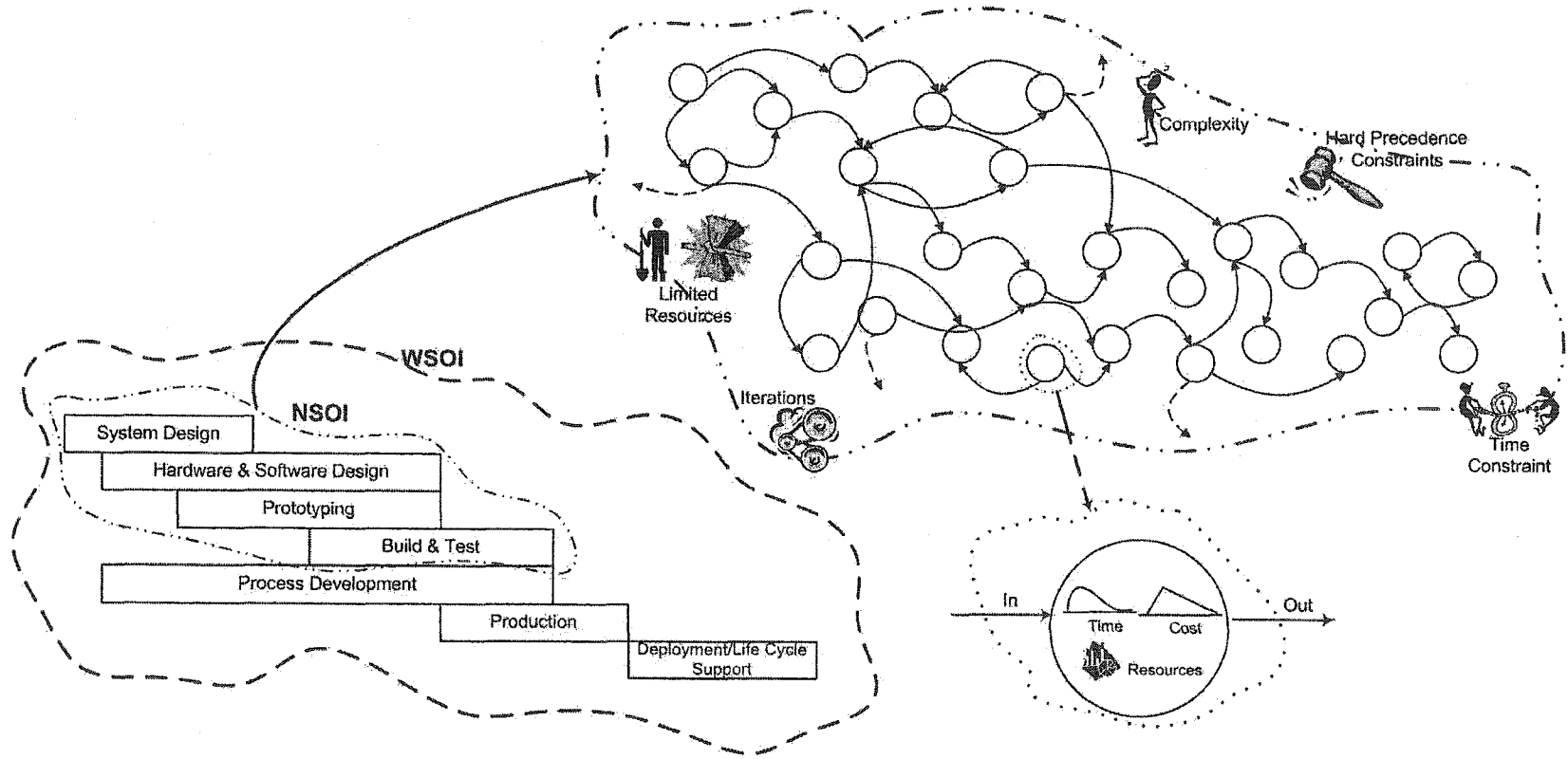


Figure 3. Rich Picture Diagram Representing the Problematic Situation

### 1.3 Introduction to the Research: Research Objectives

Today, the highly competitive market pushes companies not just to deliver products that work, but also to introduce these products to market as early as possible (i.e. minimum time-to-market or shorter product development cycle). A great deal of effort has been committed to developing and deploying more powerful analysis tools but, unfortunately, little work has been done in creating methods and tools for analyzing and improving the design process itself [21].

Management tools that model the interface and dependencies among process activities contain the managerial complexity of the design process. Yassine et al. [9] stated that managing the design process includes four major steps: (1) modeling of the information and dependency structure of the design process, (2) providing a design plan showing the order of execution for the design activities, (3) reducing the risk and magnitude of iteration between design activities, and (4) exploring opportunities for reducing the project cycle time.

**The main objective of the presented research is to develop an optimization-based architecture that helps guide the project manager efforts for managing the design process in complex integrated product development projects. The presented work contributes to five areas of research.**

Improving system performance can be achieved through efficient re-engineering of its structure. The Dependency Structure Matrix (DSM) provides an effective tool for system structure understanding. The first research contribution aims toward finding an optimal activity sequence of the DSM representing a design project in terms of load, time, and cost. To achieve this goal, a mathematical program representing the DSM structure was developed and a meta-heuristic optimization algorithm called Simulated Annealing (SA) was implemented to solve this program.

One unavoidable pitfall in the estimation of activity time and cost is uncertainty that arises from many different sources. Although uncertainty cannot be eliminated,



incorporating it in the model can reduce its effect. Thus, the estimate of activity time and cost can be in the form of a probability distribution, not as a single point value. The second research area concerns handling uncertainty in activity loads, time, and/or cost requirements. A simulation-based optimization framework that integrates simulated annealing with a commercial risk analysis software called Crystal Ball<sup>TM2</sup> was developed to optimally re-sequence the DSM activities given stochastic activity data.

Since simulated annealing was originally developed to handle deterministic objective functions, the third research area involves modifying the SA algorithm to tolerate stochastic objective functions (multi-point estimate) rather than deterministic ones (one-point estimate). The goal here involves determining a robust solution rather than an optimum (minimum) one.

For the DSM to serve as a means of controlling the design project (continual re-planning, re-scheduling, and follow up), activities in the optimally re-sequenced DSM need to be represented against a time scale. In other words, the DSM has to be converted into a project schedule. The fourth contribution of this research is providing a methodology for the conversion of the optimally sequenced DSM into an equivalent DSM that contains no feedback couplings. Once an equivalent DSM is obtained, a project schedule can be developed and the use of scheduling methods becomes feasible.

The fifth and final area presents a new time-cost trade-off model for project networks. The new model is a hybrid model that joins the resource assignment problem with project crashing. The presented model is based on the trade-off of resources where, in some cases, it may be possible to transfer persons, equipment, or other resources from a non-critical activity to a critical one. Thus, it helps crashing a project with little, or no, additional cost.

These areas of research will be further implemented through a developed excel add-in called "optDSM". The tool was developed by the author using Visual Basic for

---

<sup>2</sup> Developed by Decisioneering, Inc.

Application (VBA) programming language. Among its several modules, optDSM has the ability to interface with Crystal Ball™ to carry out the optimization process in cases where activity loads assume stochastic values. The main functions of optDSM are:

1. Modeling of the project under consideration in the form of a DSM.
2. Finding the optimum sequence of DSM activities based on a user selected objective function.
3. Producing a DSM equivalent to the optimized one but without feedback couplings.
4. Converting the structured DSM into a project schedule.

While the first two functions are fully operational, the later two are still under development.

## **1.4 Introduction to the Technology: Analytical Strategy and Solution**

### **Potential**

The presented architecture integrates several conceptual tools. These include:

1. The Dependency Structure Matrix (DSM): the cornerstone of the presented architecture. DSM improves understanding of the project - or the system - being analyzed by providing a compact visualization of the project and a clear understanding of the information flow patterns among its activities.
2. Mathematical Programming (MP): a basic step in any optimization process is building a model that represents the system under consideration. MP was used in this dissertation to:
  - a. Translate the DSM from its visual form into a mathematical form.
  - b. Describe a project network by a set of equations (relationships, constraints, and objective function).
3. Simulated Annealing (SA): a meta-heuristic optimization algorithm with proven efficiency in solving hard combinatorial optimization problems. In this dissertation, SA was modified - to fit the architecture needs - and used in both

optimization components of the architecture; DSM re-sequencing, and time-cost trade-off.

4. Monte Carlo (MC) Simulation: an important decision-support tool in for wide variety of disciplines. The architecture deploys MC – through an interface with Crystal Ball™ – as a risk analysis tool to tackle cases in which project activities assume stochastic data.
5. Critical Path Analysis (CPM): a fundamental scheduling method used in project management. CPM is based on a mathematical model that calculates the total duration of a project and identifies critical activities. CPM serves as the basis for the TCTO model incorporated in the architecture.

Application wise, the architecture:

1. Is a user-friendly tool for IPD project managers that provide:
  - a. Compact visualization of the project (activities and interfaces).
  - b. Clear understanding of project structure.
  - c. Efficient tool for time and cost reduction of design projects.
  - d. Improved final design quality.
  - e. Risk analysis tool.
2. Integrates off-shelf applications (MS Excel, Crystal Ball, and MS Project) with an efficient optimization algorithm (SA).
3. Provides ways to facilitate and manage the complexity of the information intensive activities during product design process.
4. Is implemented in a spreadsheet environment due to the familiarity of this software tool in the engineering and business communities.
5. Provides a promising efficient methodology for resource assignment.

## 1.5 Introduction to the Dissertation: Reader's Guide

As shown in Fig. 4, the dissertation is divided into two main parts: Introduction and Background (Chapters II to V), and Research Methodology and Results (Chapters VI to X).

Following the introduction chapter, the dissertation is organized as follows.

Chapter II presents the DSM methodology and reviews five prototypes related to the tool developed in the presented research. Chapter III provides background on Simulated Annealing algorithm describing its basic principal, concepts, implementations, and advantages over classical optimization algorithms. In Chapter IV, a short introduction to simulation is provided along with a literature review of work related to interfacing optimization methods with simulation. Finally, the concepts of project crashing and time-cost trade-off analysis, and reviews different methods used to tackle this problem cited in literature are introduced in Chapter V.

Chapter VI discusses all aspects related to the optimization and analysis of the DSM contributed by this research. In Chapter VII, the proposed time-cost trade-off model is presented. Basic concepts, illustrative examples, mathematical model, and implementation details are provided. Chapter VIII presents the conceptual architecture proposed in the dissertation for managing Integrated Product Development projects. Case studies used to present the performance of the architecture are provided in Chapter IX. Finally, summary; discussions of the results and future research directions are provided in Chapter X followed by Appendix A that introduces the implementation tool; "optDSM."

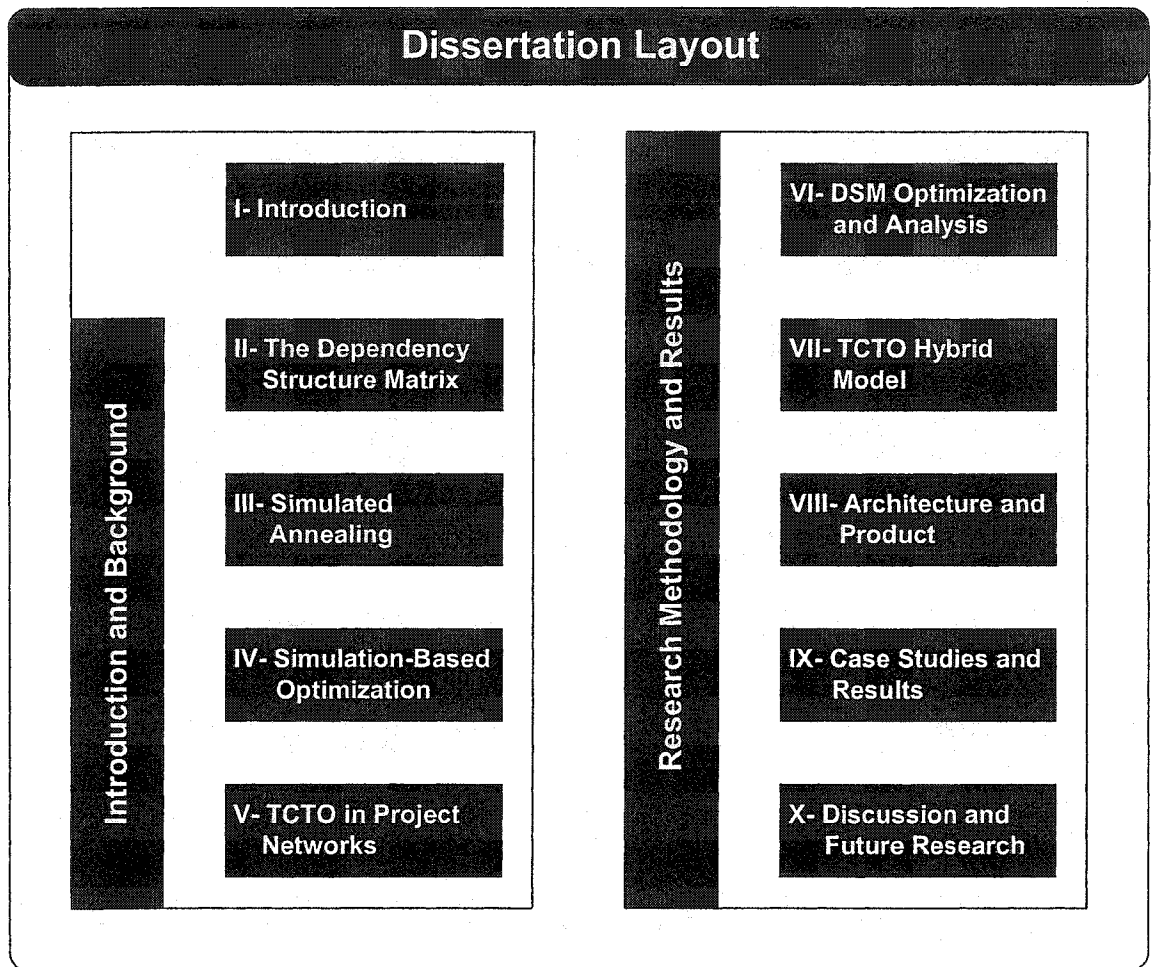


Figure 4. Layout of the Dissertation at a Glance.

## CHAPTER II

### THE DEPENDENCY STRUCTURE MATRIX (DSM)

#### 2.1 The DSM Methodology

Integrated product development projects (IPDPs) are complex systems. A prerequisite to improvement is system understanding. Systems can be described by their structure - presented by a graph or a matrix showing which components affect what other components, and by semantics (concern how these effects occur) [22]. System structure (or architecture) affects its efficiency and effectiveness [23, 24]. Therefore, it can be an important source of competitive advantage [25]. Improved understanding of system architecture can be gained by using process models [23]. These models must be able to capture the decomposed system activities, their information interfaces (or couplings), and enables associated integration analysis.

A product development project (PDP) fundamentally differs from a construction (or a manufacturing) project in two major aspects:

1. While the later is activity-based (i.e. an activity to be carried out only when its predecessors are physically done), the first is information-based (activities execution and results are based mainly on information exchanged with other coupled activities).
2. A typical PDP is characterized by its highly coupled, interdependent activities, which must converge iteratively to an acceptable design solution [23, 26]. There always exists a high possibility of many activities that need to be repeated before the desired specifications are met. The most common causes for such repetition (known as feedback loops in DSM terminology) are due to activities that begin work without the necessary information; the arrival of new information; change of information that leads to rework; or re-evaluated assumptions in previous activities [24, 27-29].

A project typically consists of a number of interrelated activities. For more than fifty years, a number of techniques, such as Program Evaluation and Review Technique (PERT) and Critical Path Method (CPM), have been used to handle complex projects [30]. Unfortunately, these methods succeed only if activities are sequential and/or parallel, but fail significantly if there are iterative sub-cycles since they do not tolerate feedback relationships.

Although the idea of representing the system architectural components and relationships in the form of a matrix is not new, the term “Design Structure Matrix” (DSM) was coined by Steward [22, 31] to denote a generic matrix-based model for project information flow analysis. Since then, the DSM is becoming a popular representation and analysis tool for system modeling, especially for purpose of decomposition and integration [26]. Because it can be used in many other areas, besides design, Stephen Denker, a member of Project Management Institute (PMI), termed it the “Dependency Structure Matrix.”<sup>3</sup>

Since its original introduction by Steward in the 1980’s, the DSM has been extended to cover many application areas. Browning [26] presented four main DSM applications. These applications are summarized in Table 1. This research interest is activity-based DSM.

---

<sup>3</sup> In this research, the term “Dependency Structure Matrix” is used instead of “Design Structure Matrix.”

Table 1. Summary of DSM Characteristics.

DSM		Representation	Application	Integration Analysis via
Category	Type			
Static	Component-Based or Architecture	Components in a product architecture and their relationships	System architecting, engineering, design, etc.	Clustering
	Team-Based or Organization	Individuals, groups, or teams in an organization and their relationships	Organization design, interface management, application or appropriate integration mechanisms	
Time-Based	Activity-Based or Schedule	Activities in a process and their inputs and outputs	Project scheduling, activity sequencing, cycle time reduction, risk reduction, etc.	Sequencing, Partitioning, and Tearing
	Parameter-Based	Parameters to determine a design and their relationships	Low-level process sequencing and integration	

Ref. [26]

## 2.2 DSM Model

The basic DSM is a simple binary<sup>4</sup>  $n$ -square matrix - where  $n$  is the number of system components, with  $m$  non-empty elements - where  $m$  is the number of dependencies (information interfaces or couplings) among different system components. A sample DSM is shown in Fig. 5. In this example, the system under consideration is a design project that consists of 14 activities. So, the DSM is a 14x14 matrix. Project activity names are placed on the left-hand side of the matrix as row headings and across the top row as column headings in the same order (order of their execution)<sup>5</sup>. Off-diagonal marks (X) represent coupling (information flow, or dependency) between two activities. If an activity  $i$  depends on (receives information from) activity  $j$  (where  $i, j \in (1, n)$ ), then the matrix cell  $ij$  (row  $i$ , column  $j$ ) contains an off diagonal mark (X) otherwise the cell is empty. As a result, reading across a single row of the DSM reveals information provided to the activity corresponding to that row (i.e. off-diagonal marks on that row correspond to activities whose output is required to perform the activity under consideration). On the

<sup>4</sup> A cell can hold one of only two values (0, 1), or in other cases ("X" mark, empty cell).

<sup>5</sup> A main DSM assumption is that activities are undertaken in the order listed from top to bottom.



other hand, reading down a specific column reveals information flow from the activity corresponding to that column (i.e. which activity receive information from the activity under consideration). For example (illustrated in Fig. 6) activity 4 provides information to activity 1 (a feedback coupling) and to activity 7 (a feed-forward coupling), while receives information from activity 9 (a feedback coupling)

Marks below the diagonal (sub-diagonal marks) are indicative of feed-forward couplings (i.e. from upstream activities to downstream activities), while those above the diagonal (super-diagonal) represent feedback couplings (i.e. from downstream tasks to upstream activities)<sup>6</sup>. As they imply iterations, the latter type of couplings should be eliminated if possible or reduced to the maximum extent. If certain feedback couplings cannot be eliminated, the activities are grouped into iterative sub-cycles. For example, in Fig. 5 activities (1,2,3), and (6,7,8,9,10) are grouped into two iterative sub-cycles (blocks).

Three basic types of activity interactions can be observed in Fig. 5: (1) Activity 4 and Activity 5 are 'independent' (can be carried out concurrently since no information is exchanged between them), (2) Activity 11 and Activity 12 are 'dependent' (they must be carried out sequentially, i.e. Activity 12 needs information (output) from Activity 11 to start), and (3) Activities 13 and 14 are 'coupled' (each activity needs information from the other) in this case Activities 13 and 14 are called a 'circuit.'

---

<sup>6</sup> The convention used in this research is the one proposed by Steward [22]. Other researchers (following Rogers [36]) used a reversed convention by placing feedback couplings below the diagonal, and feed forward couplings above the diagonal.

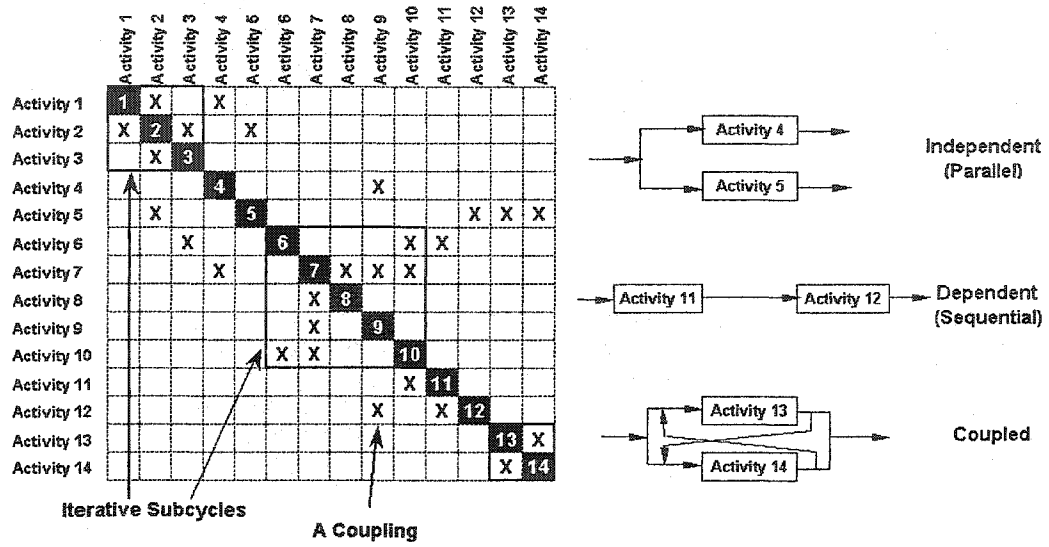


Figure 5. A Typical Dependency Structure Matrix (DSM).

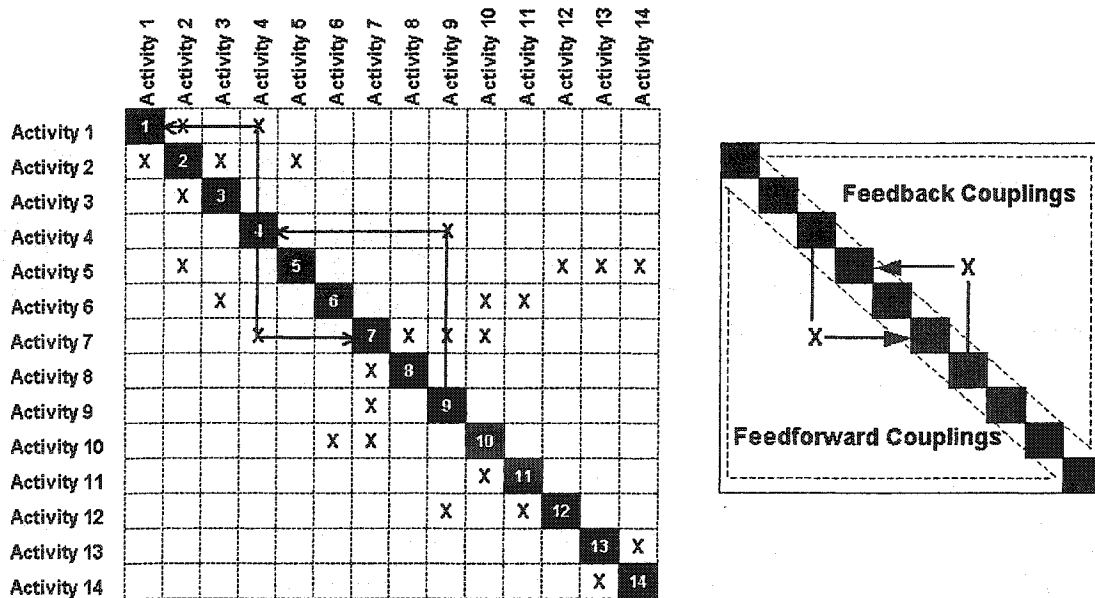


Figure 6. Feedback vs. Feed forward Couplings.

### 2.3 DSM Analysis

The activity-based DSM<sup>7</sup> provides an effective visual format for understanding and analyzing information flow-based projects such as integrated product development projects.

Generally, the application of an activity-based DSM involves the following steps:

1. Decomposition of the system under consideration into its smallest components (activities)<sup>8</sup>.
2. Defining the information flow interfaces (couplings) among these activities.
3. Analyzing the sequence of executing the activities with the goal of minimizing the feedback flow.
4. Grouping coupled activities into circuits (blocks or iterative sub-cycles).

Thus, a primary goal in basic DSM analysis is to minimize feedback couplings and their scope by restructuring or re-architecting the process [26], i.e. by re-sequencing the execution of activities to get the DSM into as low a triangular form as possible.

Steward [22] proposed a two-phase approach to achieve this goal. Phase one is called partitioning and phase two is called tearing. Partitioning is based on system structure and involves re-sequencing the DSM activities in order to: (1) eliminate feedback couplings as much as possible, (2) pull the rest of the feedbacks close to the diagonal as possible, and finally (3) group the activities into blocks such that each block represents an iterative sub-cycle. In the second phase, tearing, each block resulted from phase one being considered individually. Tearing is based on the semantics of the systems and aims to relatively order the activities within each block to achieve the same previous objectives. Tearing involves the following steps: (1) choosing a set of feedback couplings that can be ignored based on the semantics of the system, (2) tearing these arcs so that no circuit exists, (3) re-ordering the activities within the block by partitioning, (4) if new smaller nontrivial blocks result, then the process is to be repeated, otherwise stop. For further details, the reader is referred to [22, 31].

---

<sup>7</sup> A time-based DSM. Through out this dissertation, the term DSM will be used to indicate an activity-based DSM.

<sup>8</sup> The terms 'Task' and 'Activity' are used interchangeably in this research with the same meaning.

In addition to the partition heuristic provided by Steward [22], several methods to determine iterative blocks are found in literature: the Path Searching method [32], the Reachability Matrix method [33], the Triangularization Algorithm [34], and the Powers of the Adjacency Matrix Method.

Figure 7 is an illustrative example of this DSM analysis. Figure (a) shows the DSM corresponding to the original activity sequence. This sequence results in 6 feedback couplings. The partitioned DSM, shown in Fig. b, shows the existence of two iterative blocks: tasks 1-7, and tasks 2-3-6-9 respectively. Notice that the number of feedback couplings was reduced by one as a result of activity re-sequencing. The analysis proceeded with tearing of the second block (Fig. c), which resulted in further reduction of the feedback couplings to 4. With the possibility of further improvement, another tearing procedure was carried out on the same block, which resulted in the final DSM (Fig. d) with only 3 feedbacks brought as close as possible to the diagonal. It can be noticed from this example that achieving the activity final order in Steward's approach is mainly dependent on the tearing phase that constitutes a major drawback here since: (1) this required high user interaction, and (2) there is no optimal method for tearing. In other words, the final solution is based on user experience and knowledge.

## 2.4 Related Work

Although the theory of DSM has been applied in many areas, most of its research work has focused on deploying DSM to manage engineering design projects [26, 35]. A large number of DSM-related analysis models were proposed in literature. But, for the sake of current research interest, reviewing the body of literature will focus mainly on available prototype DSM tools regardless of their application area.

	Activity 1	Activity 2	Activity 3	Activity 4	Activity 5	Activity 6	Activity 7	Activity 8	Activity 9
Activity 1	■						x		
Activity 2		■	x			x			x
Activity 3	x	x	■						
Activity 4	x			■					
Activity 5					■				
Activity 6					x	■			x
Activity 7	x						■		
Activity 8				x				■	x
Activity 9			x			x			■

(a) Original sequence

	Activity 5	Activity 1	Activity 7	Activity 2	Activity 3	Activity 6	Activity 9	Activity 4	Activity 8
Activity 5	■								
Activity 1		■	x						
Activity 7		x	■						
Activity 2				■	x	x	x		
Activity 3		x		x	■				
Activity 6	x					■	x		
Activity 9					x	x	■		
Activity 4		x						■	
Activity 8						x	x		■

(b) Partitioned DSM

	Activity 5	Activity 7	Activity 1	Activity 6	Activity 2	Activity 3	Activity 9	Activity 4	Activity 8
Activity 5	■								
Activity 7		■	x						
Activity 1		x	■						
Activity 6	x			■				x	
Activity 2				x	■	x	x		
Activity 3			x		x	■			
Activity 9				x		x	■		
Activity 4			x					■	
Activity 8							x	x	■

(c) After tearing

	Activity 5	Activity 7	Activity 1	Activity 3	Activity 9	Activity 6	Activity 2	Activity 4	Activity 8
Activity 5	■								
Activity 7		■	x						
Activity 1		x	■						
Activity 3			x	■				x	
Activity 9				x	■	x			
Activity 6	x				x	■			
Activity 2				x	x	x	■		
Activity 4			x					■	
Activity 8					x			x	■

(d) Final DSM after second tearing

Figure 7. DSM Analysis (Ref. Steward [22]).

#### **2.4.1 Design Manager's Aid for Intelligent Decomposition with a Genetic Algorithm (DeMAID/GA)**

In 1989, a new knowledge-based tool [36-38] was released to the public. This tool, called the Design Manager's Aid for Intelligent Decomposition (or DeMaid), was aimed at aiding the design manager understanding the interactions among different components of a large and complex system. This original version of DeMaid included functions for minimizing the feedback couplings; sequencing the design processes; grouping processes into iterative sub-cycles; decomposing these sub-cycles into a hierarchical, multilevel structure for a design project; and displaying the sequence of processes in a (DSM) format [39]. Since its first release, DeMaid has witnessed many enhancements. In 1992, two enhancements were incorporated into it, these were: (1) an enhancement for enabling DeMaid to order the activities of an assembly line problem, and (2) an enhancement that allows the design manager to see what activities must be redone if a change is made in some input data [40]. A following major step was taken in 1994 by incorporating a new feature for DeMaid that allows the design manager to use coupling strength information to find a proper sequence for ordering the design activities [41].

A major shortcoming of DeMaid was basing its reordering procedure on barely reducing the number of feedbacks. But, the rapid expansion of Genetic Algorithms (GAs) applications has provided the basis for the next major enhancement, which is the addition of the GA to optimize the sequence of processes within an iterative sub-cycle. This GA examines a large number of orderings of processes in each iterative sub-cycle and optimizes the orderings based on cost, time and iteration requirements [42]. The name thus became DeMaid/GA. Finally, two interface functions were added to DeMaid/GA [39]: (1) optional displaying of the DSM in Steward's original format, and (2) the ability to save a file that can be input to other management tools (namely spreadsheets, and project management).

### 2.4.2 Problem Solving Matrix (PSM32)

PSM32 was developed in the 1990s by 'Problematics, LLC<sup>9</sup>'. While DeMAID/GA operates in Unix and Macintosh environments, PSM32 operates in a Windows<sup>TM</sup> environment. The DSM is built through either the direct input of activities and dependencies to the matrix, or by importing a data file pre-configured for this purpose. The software is mainly the application of Steward's methodology and has three main functions: (1) Partitioning, (2) Tearing, and (3) Impact/change tracing.

### 2.4.3 The Analytical Design Planning Technique (ADePT)

Over the period from 1994 to 2000, the Analytical Design Planning Technique (ADePT) was developed to offer an approach to planning construction design projects. The use of the ADePT methodology constitutes an important application of DSM analysis to highly complex design projects.

The ADePT methodology consists of three consecutive, yet integrated stages. The first stage involves modeling of the detailed design activities and their associated information requirements. The modeling process is based on a modified version of the IDEF0 methodology (for more details on this methodology refer to [43]) and breaks down the plan of work into five main disciplines (architecture, civil engineering, structural engineering, mechanical engineering, and electrical engineering). The data from the first stage is fed to an information dependency table that, in turn, is used to build a dependency structure matrix that constitutes the basis for the second stage. The second stage identifies iterations within the design process and arranges the activities (using a partitioning algorithm) with the objective of optimizing the task order. The third, and last stage, produces a design program (network) based on the partitioned DSM. The ADePT methodology requires some iteration between the second and third stages. Further

---

<sup>9</sup> <http://www.problematics.com>

detailed descriptions of the three stages of the ADePT methodology can be found in [35, 43-45].

A related valuable research was conducted by Baldwin et al. [46]. The paper presented a methodology that incorporates discrete event simulation, data flow diagrams, and DSM to help the planning and control of building design. Activity durations and resources are allocated along with any other specific constraints to evaluate the project schedule.

#### 2.4.4 A Genetic Algorithm for Decomposition of Analyses (AGENDA)

A method for structuring problem activities with optimal ordering and decomposition into sub-problems was described in [47]. Despite of the generality of the used method, in principle, the paper focused on organizing computational subroutines for multidisciplinary design optimization (MDO) problems.

The method defined a matrix called the Dependency Matrix, DM, to evaluate various functions of system performance. DM is an extension of Steward's DSM with integers in the off-diagonal elements. The element  $DM(i, j)$  corresponds to the number of outputs from subroutine  $i$  which are inputs to subroutine  $j$ .

The method further used Genetic Algorithms - as an optimization tool - and incorporated it into a computer program called AGENDA – A GENetic algorithm for Decomposition of Analyses. AGENDA was applied to two types of problems: reordering, and decomposition. For reordering problems, the objective was to reduce the extent of feedback, or mathematically:

$$Objective = \sum_{i=2}^n \sum_{j=1}^{i-1} DM(i, j)(i - j) \quad (1)$$

which is explicitly the “total length of feedback” of the system.



### 2.4.5 DSM@MIT

Following an extensive DSM research in the MIT, Cho [48] introduced an integrated project management framework. The basic modules of the framework are: structuring, modeling, and scheduling. The structuring module is a DSM-based analysis of the project. Activities are sequenced to have minimum feedbacks from a structural view (by partitioning). The module further determines different iterative blocks and levels of execution. In the modeling module, dynamic iterative processes are simulated along the time line. Furthermore, resource allocation takes place in this module. Finally the scheduling module uses the outcomes from the former two modules to construct a network-based schedule in the form of a PERT or Gantt chart with scheduled activity duration.

This framework was incorporated as an Excel add-in called “DSM@ MIT”; a product development process modeling and analysis tool using advanced simulation.

### 2.4.6 Other Models

A model based upon DSM to compute the expected duration of the iterative solution process and to suggest an initial ordering of the coupled design activities to minimize the expected duration was presented in [49]. The model handles sequential iteration relationships in design and assumes deterministic activity duration with probabilistic repetition.

The work transformation matrix, which is an extension to the DSM that considers iteration watching, was presented in [50]. The model presented determines which activities may be contributing the most to the iterative development.

A DSM framework that estimates the probability of completing a product development process over time was presented in [51]. The tool can be used to compare the development time of project for different activity sequencing and overlapping degrees.

An algorithm, based on the DSM, was presented in [52] to recognize the coupled activities during the design process, to figure out the order levels of activities, and to rearrange the DSM into a lower triangular form (minimum feedbacks). But, the execution of the presented algorithm is lengthy and would consume tremendous computation time. Thus, it is not suitable for large-scale DSMs. While the presented algorithm requires several matrix manipulations to reach the solution, the reachability procedure determines coupled activities and places them in levels in the same step.

A project scheduling and rescheduling framework based on DSM for managing new product development projects was presented in [53]. In [54], a model was developed to: (1) transform the binary activity relationships in a DSM into the quantifiable activity coupling strengths; and (2) decompose the large interdependent activity group into smaller and manageable sub-groups.

## 2.5 Critique

Among the DSM analysis prototype tools presented here, DeMAID/GA was the inspirational force beyond the presented research. DeMaid/GA is characterized by its high functionality in re-ordering the activity to an optimal (or near optimal) solution, decomposing the project into several sub-circuits. Furthermore, a rather important feature of DeMaid/GA is the ability to trace expected changes in the output as a result of a change in an input or more. But it should be noticed that: (1) the GA reordering optimization takes place after partitioning, i.e. it optimizes the order of activities within each circuit rather than optimizing the order of activities with respect to the system (DSM) as a whole, and (2) no resource allocation or resource availability considerations were incorporated.

As discussed earlier in this chapter, PSM 32 is mainly the application of Steward's methodology (partitioning and tearing). The software functions do not go beyond that so PSM: (1) does not include any optimization procedure for re-sequencing, and furthermore (2) lacks an interface with a network-based scheduling module.

As DeMAID/GA, ADePT is a distinguished tool for applying the DSM methodology. ADePT is further characterized by taking construction design project as an application area (construction projects). But, the DSM analysis module in ADePT merely follows Steward's methodology.

The only application that considered optimizing the sequence of all the activities within the DSM is AGENDA. But, as in PSM32, the analysis process ends in the form of an optimized DSM and there is no means for transforming it to a program.

While the previous prototypes were stand alone applications, DSM@MIT took DSM analysis a step further by being an Excel add-in implementation; a characteristic that permits taking advantage of already developed Microsoft tools.

Although these five prototypes are distinguished tools, and can help project managers much, the following can be noticed:

1. Three of them do not optimize the sequence of project activities. They only deploy Steward's methodology to reach a better sequence with reduce number of feedback couplings.
2. The optimization (re-sequencing) of Altus et al. [47] was merely based on the total length of feedbacks and didn't not consider task time and cost.
3. DeMaid/GA favors AGENDA by optimizing the activity sequence based on project total time and cost, but it assumes a deterministic time and cost for each task.
4. None of these tools consider hard (logical) constraints, i.e. they assume that an activity order can be changed freely.

Table 2 compares the different features of previously discussed prototypes.

Table 2. Comparison of Different DSM Analysis Tools.

Feature	Tool	DeMAID/GA	PSM32	ADePT	AGENDA	DSM@MIT
Opt. Re-sequencing		✓	✗	✗	✓	✗
Multi-Objective		✓	N/A	N/A	✗	N/A
Optimization Technique		GA	N/A	N/A	GA	N/A
Time Considerations		✓	✗	In planning	✗	✗
Cost Considerations		✓	✗	N/A	✗	✗
Uncertainty Considerations		✗	✗	N/A	✗	✓ (PERT)
Resource Allocation		✗	✗	✓	✗	✓
Coupling Strength		✓	✗	✗	✗	✗
Rework Probability		✗	✗	✗	✗	✓
Commercial		✗	✓	✗	✗	✗
Tools Integrated		Heuristic-based (expert system)	N/A	Simulation	N/A	Simulation

## 2.6 Summary

Since its original introduction by Steward in the 1980's, the DSM has proved itself as an effective tool for analyzing and understanding system architecture, especially in product development. Hence, achieving improved performance. The use of the DSM is the cornerstone of the architecture proposed in the current research. This chapter provided a background on the DSM methodology, on the DSM model, and on the DSM analysis. In addition to several related DSM models found in literature, the chapter further reviewed five DSM prototype software related to the focus of the current research. These prototypes were briefly described, compared, and critiqued.

## CHAPTER III

### SIMULATED ANNEALING (SA)

#### 3.1 Optimization

Optimization is “*the act of finding the best.*” The power of optimization methods is the ability to determine the best solution without actually testing all possible solutions (i.e. without enumeration). This power comes through the use of a modest level of mathematics and clearly defined logical procedures (algorithms). Mathematical Programming is a branch of Mathematical Modeling that is concerned with finding the best possible solution to a problem in which there are a number of conflicting considerations. To apply mathematical programming techniques to any system, it is necessary to clearly define the following:

1. Systems’ boundaries. The first decision to be taken by the analyst is to determine the system boundaries; those imaginary limits that isolate, and define, the system under consideration.
2. The quantitative measure(s) of performance. There must be a quantitative performance criterion - called the objective function, which forms the basis upon which candidate solutions - or system configurations - will be compared in order to find the *best*. In some cases, when it is not possible to choose only one criterion, a multi-objective function can be formulated. In such circumstances, the search will be aimed towards finding a satisfactory solution rather than an optimum one.
3. The independent variables. These variables (called decision variables) characterize possible systems’ operating conditions. And, thus, define its output represented by the objective function value.
4. Systems’ constraints. These direct the way by which decision variables’ values are chosen.

Optimization problems can be mathematically formulated as problems requiring the minimization<sup>10</sup> of a real-valued *objective function*  $f(X)$  of an  $N$ -component *decision variables* vector  $X = (x_1, x_2, \dots, x_N)^T$  whose values are restricted, and with restrictions on the model to satisfy a number of real-valued equations (called constraints). The vector  $X$  is called a *solution* or a *configuration*. The *solution space* is the set of all solutions. The objective function has to be defined on all solutions. That is, for any solution  $X$  there exists  $f(X)$ . Thus, the general mathematical programming problem (optimization problem) can be formulated as in Eq. (2).

$$\begin{array}{llll}
 \text{Find} & X = (x_1, x_2, \dots, x_N)^T & \text{which} & \\
 & \text{minimize} & f(X) & \text{Objective Function} \\
 & \text{subject to} & h_k(X) = 0 & k = 1, 2, \dots, K \quad \text{Equality Constraints} \\
 & & g_j(X) \leq 0 & j = 1, 2, \dots, J \quad \text{Inequality Constraints} \\
 & & x_i^{(U)} \geq x_i \geq x_i^{(L)} & i = 1, 2, \dots, N
 \end{array} \quad (2)$$

This class of problems is known as *constrained* optimization problems. On the other hand, problems in which no constraints exist, i.e.  $J = K = 0$  and  $x_i^{(U)} = -x_i^{(L)} = \infty$ , are known as *unconstrained* optimization problems.

Optimization problems can be further classified into:

1. Based on equations nature,
  - a. Linear Programming Problems (LP), in the objective function and all constraints are linear functions of the decision variables.
  - b. Non-linear Programming Problems (NLP), if any of the function or the constraints is nonlinear.
  - c. Quadratic, is the objective function is quadratic.
2. Based on the decision variables nature,
  - a. Integer (IP or discrete), if some (mixed) or all (pure integer) of the decision variables are restricted to take on only integer values.
3. Based on the deterministic nature of the decision variables,
  - a. Deterministic.

---

<sup>10</sup> A maximization problem can be solved by minimizing the negative of its objective function.

- b. Stochastic, in which some or all of the decision variables are probabilistic.
4. Based on the number of objective functions,
    - a. Single objective.
    - b. Multi-objective.

For detailed discussion on optimization, the reader is referred to [55-57].

### 3.2 Meta-Heuristics

In the field of Operations Research, there are numerous numbers of combinatorial optimization problems for which finding the exact optimal solution is computationally time consuming. This class of problems is known as NP-hard. For such problems, an organized search through the solution space is required, since an unguided search will be extremely inefficient. Fortunately, decision makers in practice can accept a near-optimal solution. Hence, Metaheuristic, one of the most recent developments in approximate search methods, can play an important role. "Meta-heuristics have dramatically developed since their inception in the early 1980's. They have had widespread success in attacking a variety of practical and difficult combinatorial optimization problems[58]." This class of search (optimization) methods includes, but is not limited to Genetic Algorithms, Simulated Annealing, Tabu Search, and Ant-Colony Algorithm. Osman & Kelly [58] defined a meta-heuristic as:

"an interactive generation process which guides a subordinate heuristic by combining intelligently different concepts for exploring and exploiting the search spaces using learning strategies to structure information in order to find efficiently near-optimal solutions."

The most attractive features of meta-heuristics may be simplicity and robustness. Thus, these methods can be deployed even in cases where complex mathematical models of the problem exist.



### 3.3 Metropolis Algorithm

An algorithm that uses the Monte Carlo method to simulate the annealing process was proposed by Metropolis et al. [59]. For a given temperature  $T$ , the Metropolis algorithm samples the states of the system with the Boltzmann distribution. Given the current state,  $S$ , of the crystal solid, characterized by the position of its molecules (i.e. configuration), a small perturbation is applied by a small displacement of a randomly chosen molecule. The new, perturbed state, is accepted if either: (1) the energy difference between the current state and the new state - Eq. (3) - is negative, i.e. the new perturbed state is of a lower energy, or (2) equation (4) holds. The second acceptance rule is known as the *Metropolis Criterion*. When the perturbation is accepted, the process continues with the perturbed state replacing the old one, otherwise the old state is maintained and a new perturbation is attempted. The process stops when thermal equilibrium is reached. The Metropolis procedure is presented in Fig. 8.

$$\Delta E = E_{new\_state} - E_{current\_state} \quad (3)$$

$$e^{\left(\frac{-\Delta E}{T}\right)} \geq \theta \quad (4)$$

where

$T$  : the control parameter (temperature)

$\theta$  : a random number between 0 and 1

### 3.4 The Simulated Annealing Algorithm

#### 3.4.1 Background

Simulated Annealing (SA) is a meta-heuristic algorithm that can provide near-optimal solutions to hard combinatorial optimization problems. SA has its origin in statistical mechanics. As its name implies, SA exploits an analogy between the annealing process of solids and solving combinatorial optimization problems. The interest began with the work of Kirkpatrick et al. [60, 61], and independently Cerny [62]. Since then, SA has been applied to a large number of operations research problems, such as cell formation [63],

scheduling with resource constraints [64], scheduling with multi-level product structure [65], lot sizing [66, 67], and machine conditioning [68]. A good survey of SA application can be found in [69].

---

```

procedure Metropolis Algorithm;
   $S := S_0$ ; {initial solution}
  repeat
     $S' = \text{perturb}(S)$ ;
     $\Delta = E(S') - E(S)$ ;
     $\theta = \text{random}[0,1)$ ;
     $\text{prob} = e^{-\Delta/T}$ ;
    if  $\Delta < 0$  or  $\text{prob} \geq \theta$ 
      then  $S := S'$ 
    else retain  $S$ ;
  until stopping criterion is met;
end;

```

---

<i>S</i>	The current solution. The initial solution, $S_0$ , is a feasible solution generated either randomly or through using some heuristics.
<i>perturb</i>	A function that generates a new neighboring solution, $S' \in N(S)$ , through introducing some small perturbation to the current solution, $S$ .
<i>random</i>	A random number generator.

---

Figure 8. Metropolis Algorithm.

### 3.4.2 Physical Annealing

Annealing is a formal term for the ancient art of heating then cooling materials to forge pottery, tools, weapons, and works of art. The process consists of the following phases: (1) melting the solid material by increasing its temperature to a maximum value at which all molecules of the molten material randomly arrange themselves in a liquid phase; (2)

the liquid material is then cooled according to a precise cooling schedule: the temperature descends slowly through a series of intermediate temperatures, and at each temperature, the molten material is kept long enough to reach thermal equilibrium (meta-stable condition); and (3) the cooling process continues until the desired solid phase, the perfect lattice structure, is achieved. The material now is said to reach a frozen (a low energy ground) state. Rapid cooling, i.e. reaching the lowest ground state without allowing the liquid material to have a thermal equilibrium at the intermediate temperature values, can result in widespread imperfections within the crystal structure of the material. This process is known as '*quenching*.'

### 3.4.3 The Algorithm

Simulated annealing is a stochastic optimization technique. It constructs a sequence of solution configurations (a walk or path) through the set of permissible solutions called the state space. Based on the current solution and a certain acceptance criterion, a transition mechanism determines which solution to step up to next. The optimal solution steps from the current configuration to another configuration from its neighborhood according to the Metropolis criterion. The simulated annealing algorithm is presented in Fig. 9.

The basic structure for SA implementation consists of the following basic elements:

1. A representation of possible solution configurations (search space).
2. A generation mechanism<sup>11</sup>.
3. A means of evaluating the problem objective function (energy).
4. A cooling (annealing) schedule.

Figure 10 illustrates the flow chart of a standard SA algorithm.

---

<sup>11</sup> A generation mechanism is a means of selecting a new solution from the neighborhood of the current solution.

---

```

procedure Simulated Annealing;
  {Naive Simulated Annealing Algorithm}
   $S := S_0$ ; {initial solution}
   $T := T_0$ ; {initial temperature}
  repeat
    repeat
       $S' = \text{perturb}(S)$ ;
       $\Delta = E(S') - E(S)$ ;
       $\theta = \text{random}[0,1)$ ;
       $\text{prob} = e^{-\Delta/T}$ ;
      if  $\Delta < 0$  or  $\text{prob} \geq \theta$ 
        then  $S := S'$ 
        else retain  $S$ ;
      until inner loop stopping criterion is met;
       $T = \text{update}(T)$ ;
    until outer loop stopping criterion is met;
  end;

```

---

<i>T</i>	The control parameter.
<i>update</i>	Cooling schedule function.

---

Figure 9. Generic Simulated Annealing Algorithm.

### A. Generation Mechanism (Neighborhood Moves)

The standard implementation of the SA algorithm is one in which homogeneous Markov chains of finite length are generated at decreasing temperatures. In SA context, a homogeneous Markov chain is a series of random changes in the control variables. The SA algorithm consists of a sequence of iterations. At each iteration, the current solution is randomly perturbed to create a new solution in its neighborhood. Thus, the way in which new solutions are generated plays a very important role in the SA algorithm. The solution generating technique should (1) introduce small random changes in such a way that the generated solution is feasible, and (2) allow all possible solutions in the neighborhood to be examined. For the scope of this dissertation, the discussion of solution generation techniques shall not be extended. However, for problems with continuous control variables, suggestions can be found in [70, 71]. On the other hand, for combinatorial optimization problems, the solution representation and generation mechanism(s) are

necessarily problem-specific. It is common for the move set to permute a small, randomly chosen, part of the solution. For example, a move set has been suggested by Lin [72] for the traveling salesman problem.

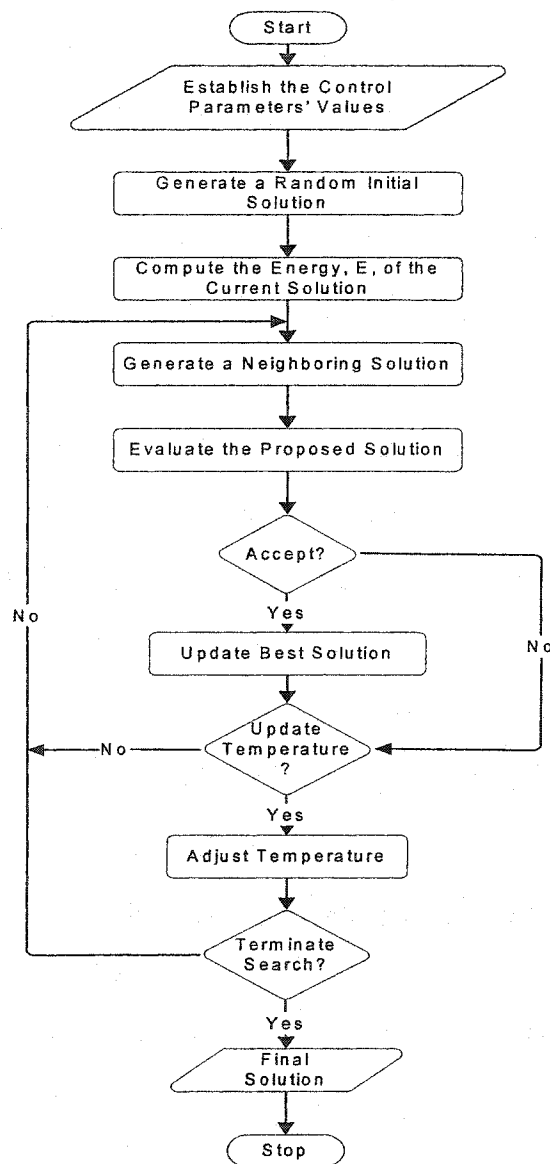


Figure 10. Flowchart of the Standard (Naïve) SA Algorithm.

## B. Objective Function Evaluation

One important characteristic of the SA algorithm is that it does not require or deduce derivative information. It merely needs to be supplied with an objective function value for each trial solution it generates. Thus, the evaluation of the problem functions is essentially a *'black box'* operation as far as the optimization algorithm is concerned (See Fig. 11). On the other hand, it is so important that the objective function evaluations should be performed efficiently for the sake of the overall computational efficiency, especially in many applications where these functions are complex and can overwhelm the most computationally intensive activity.

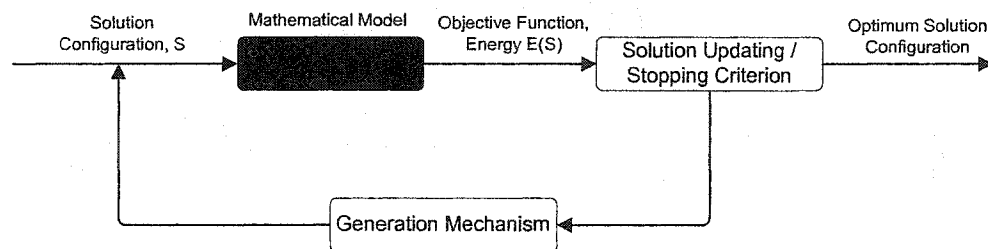


Figure 11. Black Box Optimization.

## C. Cooling (Annealing) Schedule

The objective of the cooling schedule is to achieve a finite-time implementation of the SA algorithm. It determines the degree of uphill or downhill movement permitted during the search and is, thus, critical to the algorithm's performance. But, *"choosing an annealing schedule for practical purposes is still something of a black art."* Bonds [73] In designing the cooling schedule, four rules have to be specified:

1. An initial temperature,  $T_0$ ,
2. A rule for decrementing the temperature,

3. A final temperature,  $T_f$ , or stopping criterion, and
4. A length for the Markov chains.

While the first three rules manage a finite sequence of the control parameter (the temperature), the fourth rule manages a finite sequence of transitions at each value of the control parameter.

### Initial Temperature

The selection of the initial temperature  $T_0$  is critical. On one hand, the value of  $T_0$  should be high enough to allow all, or most, transitions to be accepted. This, of course, would result in a lot of consumed time in the beginning of the process without progress towards the optimal solution. But, on the other hand, a low initial temperature would reduce the quality of the final solution.

Kirkpatrick [74] suggested that a suitable initial temperature  $T_0$  is one that results in an initial acceptance ratio;  $x(T_0)$ , of about 0.8 (In other words, there is an 80 percent chance that a change which increases the objective function will be accepted). Equation (5) determines the acceptance ration at any level  $k$ .

$$x(T_k) = [(number\ of\ accepted\ transitions)/(number\ of\ proposed\ transitions)]_k \quad (5)$$

Since the value of  $T_0$  depends on the scaling of the objective function,  $E$ , and, hence, must be problem-specific, it can be estimated by conducting an initial search in which all increases are accepted, and calculating the average objective function increase observed. An approximation of  $T_0$  is then given by Eq. (6).

$$T_0 = \frac{-\overline{\Delta E}^{(+)}}{\ln(x(T_0))} \quad (6)$$

### Decrementing the Temperature

The simplest and most common temperature decrement rule is given by Eq. (7). This *exponential cooling scheme* (ECS) was first proposed with  $\alpha = 0.95$  [60]. In Randelman & Grest [75], this strategy was compared with a *linear cooling scheme* (LCS) in which  $T$  is reduced every  $L$  trials according to Eq. (8). The results suggested that reductions achieved using the two schemes were comparable and also noted that the final value of the objective function was, in general, improved with slower cooling rates, at the expense, of course, of greater computational effort. Finally, it was observed that the algorithm performance depends more on the cooling rate ( $\Delta T/L$ ) than on the individual values of  $\Delta T$  and  $L$ . Obviously, care must be taken to avoid negative temperatures when using the LCS.

$$T_{k+1} = \alpha T_k, \quad k = 1, 2, \dots, \quad (7)$$

where

$\alpha$  is a constant close to, but smaller than, one.

$$T_{k+1} = T_k - \Delta T \quad (8)$$

Many researchers have proposed more elaborate annealing schedules, most of which are in some respect adaptive, using statistical measures of the algorithm's current performance to modify its control parameter. These are well reviewed by Van Laarhoven & Aarts [76].

### Final Temperature

For simple implementations of the SA algorithm, the final temperature can be determined by fixing either the number of temperature values to be used or the total number of solutions to be generated. Alternatively, the search can be stopped when it ceases to make progress. One of the methods used to define lack of progress is when no improvement (i.e. no new best solution) is found in an entire Markov chain at one temperature.



### Length of Markov Chains

The length of the  $k$ -th Markov chain,  $L_k$ , is based on the number of transitions needed to achieve a quasi-equilibrium at each value  $T_k$ .  $L_k$  depends on the size and nature of the problem, and is independent of  $k$  (homogeneous Markov chains). In practice, the Markov chain is usually bounded by either 1- some constant  $\bar{L}$  transitions, or 2- a minimum number of accepted transitions  $n_{\min}$ , whichever comes first.

### **3.5 SA: Pros and Cons**

Simulated Annealing has been widely used for tackling different combinatorial optimization problems [77]. Depending on the problem to which it is applied, SA appears competitive with many of the best heuristics, as shown in the work of Johnson & McGeoch [78].

As any other optimization technique, SA has its own advantages and disadvantages. Among its advantages are: (1) relative ease of implementation, (2) its wide range of applications, (3) the ability to provide reasonably good solutions for most problems, (4) can be combined with other techniques, and (5) its robustness [77]. Moreover, SA statistically guarantees finding an optimal solution [79]. However, the standard SA has its critics. Some of the drawbacks are: (1) being time consuming, (2) difficult to fine tune to specific problems, and (3) being short on mathematical rigor [79, 80].

#### **3.5.1 Simulated Annealing vs. Local Search**

Classical neighborhood (or local search) methods form a general class of approximate heuristics based on the concept of exploring the neighborhood of the current solution. Neighboring solutions are generated via a specified generation mechanism, and the algorithm accepts only those neighborhood moves that lead to incremental improvement of the objective function. As shown in Fig. 12, for a minimization problem, only moves

that decrease the objective function value (i.e. moving downhill) are accepted. Thus, the inherent problems with this class of algorithms are: (1) they can be easily trapped in local optima, and (2) they depend entirely of the initial solution. On the other hand, by allowing perturbations to move to a worse solution with according to a controlled mechanism, SA, as shown in Fig.13, is able to avoid local optima and potentially finds a more promising downhill path. Although finding the global optima with SA is not fully guaranteed, SA provides a near-optimal solution. Furthermore, these accepted uphill moves provide solutions independent of the initial solution.

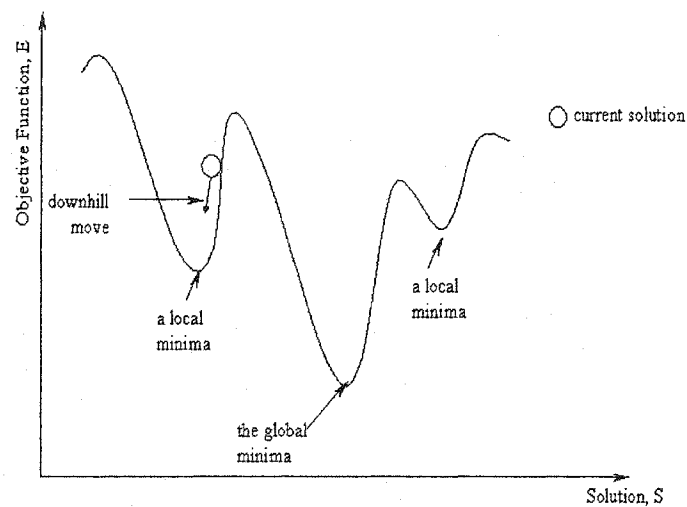


Figure 12. Local Search.

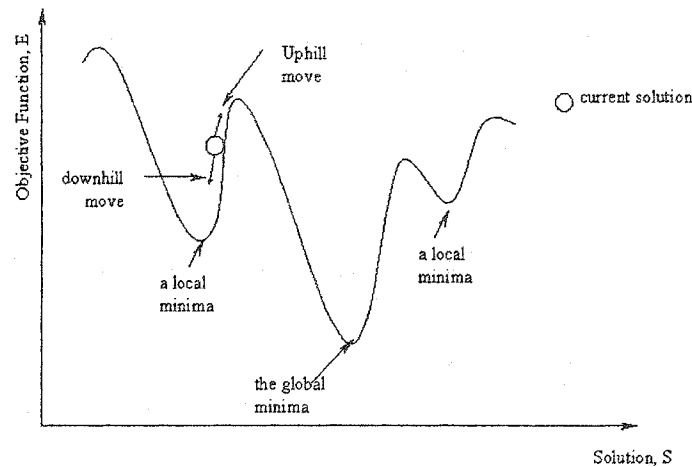


Figure 13. Simulated Annealing.

### 3.6 Summary

There are numerous numbers of combinatorial optimization problems for which finding the exact optimal solution is time consuming. For such problems, finding a near-optimal solution is satisfactory. The current research adopts a meta-heuristic algorithm – called Simulated Annealing (SA) - that can be used to tackle such problems. This chapter provided fundamental description of simulated annealing. Its algorithmic steps were explained and insights into the optimization process were given. A brief description of SA optimization process compared to classical local search algorithms was presented. The ability of the SA algorithm to avoid being trapped in local optima, in addition to its robustness, provided the justification for using it as the optimization tool in the presented architecture.

## CHAPTER IV

### SIMULATION-BASED OPTIMIZATION

#### 4.1 Simulation

sim·u·la·tion (*plural sim·u·la·tions*) *noun*<sup>12</sup>

1. reproduction of features of something: the reproduction of the essential features of something, for example, as an aid to study or training
2. false appearance: the imitation or feigning of something
3. fake: an artificial or imitation object
4. Computing Statistics: the construction of a mathematical model to reproduce the characteristics of a phenomenon, system, or process, often using a computer, in order to infer information or solve problems

For a long time, simulation has served as an important decision-support tool in a wide variety of disciplines. Simulation was defined in [82] as:

“ a numerical technique for conducting experiments on a digital computer, which involves types of mathematical and logical models that describe the behavior of business or economic system (or some component thereof) over extended periods of real time.”

This definition contains several important terms that define the main characteristics of simulation:

1. Simulation , of our interest, is *numerical*.
2. It requires extensive calculation time, so a *computer* is needed.
3. It requires some kind of *mathematical* and/or *logical modeling*.
4. It can be applied to a broad variety of disciplines.

Simulation provides an alternative to analytical solution procedures [83]. Generally, the process of simulation involves modeling the system of interest in an appropriate form and then executing this model to obtain operational information. Figure 14 presents the flow chart of a typical simulation process. Computer simulation models are classified into several categories. The study here considers stochastic simulations, in which different components and variables of the model are subject to uncertainty factors. For further details, refer to Law & Kelton [84].

---

<sup>12</sup> [81]

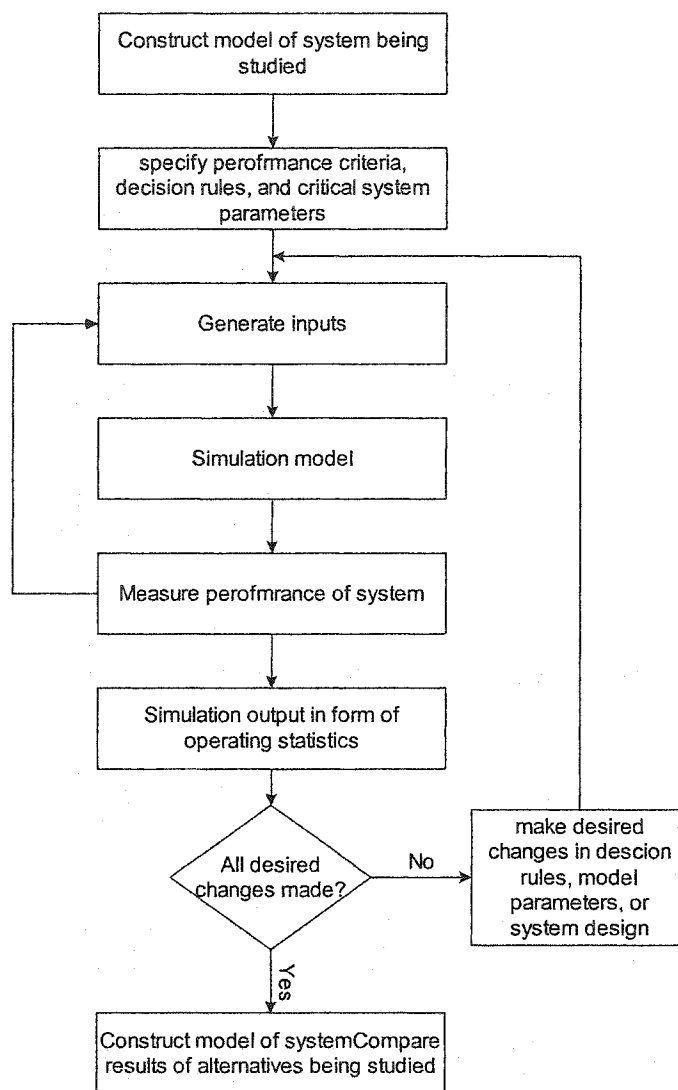


Figure 14. System Experimentation with Simulation. (Reference: [83]).

## 4.2 Advantages and Disadvantages

Although simulation has been, and is sometimes still viewed as analyst 'last choice' to be employed when all other optimization techniques are inapplicable, recent advances in simulation methodologies, software development, stochastic optimization, and the great breakthrough in computing capabilities would make simulation "*one of the most widely*

*accepted and practiced tools in systems analysis and operations research [85]."*

Motivations for using simulation includes:

1. The increase in complexity of large-scale systems, which makes the formulation of a mathematical model, and hence solving it, a difficult task.
2. In some cases, the degree of simplification needed would seriously affect quality of the obtained solution.
3. The relative simplicity of simulation models leads to a better understanding of the real system.

But, on the other hand, simulation has its own associated limitations, some are:

1. The output provided is an approximate solution rather than an exact one.
2. For large-scale simulations, the development of the model can be both time and effort consuming.

For further discussion refer to [85-87].

### **4.3 Uncertainty Analysis**

Risk is often viewed as the probability of an undesired or harmful event. The connection to probability is implied by the uncertainty in the occurrence of the event. Uncertainty is defined, in a statistical or probabilistic context, as "*the implication that uncertainty exists when the probability of an event occurring is not zero or one* [88]." Uncertainty analysis is the part of risk assessment that focuses on the uncertainties in the assessment. Uncertainty analysis includes both a qualitative component in which uncertainties are identified and quantitative component of the effects of these uncertainties [89].

From a modeling view point, uncertainty is categorized into: structural (refers to uncertainty due to lack of knowledge about the correct model); parameter (associated with the uncertainty introduced by having to use values of model parameters that are not surely known); and stochasticity (occurs when parameters or other quantities are not fixed but may vary [89]). In quantitative analysis, measuring uncertainty is based on some statistical measures of the distribution describing it. The most common measure is the *variance*, which describes how an estimated parameter would vary in repeated sampling.

#### 4.4 Monte Carlo Simulation

The theoretical basis of the Monte Carlo (MC) method has long been known, but it traces its modern origin and name to the work of von Neumann and Ulam in the late 1940's when they coined the term during the Manhattan Project of World War II in their article entitled "The Monte Carlo Method<sup>13</sup>." The method gets its name as a result of the similarity of probabilistic simulation to games of chance and gambling, and because of the famous Mediterranean resort associated with these games (the capital of Monaco). Sobol [91] stated the general objective of MC by the following definition:

"The Monte Carlo method is a method of approximately solving mathematical and physical problems by simulation of random quantities."

In contrast to conventional numerical discretization methods, which typically applied to ordinary or partial differential equations that describe the system of interest, the application of MC requires only that the system be described by probability density functions. Monte Carlo simulation then proceeds by random sampling from these pdf's (using some random number generators) to generate an artificial history data. The random numbers generated are further used in calculations to duplicate the expected system outputs. The method is relatively simple in concept.

Figure 15 illustrates the idea of MC simulation. The process goes as follows:

1. The system of interest is being modeled, this includes defining input variables, and output measures. Then, a probability density function is assigned to each input variable.
2. MC generates random numbers uniformly distributed on the interval [0,1].
3. These random numbers are then transformed by the pdf's to generate stochastic input values.
4. The values are fed to the model to determine the output measure.
5. Finally, the outputs are accumulated to produce the desired results.

---

<sup>13</sup> Reference [90]

It is worth mentioning here that the wide spread use of MC is linked directly to the breakthrough of computational capabilities of computers. Despite of the simplicity of its computational algorithm structure, MC was not widely applicable prior to the appearance of computers, since the simulation of random variables by hand is a very exhaustive process. Furthermore, one main feature of MC method is that: since the error expected from calculations can be defined by Eq. (9) where  $D$  is some constant and  $N$  is the number of trials. A sufficiently large number of trials is required in order to attain high precision in MC,.

$$\text{error} = \sqrt{D/N} \quad [91] \quad (9)$$

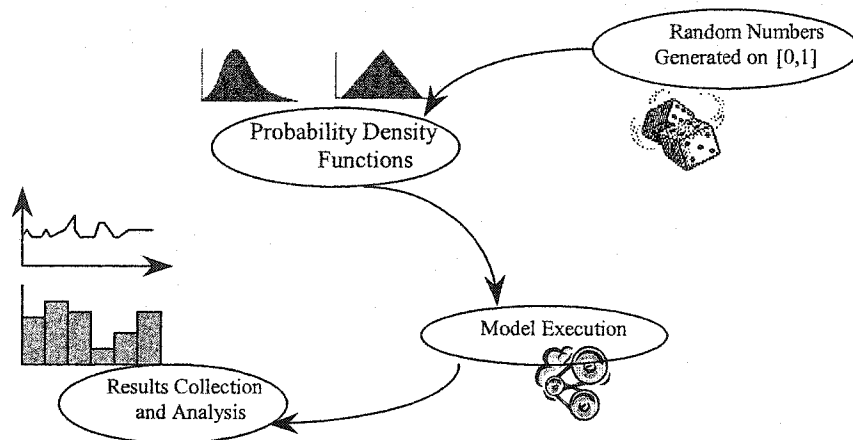


Figure 15. Monte Carlo Simulation.



#### 4.5 Interfacing Optimization Methods with Simulation

Suri [92] divided the field of models into two main categories: (1) *generative* techniques such as linear programming, and (2) *evaluative* techniques such as simulation. A generative technique will show the optimum solution when given the input of parameters and constraints. On the other hand, simulation, which is an evaluative technique, is quite different. It only shows the outcome of an operation given that certain variables are put into the model. In this case, the model is handled as a black box. While the use of generative techniques might require certain (sometimes unrealistic) assumptions, simulation models can incorporate a greater level of details and capture specific features of the real system, such as time dynamics and overall behavior. “But using simulation models only for descriptive purposes does not alone justify the effort to build them [93].” In real life application, a typical decision-maker would like to have answers to many “what-if ” questions, i.e. what will be the effect on the model output when some parameters are changed. The goal here is either to fine-tune the system in reality, or to take further chances (risks, or uncertainty) into consideration. Thus, the question arises: instead of using rudimentary optimization techniques, like trial and error, can a more effective optimization method be implemented to guide the simulation process? And, if the answer is yes, which method or family of methods is best suitable for this purpose?

Why do we need to implement an efficient optimization method rather than simply trial and error? The answer is that simulation optimization models has the following unpleasant features [94-97]:

- Model behavior is very complex – a result of its high non-linearity.
- Noisy model output – simulation models are stochastic in nature, thus their output is not deterministic with respect to the model parameters.
- The parameter space is not continuous – often there is a need for discrete parameters such as integer, logical or linguistic.
- The search space is relatively large.

Performance measures could have many extrema or there could be multiple global solutions with the same value.

## 4.6 Related Work

Recently, there has been considerable research devoted to finding methods to optimize a simulation [84]. As will be discussed later in this chapter, these methods generally involve guiding a sequence of simulation runs by supplying the simulation model with a set of system configurations, with the results from simulating earlier configurations being used to suggest a new promising direction in the search space. Sufficient background on this issue is provided in [98-100].

For many years, using optimization methods to guide a sequence of simulations or simply ‘*Simulation Optimization*’ has challenged researchers. Simulation is an expensive tool (in terms of both time and cost), thus, optimization has to be achieved with as minimal number of runs as possible. In literature, simulation optimization using response surface methods and finite differences approximation of the gradient have been reported, but still, “the number of computer runs needed for these method can be very large [101].” By reviewing the bulk of literature related to simulation optimization, the optimization methods used in simulation can be categorized as follows:

1. Classical methods,
  - a. Gradient search methods.
  - b. Pattern Search techniques.
  - c. Deterministic search techniques.
2. Stochastic Approximations.
3. Artificial Intelligent,
  - a. Evolutionary Techniques.
  - b. Meta-heuristics.

Table 3 provides a summary of publication in each of the previous categories. Additional surveys are found in [94, 95, 102]. The table suggests that most of the research was devoted to using classical methods and stochastic approximation methods. But, recognizing the limitation of these methods, researchers started to investigate different techniques, which can achieve preferable results with less time. These methods (which align the focus of our research here) are known as meta-heuristic techniques. Compared

to other methods, it seems that the attention of using meta-heuristics in simulation optimization started relatively late.

In [103], genetic search was compared to two other algorithms; the pattern search (using the Hooke-Jeeves algorithm) and the response surface method search. The comparison of these three algorithms was based on accuracy (how close the algorithm comes to the optimum) and stability (lower variability). The authors tested the three methods on an example problem common in simulation with its optimum determined by exhaustive search. The results showed that genetic algorithm executes a superior search compared to pattern and response surface search, taking into consideration that the speed of the search was not a critical factor in the evaluation the algorithms. An attempt to apply genetic algorithms (GAs) to the problem of optimizing an existing simulation model was done by [93]; where a simple real-coded GA was presented and used to change the simulation model parameters. Azzaro-Pantel et al. [104] presented a two-stage methodology for solving industrial-size scheduling. The first step involved the development of a discrete-event simulation (DES) model and the second step used GA for optimization. In a following research, Azadivar & Tompkins [105] developed a methodology that allows qualitative variables to be optimized in a manufacturing system using simulation-optimization. The proposed methodology used a GA coupled with an automatic object-oriented simulation-model generator.

One of the most famous software that employs meta-heuristics in simulation optimization is 'OptQuest' (developed by OptTec Systems, Inc.). This software uses Tabu and Scatter search methods linked to a famous risk analysis tool 'Crystal Ball' (developed by Decision Engineering, Inc.). The software, described in [106], effectively integrates Crystal Ball simulation and optimization. The ability of the system to find optimal and near optimal solutions in minutes for applications where an exhaustive examination of relevant alternatives requires days or months was also demonstrated. In Laguna [107], description and comparison of the functionality of three general-purpose optimizers that implement meta-heuristic algorithms were provided. These optimizers are: (1) Evolver: a commercial genetic-algorithm software that in its most simple form operates as an add-in

function to Microsoft Excel; (2) Genocop: an experimental genetic algorithm implementation; and (3) OptQuest: based on the scatter search methodology, has been commercially implemented to add optimization capabilities to simulation software.

Regarding Simulated Annealing (SA), Bulgak & Sanders [108] integrated an extension of the simulated annealing algorithm with a discrete event simulation of the manufacturing system to find optimal buffer sizes for asynchronous assembly systems which involve automated inspection as well as automated assembly. The basic idea in this modified version of SA is to make the comparisons based on whether or not the values of the objective function indicate statistically significant (based the confidence intervals set for these values) differences at each iteration. A theoretical analysis for the SA algorithm when the objective function includes noise was presented in [109]. SA was further applied to a stochastic optimization problem in [110]. This approach, however, made it necessary to store all feasible solutions encountered during the execution of the algorithm and to compare them with each newly generated solution. Thus, this approach can be considered not realistic for practical applications since a high computational burden is involved.

Haddock & Mittenthal [111] attempted to investigate the feasibility of using a simulated annealing algorithm in conjunction with a simulation model to optimize a non-convex, non-concave objective function of the input parameters. Multiple runs of the simulated annealing algorithm were used to find an optimal or near-optimal solution to the problem. The experiment considered a relatively small number of decision variables. The authors treated the point estimate coming from the simulation output as a deterministic value and used it in the simulated annealing algorithm to obtain the optimal solution point. When comparing two solution points, one cannot draw a conclusion and make a decision based on point estimate analysis only, even when steady states behavior have been reached, without running a large number of simulations [112]. Simulated annealing was applied to a flow shop scheduling problem with stochastic processing time in [113]. A less accurate estimate based on fewer simulations was used. Thus, in order to overcome this low accuracy, strategies were developed for taking this into account in the

rejection/acceptance criteria. The paper described four variants of the basic SA algorithm representing these strategies and presented computational experience based on the use of these methods in the solution of stochastic flow shop scheduling problem. The classical convergence result for the SA algorithm to the case where cost function observations are disturbed by random noise was generalized in [114].

In Ahmed et al. [115], a simulation-optimization integrated approach to determine the design parameters of stochastically constrained systems was presented. A simulated annealing algorithm with modified rejection/acceptance criterion (that takes into consideration the stochastic nature of the system) was used to solve the optimization model (discrete integer). In Alkhamis et al. [112], the basic convergence results for the Simulated Annealing (SA) algorithm was extended to a stochastic optimization problem where the objective function is stochastic and can be evaluated only through Monte Carlo simulation (hence, disturbed with random error). Ahmed & Alkhamis [116] Presented a new iterative method that combines the simulated annealing method and the ranking and selection procedures for solving discrete stochastic optimization problems. Unlike the original SA, the presented procedure is guaranteed to converge almost surely to a global optimal solution (The original SA method is only guaranteed to converge in probability). It should be noted that most methods for discrete case mentioned so far have been applied and developed for unconstrained optimization problems [115].

Table 3. Summary of various simulation-based optimization research.

Category	Methods	References*
Gradient search Methods		98, 117 - 122
Pattern Search Techniques	Conjugate direction search--Coordinate search--Hooke and Jeeves--Parallel tangent search--Simplex-based techniques--Steepest Ascent (descent))	123 - 125
Deterministic Search Techniques	Response surface method-- Simple search techniques	126 - 132
Stochastic approximation	Kiefer-Wolfowitz Type techniques--Robbins-Monro Type Techniques)	94, 95, 101, 133 - 143
MetaHeuristics	Simulated Annealing	111, 112, 115, 116
	Genetic Algorithms	93, 104, 105
	Tabu Search Scatter Search	106, 107
Evolutionary Techniques		145

\* The references in the table are representative of the type of solution; this table does not contain an exhaustive list of published works

#### 4.7 Summary

Although uncertainty in the estimation of activity durations (and cost) cannot be eliminated, its effect can be reduced by incorporating it in the model. One method for handling uncertainty is the use of Monte Carlo (MC) simulation. This chapter briefly presented MC basic operation. Since the presented architecture in this research involves a simulation-based optimization framework, the chapter further reviewed work related to interfacing optimization methods with simulation.

## CHAPTER V

### TIME-COST TRADE-OFF IN PROJECT NETWORKS

#### 5.1 Project Management

A project can be defined as *“a group of tasks performed in a definable time period in order to meet a specific set of objectives [145].”* Or, as defined in [146], *“an endeavor to accomplish a specific objective through a unique set of interrelated tasks and the effective utilization of resources.”* Generally, a project exhibits most of the following conditions:

1. It is likely to be a unique, one-time program.
2. It has a well-defined objective stated in terms of scope, schedule, and cost.
3. It has a specific time frame, a life cycle or a finite life span. In other words, a project must have a start time and a date by which the objective must be accomplished.
4. A project is carried out through a series of independent tasks - that is, a number of non-repetitive tasks that need to be accomplished in a certain sequence in order to achieve the project objective.
5. A project utilizes various resources to carry out the tasks.
6. It has a budget.
7. A project involves a degree of uncertainty.

The management of a project is quite different from the management of a continuing operation. The generally accepted definition of management is

*“the planning, organizing, directing, and controlling of company resources to meet the company’s financial and non-financial objectives.”*

Project management, on the other hand, can be defined as:

1. *“the application of knowledge, skills, tools, and techniques to project activities in order to meet or exceed stakeholder needs and expectations from a project [147],”*
2. *“the planning, organizing, directing, and controlling of resources for a specific time period to meet a specific set of one-time objectives [148],”* Or

3. “the process of managing, allocating, and timing resources in order to achieve a given objective in an expedient manner [145].”

From these definitions, two major differences appear between the two kinds of management: (1) in project management the manager is not directly responsible for staffing and must use and direct resources from other components or companies, and (2) project management concerns about “specific time period” to meet “one-time objectives”.

Project management involves a process of first establishing a plan and then implementing that plan to accomplish the project objective. Once the project starts, the project management process involves monitoring progress to ensure that everything is going according to plan. The ultimate benefit of implementing project management techniques is having a satisfied customer. Completing the full scope of work of the project in a quality manner, on time, and within budget provides a great feeling of satisfaction. Thus, meeting the project objective(s) involves compromising competing demands on: scope, time, cost, and quality.

There are three project management techniques that are commonly used:

1. Critical Path Method (CPM). A mathematical model that calculates the total duration of a project based on individual task durations and dependencies, and identifies which tasks are critical. This model is the fundamental scheduling method used in project management software today.
2. Program Evaluation Review Technique (PERT). Uses statistical probabilities to calculate expected durations.
3. Gantt chart. A way to graphically represent activities across a time scale.

## **5.2 Project Management in Manufacturing**

*“Manufacturing is the act of making something through deliberate processing from raw material to the desired object, usually with the use of machinery [149].”* This act encompasses several functions that must be strategically planned, organized, scheduled,



controlled, and terminated. A manufacturing cycle includes, but is not limited to, such functions as forecasting, decision analysis, cost analysis, inventory control, process planning, machine scheduling, quality control, production planning, process control, work and time analysis, and a host of other functions. These are all functions that fall within the planning, organizing, scheduling, and control functions of project management.

Through the years, the Critical Path Method (or CPM) has been used for many applications, such as effective project planning, identification of bottlenecks; communications improvement; and resource allocation. And despite the fact that the widespread use of CPM was mainly achieved by, and for, construction applications, CPM is equally suitable for planning any one-time project involved in the manufacturing such as setting up a new department, new product innovation, research and development projects, and most importantly the manufacturing of a large and complex product (an aircraft for example).

Project management is characterized by qualified collaborators and by suitable planning and controlling methods. The strong point of the management concept for innovation projects lies in the formulation of the activity and in the clear representation of the project situation. Heuer [150] discussed the applications of project management in mechanical engineering, planning and controlling of industrial intentions. The failure of some newly installed manufacturing systems to live up to their pre-installation expectations has been blamed on a number of factors. One overriding factor is poor project planning. Brown [151] applied Project management to the design and supply of a power station's mechanical and electrical plant. Feldermann [152] described the setup and the concept of an effective project planning and control in the manufacturing area. There is a dramatic rise in the use of project management as organization shift to provide customer-driven results and systems solutions. Actions that upper managers can take to create an environment for more successful projects in their organizations were reviewed in [153].

New product development best practice models advocate the integration of teamwork; simultaneous engineering tools and techniques; and process and project management. The design of models is relatively straightforward compared to the implementation of these models which is significantly more difficult. Boznak [154] discussed the role that project management could play in employing company strategies to reduce new product development time. In Churchill [155], the principles behind quality assurance as an effective strategy for management of large-scale capital projects were discussed. It is of vital importance for the manufacturing industry to respond to the requirements of the market in a flexible, cost favorable and above all quick way. In Beghini & Romanin [156], an integration among the solutions of the problems concerning project planning, material purchasing and information exchange in a firm working by orders was studied. A new discipline that faces the project evolution starting from the feasibility study up to the production delivery was developed. An interpretative model that explains firms' dynamic behavior in multi-project management of new product development was proposed in [157]. The model could be used as a unique and homogeneous framework that supports the processes of project selection, resource allocation, risk management, priority management and ongoing control.

In [158], the conception of an integrated product and process model was introduced which is particularly suitable for areas of project management, design, and assembly planning. A novel approach supporting administrative tasks within the lifecycle of design projects was presented in [159]. The approach was based upon comprehensive models of design environments and design activities and combined known techniques from project management and mechanisms for design flow control.

### **5.3 Project Crashing and Time-Cost Trade-Off (TCTO)**

There are three main points that are most important to a successful project: (1) a project must meet the customer requirements, (2) it has to be within budget, and (3) it has to be on time. Furthermore, project managers in addition to scheduling projects, are frequently confronted with the problem of having to reduce the scheduled completion time

(indicated by the CPM or PERT network analysis) to meet a pre-specified deadline. Project duration reduction (or project crashing) can be achieved by assigning more resources (labor, material, equipment, etc.) to some critical activities of the project. Taylor III [160] defined project crashing as:

“a method for shortening the project duration by reducing the time of one or more of the critical project activities to a time that is less than the normal activity time.”

However, additional resources cost money, and, hence, increase the overall project cost. Thus, the decision to reduce the project duration, and by how much, must be based on an analysis of the trade-off between desired project duration and the extra cost needed.

There are, basically, three methods for crashing a project: (1) to re-plan the project using different methods, (2) to re-plan the sequence of activities so that activities that formerly were in series are now done in parallel, or (3) to apply additional resources (manpower, equipment, money) to the project to speed it up, and this, of course, may include outsourcing.

The main assumptions underlying most crashing practices include the following: (1) jobs can be done more quickly if more resources (men, machinery, and/ or materials) are allocated to them, (2) these resources can be measured and estimated, reduced to monetary units, and summarized as a direct cost per unit time, and (3) unlimited resources are available.

The importance of the time-cost trade-off problem arises from the wide range of its application involvement. In Batson [161], an implementation of a time-cost trade-off algorithm in aircraft technology development projects was discussed. Graves [162] presented a brief review of the key concept of a convex time-cost trade-off, which by assumption forms the basis for both static and dynamic models of research and development investment expenditure. In Haffner & Graves [163], the time-cost trade-off was used to maintain the planned market entry of a product.

## 5.4 Time-Cost Relationship Models

Among different CPM advantages is the ability of its basic calculations to be extended to incorporate cost explicitly, thus integrating the planning and control aspects of project management with the financial and budgeting activities. This is done by defining some specific cost model representing the activity time-cost relationship. Such models can be used to determine: (1) the cost of speeding up (accelerating, or 'crashing') a project to meet a specified dead line, and (2) the most economical (optimum) schedule for meeting a specified completion date. This relationship can be represented by means of a graph of cost versus duration, as shown in Fig.16-a. Point A is the result of using the cheapest (and usually the slowest) method of completing the activity. This is called the 'normal point'. The cost of completing the activity is then called the 'normal cost', and the associated completion time is called the 'normal duration'. As the activity speeds up or 'crashes', its cost goes up, as shown by the line A-B. Finally, point B is reached, which is the shortest possible completion time (duration) for this activity. This is the 'crash point'. The cost is then called the 'crash cost', and the associated completion time is called the 'crash duration'. Additional manpower or other resources would increase costs but would not shorten the job. A-B is called the time-cost curve. In Fig. 16-a, this curve is shown as a straight line. Actually, it could have any shape depending on what type of resource associated with the activity in question. In most cases, however, a straight line drawn between the crash and the normal point, as shown in Fig.16-b, can approximate the curve. As discussed in Chapman [164], the activity time-cost relationship may take one of several shapes. It may be concave, as in Fig. 17-a, or piecewise linear approximation to more general functions as in Fig.17-b. In some exceptional cases, the curve can be shown by a series of straight lines as in Fig.17-c. Furthermore, there are cases where the relationship between time and cost does not result in a continuous curve. This would occur when there are only two or more distinct ways of accomplishing the operation, and no 'in-between' possibilities as shown in Fig.17-d.

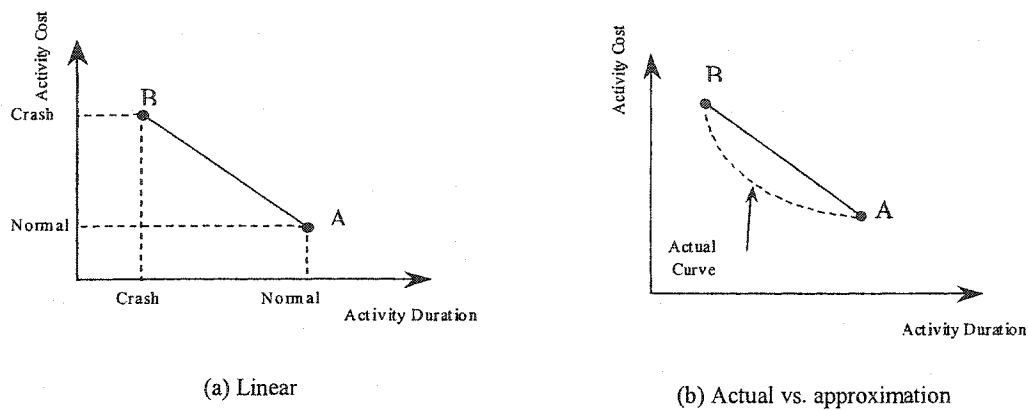


Figure 16. Linear Time-Cost relationship.

The formulation of the time-cost trade-off (TCTO) problem in project management networks has been handled in several ways. Kelly [165] presented a linear programming approach for project crashing assuming that cost varies linearly with activity completion time. Related work is also presented in Siemens [166] and Goyal [167]. Several researchers [168-171] have developed models and solution procedures to incorporate a non-linear relationship between cost and activity completion time.

A solution methodology for project crashing problems with convex or concave activity duration functions was developed in [172]. The proposed procedure actually approximates these relationships by piece-wise linear time-cost curves. In Babu & Suresh [173], the concept of time cost model was extended to include the project quality that was assumed to be affected by project crashing. An optimization model was developed to consider the time-cost-quality tradeoffs in project management simultaneously. In Pulat & Horn [174], a multiple objective linear programming model was presented. The TCTO technique is extended to solve the time-resource trade-off problem with two resources or two groups of resources. In Demeulemeester [175], a description was given of a new exact procedure for the discrete TCTO problem in deterministic activity-on-the-arc

networks where the duration of each activity was a discrete, non-increasing function of the amount of a single resource (money) committed to it. The objective was to construct the complete and efficient time-cost profile over the set of feasible project durations. In Abdelsalam & Bao [167], a modified TCTO model was being presented and implemented. The presented model extends the classical crashing model to include more than one crashing scenario.

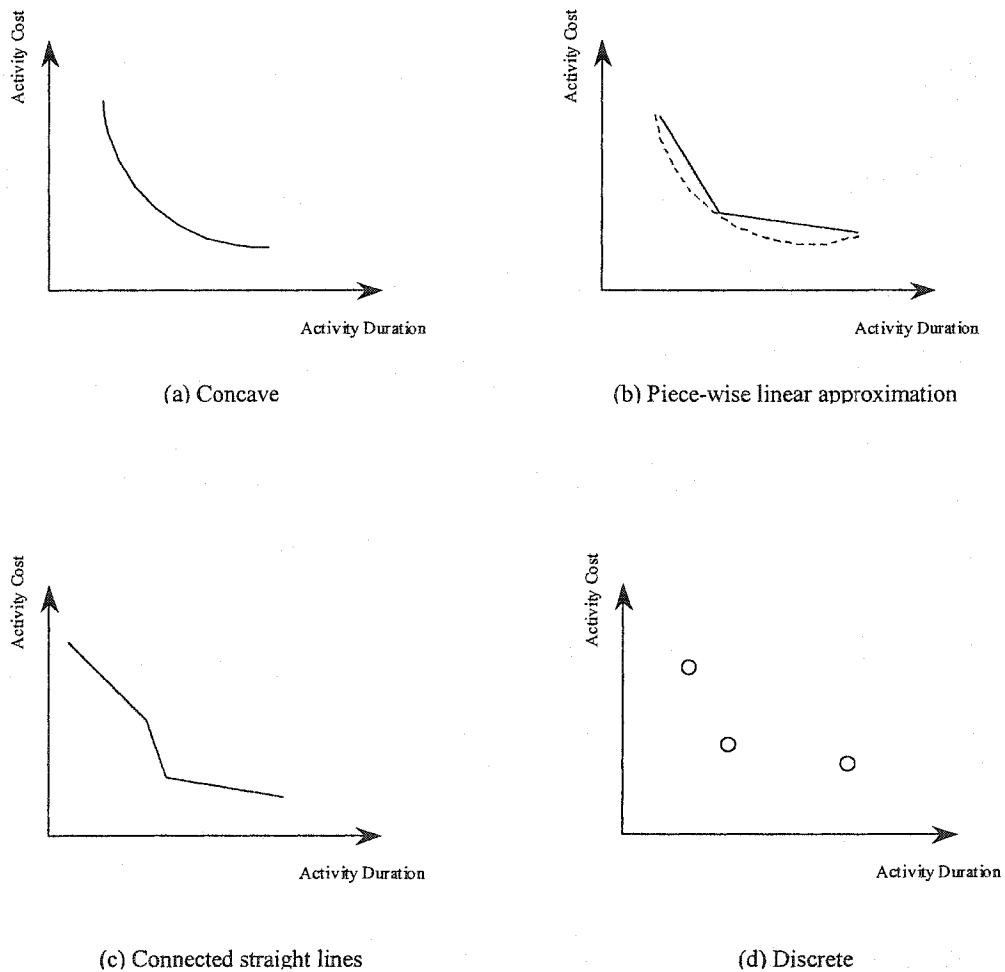


Figure 17. Different Time-Cost Models.

### 5.5 Criticisms of Current Practices (Constant Cost-Slope Concept)

Although the concept of “cost-slope”, Fig. 16, is appealing in its simplicity, it must be pointed out that it is often extremely difficult to obtain reliable figures for the changes in cost resulting from changes in duration time. These difficulties are so great that, in practice, the cost-slope concept may be inapplicable. Moreover, the relationship between cost and time is not a simple one. Multiplying labor time by wage cost is obviously inaccurate and, on the other hand, to “extend” the resultant labor cost by a constant overhead factor can be equally misleading, since the reduction in time may be obtained, for example, by hiring special plant that has a non-linear hiring rate. These difficulties make it dangerous to assume that cost slopes are constant.

### 5.6 Project Crashing with Mathematical Programming

The TCTO problem aims towards reducing the overall completion time of a project by ‘crashing’, i.e. reducing the time of a number of activities in the project while holding the total cost of the project to a minimum. As discussed in [160, 177-179], both CPM/PERT network and project crashing network can be formulated as a linear programming problem; to minimize the cost of crashing given the limits on how much individual activities can be crashed.

Project crashing with mathematical programming involves definition of the followings:

1. The decision variables: ‘x’ is the time an event will occur, and ‘y’ is defined as the number of time units that each activity is crashed.
2. The objective is to minimize the additional cost of crashing the project.
3. Three set of constraints, in addition to non negativity, are imposed on the model:
  - a. Network constraints. This set of constraints describes the structure of the network by specifying the precedence relationships among different network activities. There should be one or more constraints for each event. This set of constraints insures that no activity can start before the preceding activity(s) has been realized.

- b. Crash time constraints. This set of constraints determines the limit/extent to which an activity can be crashed. The maximum for each 'y' variable is equal to the difference between the normal time and the maximum allowable crash time.
- c. Project completion constraint. This constraint insures that the project schedule was set in such a way that the project will be completed within the desired time span. This is done by specifying that the last event (matching the end of the last activity) must take place before the desired project schedule deadline date.

### **5.7 The TCTO Problem in Literature**

The TCTO problem was the subject of a large number of research articles. Tufeki [180] introduced an iterative solution procedure for solving the time-cost trade-off problem that utilizes a labeling algorithm for locating a minimal cut in the flow network. Rosenblatt & Roll [181] analyzed optimal project duration for situations where project duration can be shortened by 'crashing' activities. The cost components considered are: regular direct costs, crashing costs and overhead costs. In Law & Hsing-Wei [182], two predictive models for estimating the computer execution time required by two network flow based algorithms to solve the time-cost trade-off problem were presented. Models were developed in [183] for two specific resource-constrained project crashing cases: (1) a model for resource critical crashing case, and (2) a model for the activity duration crashing case. A network in which each arc is associated with a time-cost trade-off function was considered in [184]. This function was assumed to be non-increasing, piecewise linear and convex and objected to enumerate all efficient chains in the context of two criteria, the total time and the total cost required to traverse from source node to sink node. An approximation algorithm for the discrete time-cost trade-off problem was presented in [185].

The importance of the TCTO problem arises from the wide range of its application involvements. Gander [186] introduced different forms of government involvement in the



innovation process, both direct and indirect, into a standard innovation TCTO model. Batson [161] discussed an implementation of a TCTO algorithm in aircraft technology development projects. Graves [162] presented a brief review of the key concept of a convex TCTO, which by assumption forms the basis for both static and dynamic models of research and development investment expenditure. The TCTO was used in [163] to maintain the planned market entry of a product. A cost-minimization model to investigate scheduling strategies for multistage projects in a client-contractor environment was considered in [187]. The model is designed primarily to address the interaction between earliest-, intermediate-, and latest-start options and project-crashing strategies for a broad range of penalty costs. Reda & Carr [188] handled the problem among related activities.

A survey of project scheduling problems since 1973 limited to work done specifically in the project scheduling area is found in [189]. The survey includes the work done on several fundamental problems such as the TCTO problem. De Reyck & Herroelen [190] investigated the relation between the hardness of a problem instance and the topological structure of its underlying network, as measured by the complexity index. It also demonstrates that the complexity index plays an important role in predicting the computing effort needed to solve easy and hard instances of the multiple resource-constrained project scheduling problem and the discrete TCTO problem. Hajdu [191] dealt with some special problems concerning least cost scheduling problem in precedence diagramming. De et al. [192] addressed the discrete version of the well-known TCTO problem for project networks, and discusses the complexities of various network structures and validate an old conjecture that certain structures are necessarily more difficult to solve. The discrete TCTO problem in which the duration of project activities were assumed to be discrete, nonincreasing functions of the amount of a single nonrenewable resource was addressed in [193]. The paper described a procedure for scheduling project activities in order to minimize the total cost of the project while meeting a given deadline.

The TCTO problem has been tackled by several methods. The following sections provide a short literature review on related research.

### 5.7.1 Mathematical Programming

A multi-objective project crashing model was introduced in [194] where the problem was formulated as a lexicographical optimization model. An efficient lexicographical maximal flow algorithm was implemented to obtain the lexicographical minimal cuts at each step to determine the activities to be modified. In Kanda & Rao [195], a procedure was developed to obtain the project-cost curve when there are linear penalty costs for delays of certain key events in a project in addition to crashing costs for activities. A linear programming formulation is given.

The model presented in [196] involved a mixed integer linear programming formulation to determine the optimum allocation of the project duration reduction. The main advantage of this model was its ability to determine the optimum allocation among activities for four different time/cost functions. Erenguc et al. [197] determined the activity durations and a schedule of activity start times so that the net present value of cash flows is maximized in a project scheduling problem. The problem was formulated as a mixed-integer nonlinear problem. An algorithm to assist construction planners in making TCTO decisions was presented in [198]. This approach, called the LP/IP hybrid method, took advantage of linear programming and the convex hull method for efficiency, and integer programming to find the precise solutions. This hybrid method, along with a spreadsheet tool, provides the construction planner with an efficient means to obtain resource selections that optimize time and cost of a construction project.

Two algorithms, based on dynamic programming logic, were described in [199] for optimally solving the discrete time-cost trade-off problem in deterministic CPM networks. An algorithm that employs an integer programming formulation for obtaining the optimal solution for the time-cost trade-off problem in large projects was presented in [200].

### 5.7.2 Heuristic Algorithms

Barber & Boardman [201] established the definition of an easy-to-use tool for project crashing problems with two key features: an algorithm to generate a range of increasingly pragmatic solutions by the inclusion of heuristics to portray real-world objectives and an intelligent knowledge-based system to assist in the generation of strategies and to postulate the resultant time-cost trade-off function, for each activity considered. Barber [202] presented a prototype system which allowed a project network to be portrayed graphically as a CPA network and then crashed using a heuristic algorithm with the aid of a knowledge based system. Bowman [203] presented a heuristic using the gradient estimators to give close to locally optimal performance relatively quickly for PERT networks. In Sunde & Lichtenberg [204], a new heuristic for TCTO which balances cost, time, and resources was presented. The new method was called net-present-value TCTO. In Taeho & Erenguc [205], a combination of the TCTO problem and the resource constrained project scheduling problem was solved using a heuristic procedure, a multi-pass algorithm.

### 5.7.3 Simulation

In Ramani [206], a computer simulation project has been outlined to achieve optimal crashing of a PERT network, where a probabilistic PERT model was converted into a deterministic CPM model for the purpose of carrying out the TCTO analysis. In Patrick & Topaz [207], a project-scheduling simulation model of the longwall move process was developed to analyze and assess the economic viability of innovative transfer methods and equipment. Longwall face-to-face equipment transfers or moves are the largest source of nonproductive time in a longwall-mining system.

#### **5.7.4 Artificial Intelligence**

A new procedure for the TCTO problem that involved new assumptions and a fuzzy linear programming formulation was demonstrated in [208]. Another algorithm based on genetic algorithms principles for construction TCTO optimization, and a computer program that can execute the algorithm efficiently were presented in [209]. Li et al. [210] presented a computer system called Machine Learning and Genetic Algorithms based System (MLGAS). With MLGAS, quadratic time-cost curves are generated from historical data and used to formulate the objective function that can be solved by the genetic algorithm. The capacity of the GA was enhanced to prevent premature convergence. When compared with an experienced project manager, MLGAS generated better solutions to nonlinear time-cost trade-off problems. To provide construction engineers with a more realistic way of analyzing projects' TCTO decisions, Feng et al. [211] presented a hybrid approach that combines simulation techniques and genetic algorithms to solve the TCTO problem under uncertainty.

#### **5.8 Summary**

This chapter provided a background on the role that project management techniques could play in a manufacturing environment especially in new product development projects.

In addition to planning, scheduling, and following-up activities, project managers are frequently confronted with the problem of having to reduce a project scheduled completion time (indicated CPM or PERT network analysis) to meet a pre-specified deadline. The chapter further presented a problem that project crashing or the time-cost trade-off (TCTO) problem. Different time-cost relationship models were presented and the classical TCTO practice was critiqued. Moreover, related research in literature concerning solving this problem was reviewed.

## CHAPTER VI

### DSM OPTIMIZATION AND ANALYSIS

#### 6.1 Objective

The main objective of the DSM analysis is simply to reduce the effect of feedback loops by:

1. Decreasing the number of feedback couplings to the maximum possible extent, then
2. Reducing the scope of the remaining feedback couplings by:
  - a. Bringing them as close to the diagonal as possible, and
  - b. Grouping activities in unsolved feedbacks into iterative blocks.

Figure 18 schematically shows these sequential steps of DSM analysis.

As discussed in Chapter II, to perform the former steps, most of the DSM analysis research work reported in literature adopted Steward's heuristic methodology and only two researchers applied an optimization technique (Genetic Algorithm).

The current research employs a mathematical-based approach to optimize the DSM. Hence, a quantitative objective function must be defined. Since the implementation of the simulated annealing algorithm, the optimization tool, is independent of the objective function formulation, different objective functions can be evaluated according to the data available. In the presented architecture, either one of four different quantitative objective functions can be used:

1. Number of feedback couplings,
2. Total project iterative time,
3. Total project iterative time and cost, or
4. Total project iterative load.

The following sections will elaborate more regarding the application of each of these objectives.

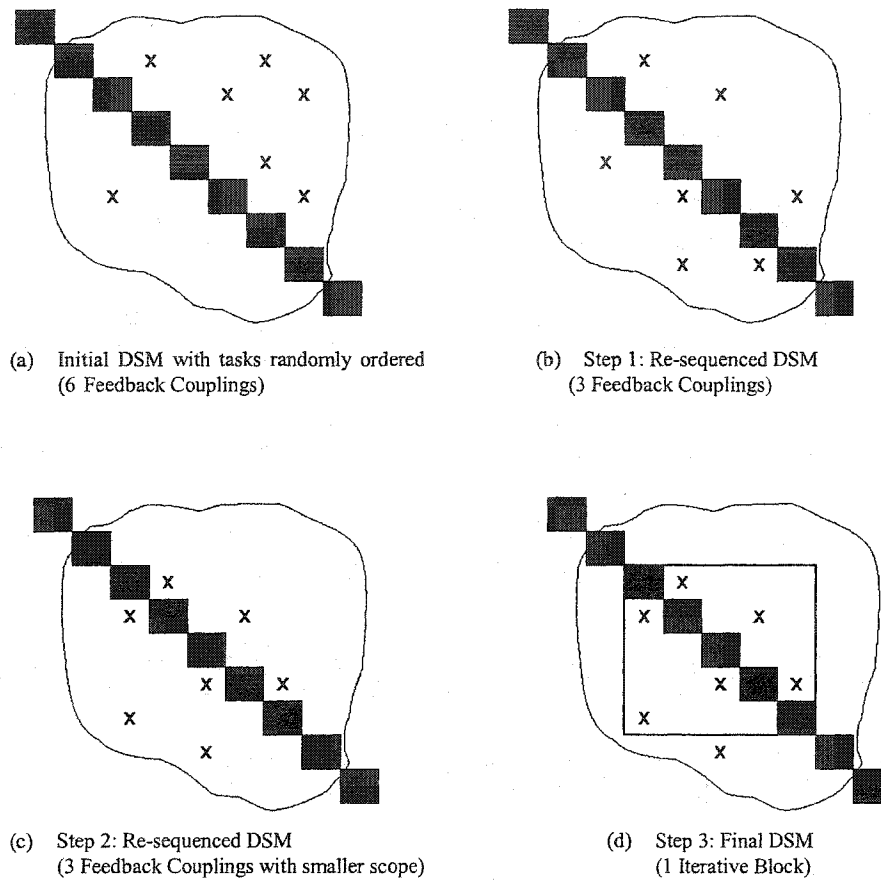


Figure 18. Schematic Representations of DSM Analysis Steps.

### The Concept of Load

To use an optimization technique, a quantitative objective function has to be defined. While 'AGENDA'<sup>14</sup> aimed towards minimizing the total length of feedbacks in the system, 'DeMAID/GA'<sup>15</sup> objective was to minimize total cost and time required for convergence.

<sup>14</sup> [47] – Refer to Section 2.4.4

<sup>15</sup> [39] - Refer to Section 2.4.1

Since the scope of the current research proceeds beyond *DSM optimization and analysis* to *project scheduling*, a new measure had to be defined to serve both parts of the presented framework. This measure is called ‘**activity load.**’

The presented research assumes that two parameters can be defined for each activity: (1) load, and (2) resource type. The nature of these parameters is dependent on the nature of the activity itself. For example, if the activity is digging a foundation for a building, then the load will be the number of cubic meters to be removed and the resource type needed would be man.

The advantages of using this concept are of two-fold: the first regarding the Time-Cost Trade-Off analysis, and this will be discussed in details in Chapter VIII; and the second is the generality for DSM optimization: if activity durations (time) are available, time will be used as the load (and the same applies in case of cost availability), and if it is only required to reduce the extent of the feedback loops, then a load of 1 unit will be assigned to all activities and the optimization proceeds.

## 6.2 Assumptions, and Limitations

The presented architecture aims toward finding an optimum sequence of all activities in the project. Although the current research is concerned with IPDPs, the architecture is general and can be applied in any environment.

The followings assumptions and limitation apply to the DSM model:

1. A basic concept: the DSM methodology assumes sequential execution of activities in the order shown on the DSM.
2. System decomposition proceeds to the smallest task, for example case (b) in Fig.19 is invalid, and must be replaced by something similar to case shown in (a).
3. An activity starts immediately after all the required input information to begin is available, i.e. as-early-as-possible.
4. As shown in Fig. 20-a, an activity can be started only after each of the predecessors has been entirely finished [22]. Or, in other words, an activity

cannot start until all required information input is available. This implies that no activity can start with preliminary information and also receives other information during its execution as in the case shown in Fig. 20-b.

5. Only planned iterations determined by the project manager or the system analyst at early phases are considered in this methodology. Unplanned iteration requirements that emerge during project execution are not incorporated in the presented model.
6. An activity provides its output information once it is finished (Fig.21-a), not during processing (Fig.21-b).
7. Start and end activities are assumed known. Thus, their order will remain fixed during the optimization process.
8. Each activity is assumed to be redone completely in each iteration.
9. In the case that an activity falls in more than one feedback loop, the activity is assumed to be redone a number of times equal to the sum of iteration factors of these feedback loops.
10. Kusiak et al. [212] classified dependencies in the design process into: information dependency, technological dependency, common-sense dependency, resource dependency, preferential dependency, and functional dependency. The proposed methodology considers only information dependencies. If required, other forms can be imposed as logical (hard) constraints.
11. To proceed with collapsing, coupling strengths are needed. If no coupling strengths were defined, a dummy coupling strength equals to 8 or 9 can be assigned to all couplings before optimization. Equal strengths will not affect the optimization results but will allow proceeding with collapsing.



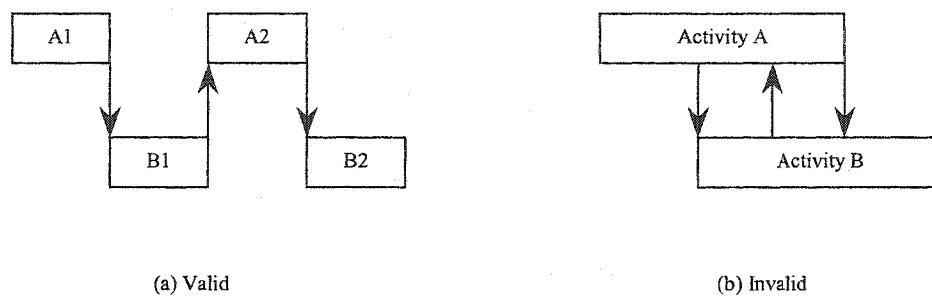


Figure 19. Assumption: System Decomposition.

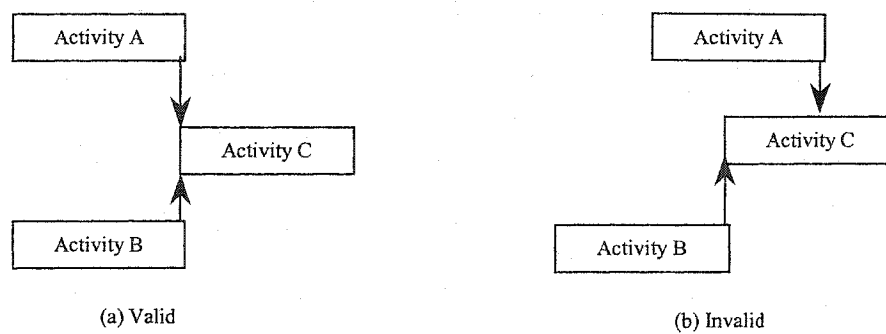


Figure 20. Assumption: Activity Start.

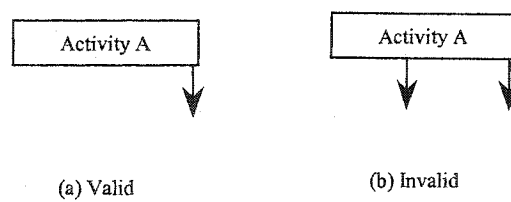


Figure 21. Assumption: Activity Output.

### 6.3 Mathematical Modeling of Feedbacks

A coupling between any two activities can be expressed as a mathematical relationship. For example, if a coupling  $coup(A, B)$  exists from activity ( $A$ ) to activity ( $B$ ), as shown in Fig. 22, a coupling indicator is defined as,

$$I_{A,B} = \begin{cases} 1 & \text{if } coup(A, B) \text{ is feed forward} \\ -1 & \text{if } coup(A, B) \text{ is feedback} \end{cases} \quad (10)$$



Figure 22. Coupling (General Form).

The value of the coupling indicator, thus, depends on the order of execution of both activities interfaced by this coupling. Now, to apply the concept of coupling indicator, let's consider the following example, shown in Fig. 23.

Let  $x_A$ ,  $x_B$  be the order of activities  $A$  and  $B$  respectively. In any DSM state, the coupling  $coup(A, B)$  can hold one of two cases:

Case 1: A feed forward coupling, Fig.23-a,

Since  $x_A = 11$  and  $x_B = 14$

Then activity  $A$  is realized before activity  $B$

i.e.  $x_B > x_A$  or  $x_B - x_A > 0$

Then  $coup(A, B)$  is a feed forward and, hence,  $I_{A,B} = 1$

Or, Case 2: A feedback coupling, Fig. 23-b

Since  $x_A = 14$  and  $x_B = 11$

Then activity  $B$  is realized before activity  $A$

i.e.  $x_B > x_A$  or  $x_B - x_A < 0$

Then  $coup(A, B)$  is a feedback and, hence,  $I_{A,B} = -1$

So, for any coupling  $coup(i, j)$  from activity  $i$  to activity  $j$ , a coupling indicator  $I_{i,j}$  is defined as:

$$I_{i,j} = \begin{cases} 1 & \text{if } x_j - x_i > 0 \\ -1 & \text{if } x_j - x_i < 0 \end{cases} \quad (11)$$

Where  $x_i$  and  $x_j$  are the order of execution of activities  $i$  and  $j$  respectively

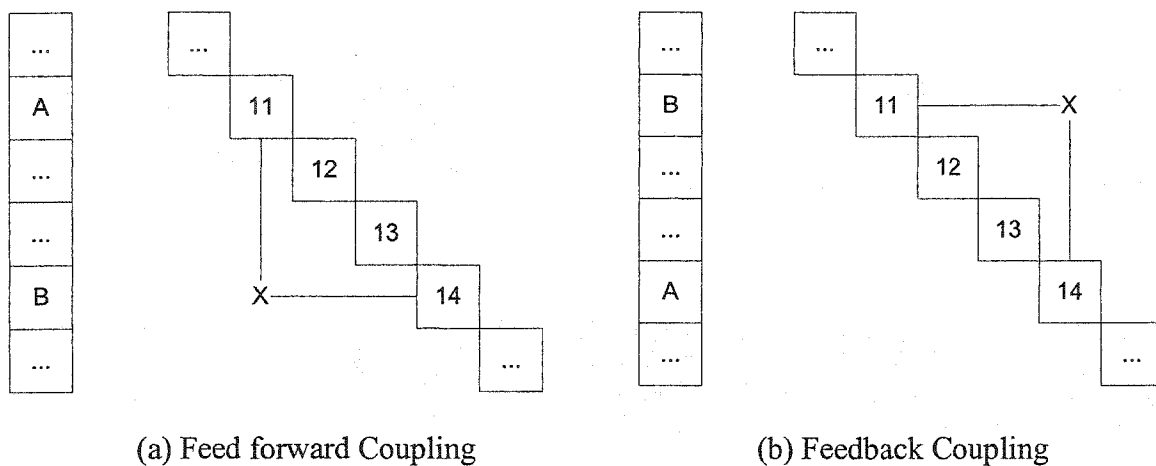


Figure 23. Feed forward vs. Feedback Coupling.

## 6.4 Logical Constraints

The current optimization model tolerates two sets of hard constraints, these are:

1. Due to the assumption that both the start and finish activities are known, their order is assumed fixed. Hence, the order of the start and finish activities will be '1' and 'm' respectively, where 'm' is the number of activities in the project.
2. In some cases it is infeasible to switch the direction of a coupling from a feedback to a feed forward, although this of course will reduce both time and

cost. For example, consider the case shown schematically in Fig.24. In this case, there is only one input to the activity “Initial Data” and this input comes from activity “Revised Data.” As a result, an unconstrained optimization will try to assign orders for these activities in a way that guarantees that the coupling (Revised Data, Initial Data) is a feed forward one because this, of course, shall improve the objective function(s). On the other hand, it can be easily noticed that this solution is infeasible in the sense that it contradicts to logic; “Revised Data” cannot be performed before “Initial Data”). Thus, to avoid having such an infeasible solution, a second set of hard constraints, logical constraints, is to be developed and tailored according to the nature of the problem.

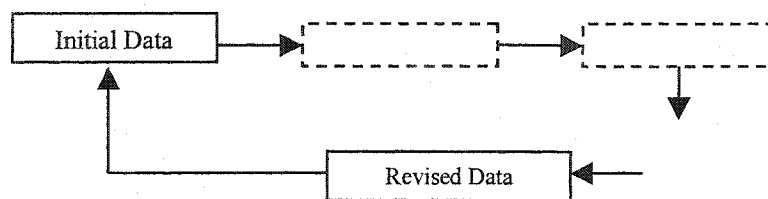


Figure 24. Hard Constraint.

### 6.5 Iteration Considerations

The number of iterations required for a certain feedback loop to converge differs from one feedback to the other depending on how good the original estimates used for upstream activities were on the sensitivity of downstream activities to these estimates and on the required quality of the final design. The trade-off here is that while the solution quality can be improved by performing more iterations, extra time and cost of doing so will be added to the total project as a result.

In order to incorporate such a factor in the presented model, an iteration factor is defined. This factor converts coupling strengths to the number of iterations required for convergence. To quantify coupling strengths, seven levels are used. These are: extremely weak, very weak, nominal, strong, very strong, and extremely strong. Although these strengths are supplied to the model directly, they can be determined through sensitivity analysis detailed in [41]. Thus, for each coupling, an iteration factor is determined based on the coupling strength according to the default values shown in Table 4.

Table 4. Iteration Factor Values.

Coupling Strength		Iteration Factor
Extremely weak	ew	2
Very weak	vw	3
Weak	w	4
Nominal	n	5
Strong	s	6
Very strong	vs	7
Extremely strong	es	8

## 6.6 Numerical DSM

The basic DSM is a binary matrix, where cells can hold one of two values ("one" or "zero") or ("X" marks or empty cells). Such matrix features a single attribute; the 'existence' (or 'absence') of an interface between different elements. DSM was later modified to hold multi-attribute, such DSM is referred to as a "Numerical DSM." Numerical DSMs allowed the development of more complex DSM analysis algorithms. For example, to guide the 'Tearing' process, Steward [22] suggested using a two-level numbers instead of the simple mark "X"; a coupling is assigned the value "0" if the

predecessor activity is *insensitive* and assigned the value “9” if the predecessor activity is *sensitive*. A predecessor activity is *insensitive* if it can be estimated with a high confidence level or if a bad estimation would not be of much effect on the results of the successor activity. More numerical DSM models are discussed in [9, 27, 50, 51]. In the current research, a numerical DSM is used; in which coupling marks are replaced with numbers (iteration factor) indicating the strength of the coupling.

## 6.7 Computing Load

Each activity has its associated load. And each feedback coupling has a number of iterations required for its convergence. In order to determine the project iterative load (PIL) due to feedback loops, for all activities contained within a feedback loop, load will be summed and multiplied by the loop’s iteration factor. Then, the loads of all feedback loops are summed.

To determine the iterative load for each loop, and hence for the whole project, the heuristic presented in Fig.25 is applied. In case of time and/or cost optimization, the same heuristic applies with a minor modification: PIL is replaced by PIC and PIT (project iterative cost and project iterative load respectively).

To illustrate the heuristic, consider the example shown in Fig. 26. This simple hypothetical DSM has five activities (associated loads shown) and two feedback couplings (corresponding iteration factors shown).

The steps for computing the *Project Iterative Load* go as follows:

1. Define the set of activities in the DSM and their associated loads:  

$$A = \{(1,10), (2,20), (3,30), (4,20), (5,10)\}$$
2. Determine the set of feedback couplings in the current DSM sequence and their associate iteration factor:  

$$C = \{(4,1),3\}, \{(5,3),5\}$$
3. Consider the first coupling in the set  $C$  : *coup*(4,1)

- a. Loads associated with activities within this loop are summed to determine the *coupling load*:

$$CL_{4,1} = 10 + 20 + 30 + 20 = 80 \text{ units}$$

- b. The coupling iterative load is determined by multiplying the previous quantity by the iteration factor corresponding the current feedback coupling:

$$CIL_{4,1} = 80 \times 3 = 240 \text{ units}$$

4. Step (4) is repeated for the coupling *coup(5,3)* :

$$CIL_{5,3} = 300 \text{ units}$$

5. Finally, project iterative load is equal to the sum of all iterative loads of feedback couplings in the DSM determined previously, thus

$$PIL = 240 + 300 = 540 \text{ units}$$

---

```

procedure Project Iterative Load;
  {Determining Project Iterative Load}
   $PIL = 0$  ; {initial project iterative load}
   $A = \{(x, load_x) : 1 \leq x \leq NA\}$  ;
   $C = \{(coup(i, j), IF_{i,j}) : I_{i,j} = -1\}$  ;
  repeat
    select_coupling(coup(i, j)) ;
     $CL_{i,j} = 0$  ; {initial coupling load}
     $CIL_{i,j} = 0$  ; {initial coupling iterative load}
     $k = 1$  ; {counter initial value}
    repeat
      select_activity(k) ;
      if  $x_j \leq x_k \leq x_i$ 
      then  $CL_{i,j} = CL_{i,j} + load_k$ 
         $k = k + 1$  ;
    until  $k > NA$  ; {all activities in A are considered}
     $CIL_{i,j} = CL_{i,j} \times IF_{i,j}$ 
     $PIL = PIL + CIL_{i,j}$  ;
  until all couplings in C are considered;
end;

```

---

<i>A</i>	Set of activities in the DSM and their associated loads
<i>NA</i>	Number of activities
<i>C</i>	Set of feedback couplings in current DSM sequence and their associated iteration factors
<i>coup</i> ( <i>i, j</i> )	Coupling from activity ( <i>i</i> ) to activity ( <i>j</i> )
<i>IF</i> <sub><i>i,j</i></sub>	Iterative factor of <i>coup</i> ( <i>i, j</i> )
<i>I</i> <sub><i>i,j</i></sub>	Coupling indicator
<i>select_coupling</i>	A function that selects, in order, a coupling from the set <i>C</i>
<i>select_activity</i>	A function that selects, in order, an activity from the set <i>A</i>
<i>load</i> <sub><i>k</i></sub>	Load associated with activity <i>k</i>

---

Figure 25. Load Computations Heuristic.



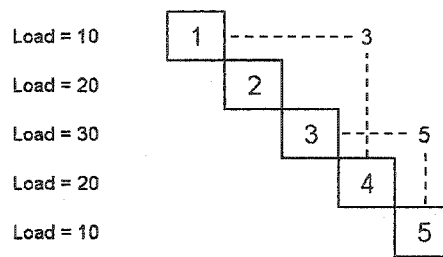


Figure 26. Load computations example.

## 6.8 Optimization with Simulated Annealing

The objective of DSM optimization is to determine the sequence of activities execution that results in minimum project iterative load. In this section, a modified simulated annealing algorithm is proposed and its implementation steps are explained.

### 6.8.1 A two-stage simulated annealing algorithm

The SA algorithm proposed in this research (called two-stage SA) is a modified version of the Naïve SA algorithm presented in Section 3.4.3. The presented algorithm (Fig. 27) follows the same steps of the Naïve SA but adds a second stage that keeps record of the value of a best solution. The objective of this modification is to assure that the final solution provided is the best one achieved. Thus, in cases that SA moves towards a locally optimal solution, the algorithm can be redirected to avoid being trapped in it.

### 6.8.2 Decision Variables

In DSM optimization, it is required to determine the optimal sequence of project activities

$$X = (x_1 \ x_2 \ x_3 \ \dots \ x_i \ \dots \ x_{NA})^T \quad (12)$$

Where  $x_i$  is the order of execution of activity  $i$ ,  $0 \leq i \leq NA$ , and  $NA$  is the number of activities. A constraint is to be imposed here is: no duplication is allowed (i.e. two activities cannot assume the same order).

### **6.8.3 Generation of The Initial Solution Configuration**

In most of the research cited in literature, SA has proved to be a robust optimization algorithm that is independent of the initial solution configuration. Hence, in the presented research, an initial solution configuration is generated by randomly assigning order to different activities, as shown in Fig. 28. Another alternative that can be considered is generating an initial solution using Steward's partitioning procedure presented in Section 3.3.

### **6.8.4 Generation of a Feasible Neighboring Solution Configuration**

In simulated annealing, a new solution configuration is generated by perturbing the current one. Several perturbation methods are cited in literature. The use of any of these methods mainly depends on the nature of the problem being tackled. The presented research applies a "pair-wise exchange" perturbation strategy. As shown in Fig. 29, this is done by randomly selecting two activities and 'swapping' them. It should be noted here that the order of both the start and finish activities remain fixed.

Some thoughts need to be given to the generation of a feasible solution when 'logical constraints' exist. In this case, the algorithm searches only the feasible space by being programmed to reject any proposed solution configuration that results in a constraint violation. Another alternative is adding a penalty to the objective function. But since that technique may lead to wasting time computing infeasible solutions, the research here sticks to the first alternative.

---

```

procedure Simulated Annealing;
  {Two-stage Simulated Annealing Algorithm}
   $S := S_0$ ; {initial solution}
   $S_{opt} = S$ ; {initial optimal solution}
   $T := T_0$ ; {initial temperature}
  repeat
    repeat
       $S' = perturb(S)$ ;
       $\Delta = E(S') - E(S)$ ;
       $\theta = random[0,1)$ ;
       $prob = e^{-\Delta/T}$ ;
      if  $\Delta < 0$  or  $prob \geq \theta$ 
      then  $S := S'$ 
        if  $E(S') < E(S_{opt})$ 
        then  $S_{opt} = S'$ ;
      else retain  $S$ ;
    until inner loop stopping criterion is met;
     $T = update(T)$ ;
  until outer loop stopping criterion is met;
end;

```

---

$S$	The current solution. The initial solution, $S_0$ , is a feasible solution generated either randomly or through using some heuristics.
$S_{opt}$	Optimum solution
$T$	The control parameter.
<i>perturb</i>	A function that generates a new neighboring solution, $S' \in N(S)$ , through introducing some small perturbation to the current solution, $S$ .
<i>random</i>	A random number generator.
<i>update</i>	Cooling schedule function.

---

Figure 27. Two-Stage Simulated Annealing.

Order	1	2	3	4	5	6	7	8	9
Activity	A	F	C	E	B	H	D	G	I

Fixed
Fixed

Figure 28. Solution Representation.

Order	1	2	3	4	5	6	7	8	9
Activity	A	F	C	E	B	H	D	G	I

Order	1	2	3	4	5	6	7	8	9
Activity	A	F	H	E	B	C	D	G	I

Figure 29. Generating a Neighboring Solution.

### 6.8.5 Object Function Evaluation

As discussed earlier, SA merely requires the value of the objective function for each solution configuration. The algorithm picks up the objective function value of the proposed solution, compares it with the one of the current optimal solution and proceeds to the next step.

In the presented research, either one of four objective functions can be used to evaluate the proposed solution configuration. These are:

1. Number of feedback couplings,  $f_1(X)$

$$f_1(X) = \sum_{\text{coup}(i,j) \in A} I_{i,j} \quad (13)$$

2. Project iterative load

$$f_2(X) = PIL \quad (14)$$

3. Project iterative time

$$f_3(X) = PIT \quad (15)$$

4. Project iterative time and cost

$$f_4(X) = w_T \cdot PIT + w_C \cdot PIC \quad (16)$$

where

$w_T$  and  $w_C$  are relative weight selected by the decision-maker.

It is important to note, however, that the fourth objective function represents a multi-objective optimization case.

One important question that arises here is: “why not minimize the *project load* rather than *project iterative load*?” The answer can be stated as follows:

An important goal of the DSM analysis is simplifying the nest of feedback couplings, which in turn leads to a better management practice. Although minimizing the project load would theoretically result in a shorter time and lower cost, the possibility of being associated with a high number of feedbacks will, in reality, increase the complexity for management which, as a result, increases project total time and cost. Thus, minimizing project iterative load would tend to lower the time and cost of the project while providing an easy to manage sequence (reduced number of feedbacks).

#### 6.8.6 Cooling Schedule

The presented research adopts a geometric cooling schedule, which is the most commonly used cooling schedule in the simulated annealing literature [213]. In this schedule, temperature updating follows Eq. (7) The initial and final temperature values, referred to as  $T_0$  and  $T_f$ , respectively, are specified by the user along with the cooling factor,  $\alpha$ .

### 6.8.7 Stopping Criterion

To determine whether the system reaches a meta-stable state, two counters were introduced to keep track of the number of accepted and rejected solutions at each temperature. Iterations at each temperature halt when either counter reaches a pre-defined threshold. The user specifies both thresholds. The optimization process, on the other hand, proceeds until it reaches the final temperature,  $T_f$  even if no improvements are made during many temperature decrements.

## 6.9 Handling Stochastic Activities Load

In cases where activity loads assume stochastic values, Monte Carlo simulation is used to determine the value of the objective function for each proposed solution. The objective function in this case will also be represented by a probability distribution curve. And since the simulated annealing algorithm was developed to handle deterministic combinatorial optimization problems (the acceptance or rejection of a new solution follows metropolis criterion, which is based on one point estimate of the objective function), a modification had to be done to SA to tolerate stochastic objective functions.

### 6.9.1 Uncertainty in Activity Load Estimation

One unavoidable difficulty in the preparation of activity load estimate is uncertainty. Uncertainties in estimates of resource requirements for future system development arise from different sources: (1) deviations from original system configuration as the development process advances, (2) variations in resource (s) performance and cost, (3) system analyst biases, (4) errors in system modeling, and so forth...

Thus, the estimate of an activity load can be in the form of a multi-point (probability distribution function) not by a single value (one-point estimate). In turn, the system performance (objective function) will be expressed as a probability distribution so that it reflects uncertainty of estimates.

With explicit information describing each activity load, the comparison of two alternative system configurations can be aided by determining some quantitative statistical measure of the expected system performance of each alternative. According to their preferences and attitude toward risk, decision-makers should be able to choose the preferable alternative

The current research presents a modified version of the simulated annealing algorithm to extend its application to stochastic problems where the value of the objective function is represented by a probability distribution rather than a one-point estimate. The basic idea in the modified SA is that the acceptance or rejection of a proposed solution is based on the comparison of some statistical measures of its objective function distribution with those of the current optimal solution. Two methods for such comparison were suggested:

1. Min-Mean-Max rules, or
2. Utility Function Method.

While the first method assumes risk-averse decision-making, the second can be easily modified according the requirements of the decision-maker.

### 6.9.2 Min-Mean-Max (M3) Method

In this method, acceptance of a proposed solution is based on comparing three statistical measures of its objective function with those corresponding to the current optimal solution. These measures are:

1. The mean (expected, or average) point ( $\mu$ ),
2. The maximum value ( $\max$ ), and
3. The minimum value ( $\min$ ).

Let  $p$  and  $c$  denote the proposed and current solutions respectively. Solutions  $p$  and  $c$  are to be compared based on their objective function (*o.f.*). Figures 30 and 31 shows eight cases in which objective function estimates are expressed as probability distributions to reflect the actual uncertainty associated with each solution configuration evaluated. Figure 30 illustrates cases in which the mean value of  $p$  (denoted to by  $\mu_p$ ) is less than

the mean value of  $c$  (denoted to by  $\mu_c$ ). In case (1), the decision of accepting  $p$  is of no question since all possible objective function values are lower than those of  $c$ . the situation in case (2) is slightly different in that there is some probability that the actual value of  $p$  will be higher than  $c$ . If this probability is not high, the decision would be, still, to accept  $p$ . However, as this probability increase and the overlap is significant, as shown in case (3) and (4), pre-defined acceptance rules are needed. Figure 31 illustrates the flow chart that is used as basis for acceptance, or rejection, of the proposed solution in cases where  $\mu_p$  is less than  $\mu_c$ . Of course, these rules and the ratios included can be tailored according to the environment of the project carried out.

Similar rules for cases in which  $\mu_p$  equals to  $\mu_c$  (shown in Fig.32) are defined via the flow chart presented in Fig.33. Again, these rules are relative and can be changed according to the decision-maker attitude toward risk. Finally, when none of these cases holds, metropolis criterion is applied on the mean values of the distributions.



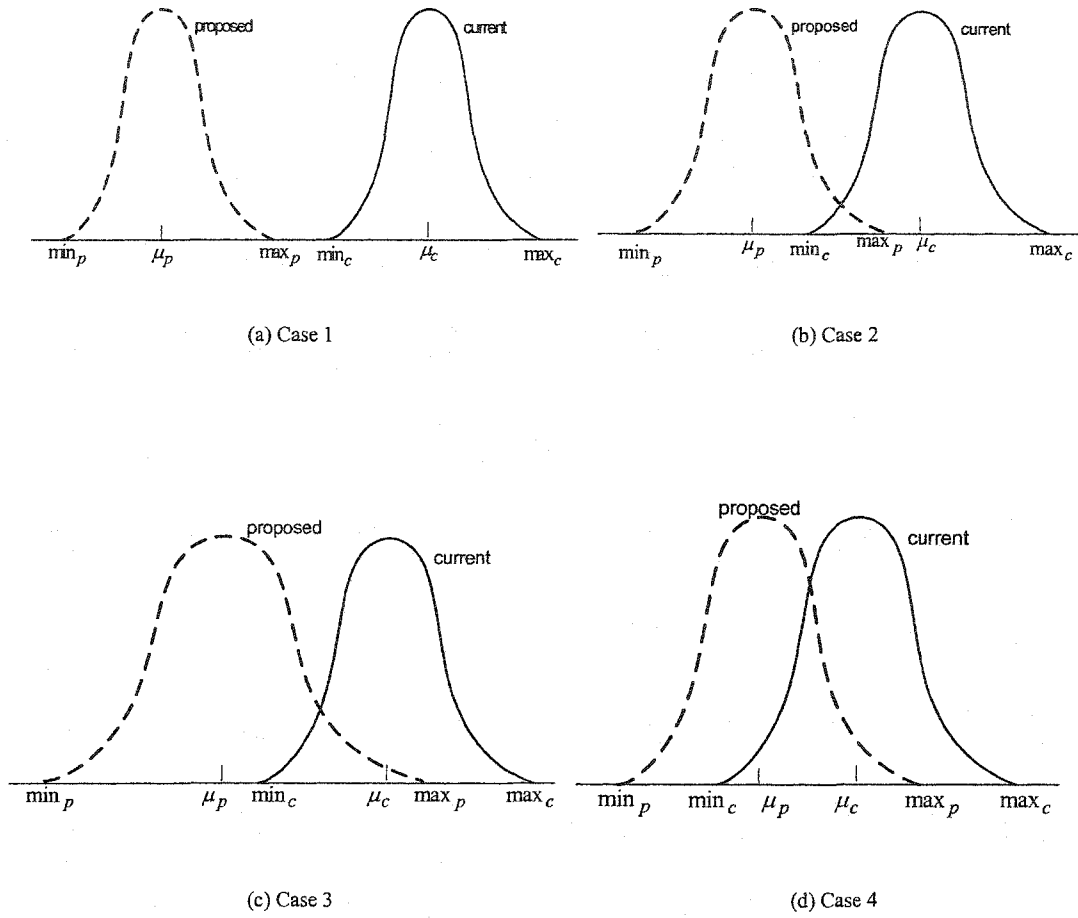


Figure 30. Cases with  $\mu_p < \mu_c$  (Schematic).

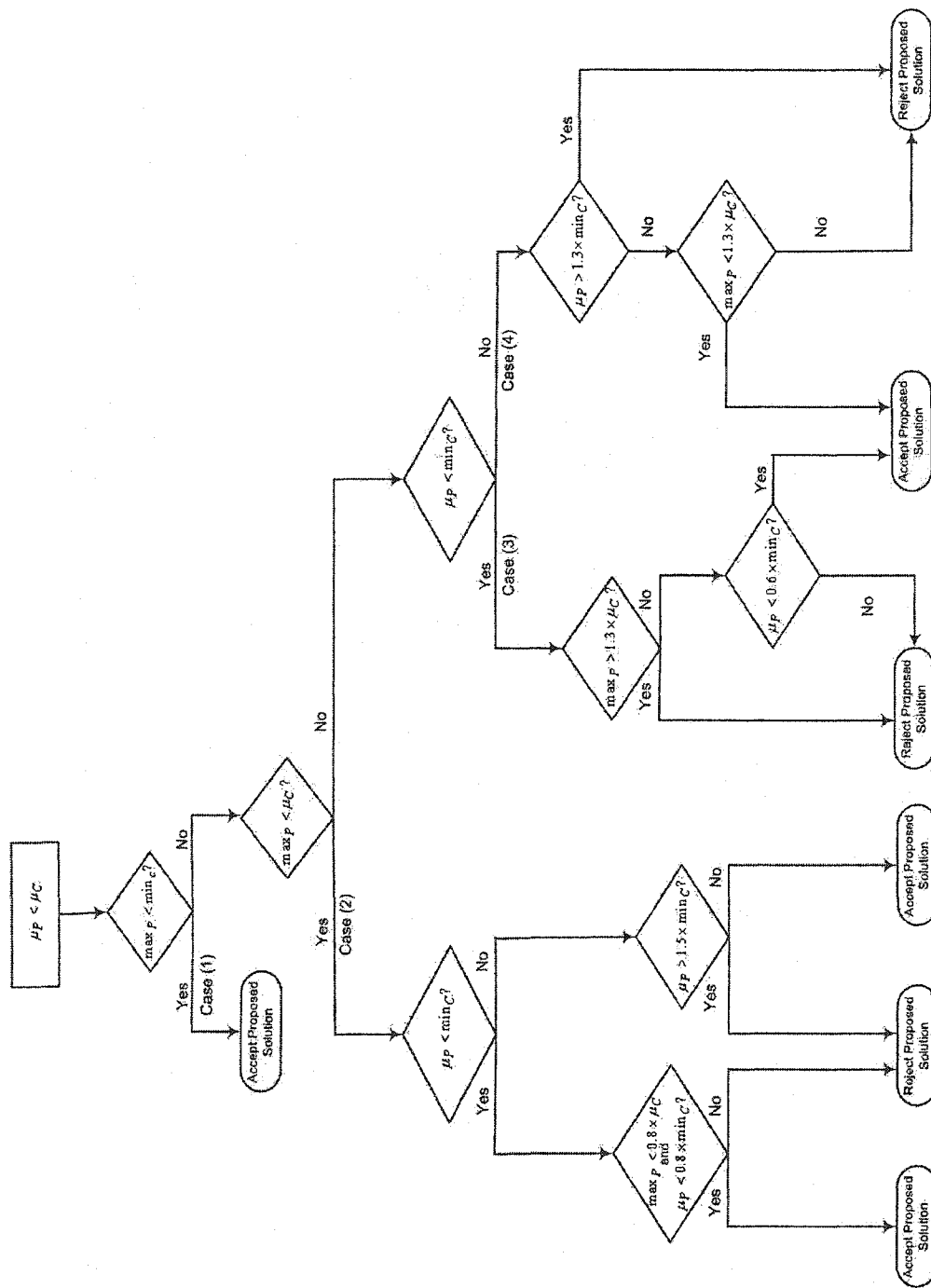


Figure 31. First Set of Comparison Rules.

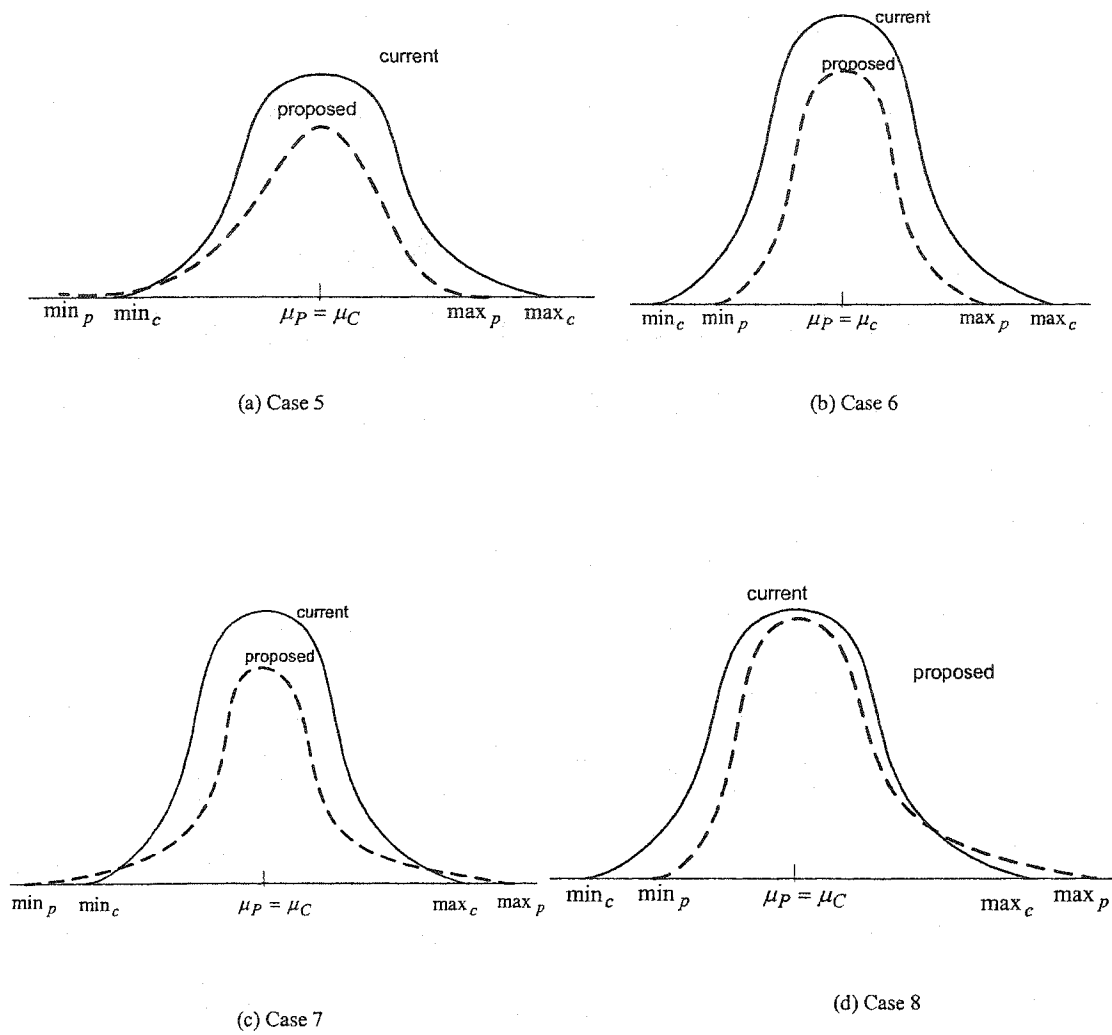


Figure 32. Cases with  $\mu_p = \mu_c$  (Schematic).

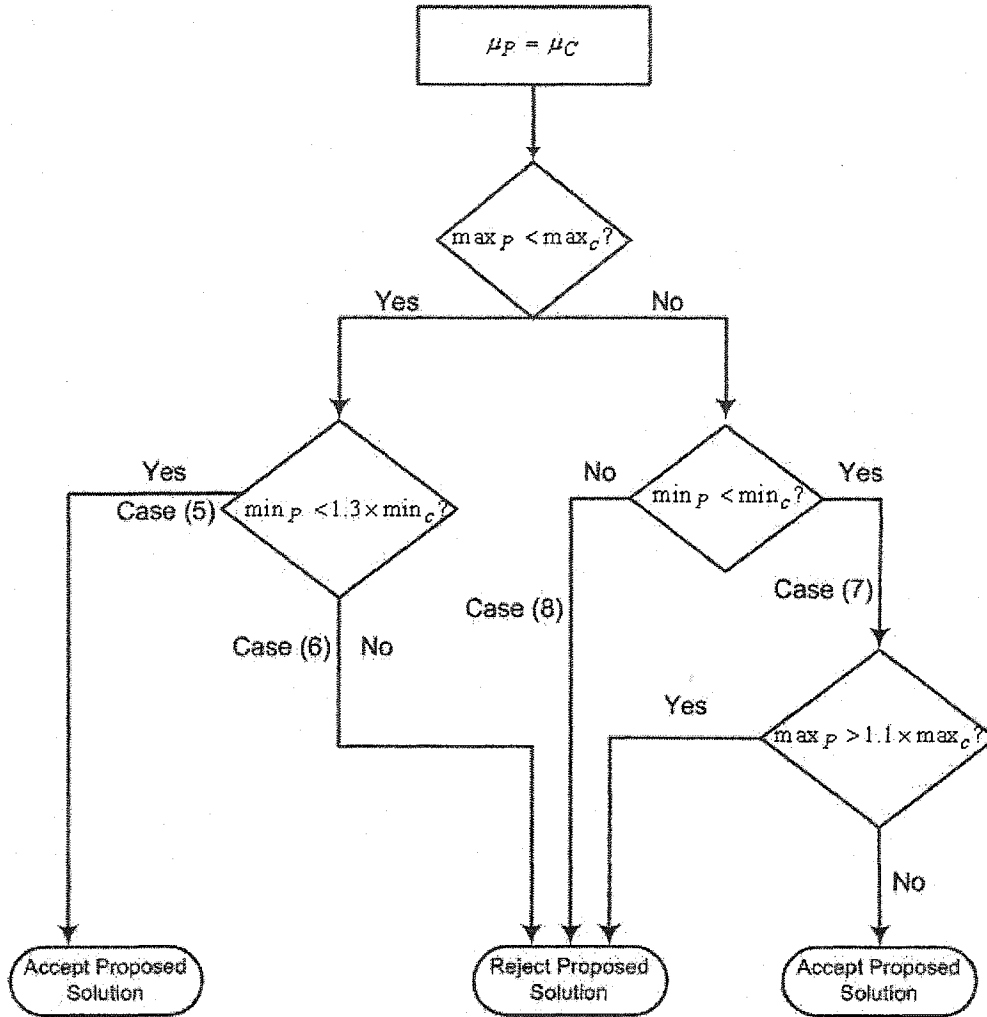


Figure 33. Second Set of Comparison Rules.

### 6.9.3 Utility Function (UF) Method

An alternative method for comparing the performance statistics (objective function) of a proposed solution with those of the current optimal solution is through the use of a 'utility function.' The proposed utility function (UF): (1) is an additive UF, (2) consists of four attributes, and (3) assumes attributes independence.

Four statistical measures were chosen to be the attributes of the UF, these are: the mean (or expected value), the variance, the range, and the maximum value. The first measure, the mean, is a central tendency measure. The concept is familiar and unique to all decision makers. Moreover, it is based on all observations. Thus, the mean is greatly affected by any extreme value, a useful characteristic here because the methodology tends to be a risk averse. The second and third measures are variation measures. The variance considers how the observations distribute or cluster and measures the average scatter around the mean and the range measures the total spread in the data. Finally, the fourth measure, maximum value, helps identify to what extent the value of objective function might reach.

A proposed solution is accepted if its UF is larger than the UF of the current solution. Otherwise, Metropolis criteria is applied. The objective here is to determine a robust solution rather than just an optimal one by minimizing the effect of uncertainty (variation) in activity loads on the objective function. As shown in Fig. 34, a robust solution is less sensitive to variations in activity loads (the uncontrollable parameters) than the traditional optimal solution where optimization is based on the mean value of the objective function.

Having the components in place, the key element for SA comparison rules is the utility function, which combines performance attributes and allows direct comparisons of solutions.

$$UF = (w_m \times I_m) + (w_v \times I_v) + (w_r \times I_r) + (w_x \times I_x) \quad (16)$$

where

- $UF$  : utility function  
 $w_m, w_v, w_r,$  and  $w_x$  : weights defined by user  
 $I_m, I_v, I_r,$  and  $I_x$  : index of mean, variance, range, and maximum values respectively. Determined from Table 5.

The weights introduced in the UF serve as importance factors. Their values are adjusted based on the decision maker attitude towards risk

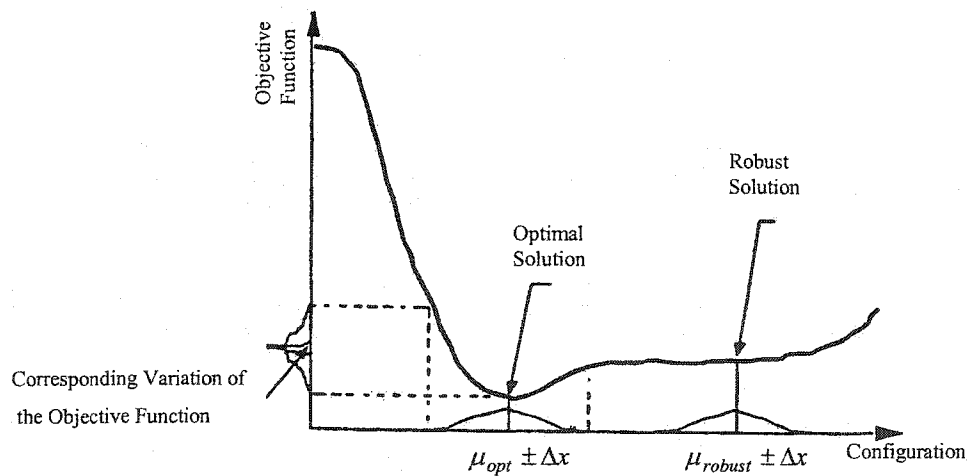


Figure 34. Robust vs. Optimal solution (Reference [214]).

Table 5. Utility Function indices.

Index of	Condition	Index value (if the condition holds)
Mean, $I_m$	$mean_p \leq 0.6 \times mean_c$	5
	$0.6 \times mean_c < mean_p \leq 0.8 \times mean_c$	3
	$0.8 \times mean_c < mean_p < mean_c$	1
	$mean_p = mean_c$	0
	$1.2 \times mean_c \geq mean_p > mean_c$	-1
	$1.4 \times mean_c \geq mean_p > 1.2 \times mean_c$	-3
	$mean_p > 1.4 \times mean_c$	-5
Variance, $I_v$	$var_p \leq 0.6 \times var_c$	5
	$0.6 \times var_c < var_p \leq 0.8 \times var_c$	3
	$0.8 \times var_c < var_p \leq var_c$	1
	$var_p = var_c$	0
	$1.2 \times var_c \geq var_p > var_c$	-1
	$1.4 \times var_c \geq var_p > 1.2 \times var_c$	-3
	$var_p > 1.4 \times var_c$	-5
Range, $I_r$	$range_p \leq 0.6 \times range_c$	5
	$0.6 \times range_c < range_p \leq 0.8 \times range_c$	3
	$0.8 \times range_c < range_p < range_c$	1
	$range_p = range_c$	0
	$1.2 \times range_c \geq range_p > range_c$	-1
	$1.4 \times range_c \geq range_p > 1.2 \times range_c$	-3
	$range_p > 1.4 \times range_c$	-5
Maximum, $I_x$	$max_p \leq 0.6 \times max_c$	5
	$0.6 \times max_c < max_p \leq 0.8 \times max_c$	3
	$0.8 \times max_c < max_p < max_c$	1
	$max_p = max_c$	0
	$1.2 \times max_c \geq max_p > max_c$	-1
	$1.4 \times max_c \geq max_p > 1.2 \times max_c$	-3
	$max_p > 1.4 \times max_c$	-5

## 6.10 DSM Conversion into a Project Schedule

One of the main advantages of DSM over project networks is its ability to represent feedback relationships. This feature is what allows the DSM to be the basis for an efficient planning of the design project. However, DSM has two major limitations: (1) a single DSM shows only a single process flow; it does not show all possible flow paths [215], and (2) the DSM does not explicitly show overlapping activities [26].

For the DSM to serve as a means of control of the design project (continual re-planning, re-scheduling, and follow up), activities in the optimally re-sequenced DSM need to be represented against a time scale. Or, in other words, the DSM has to be converted into a schedule. Thus, the DSM, in fact, does not replace the Gantt chart and CPM, but rather, they compliment each other. In this section, a three-phase procedure is presented to generate a project schedule from the final optimized DSM.

### 6.10.1 Related Practice

Three related methods were found in literature. In this section, each of these methods will be summarized and all of them will be applied to a hypothetical DSM for comparison. Consider the 10-activity DSM shown in Fig. 35 (adopted from [216]).

Initial ordering of activities, Fig. (a), results in 6 feedbacks and 8 feed forward couplings. Figure (b) represents the same DSM after partitioning. Only 4 feedback couplings and their extent are reduced. Steward's methodology (the methodology adopted for DSM analysis in all three investigations compared here) results in three iterative blocks; (Activity B, Activity F), (Activity J, Activity G), and (Activity E, Activity I, Activity C, Activity H).

Consider the largest block (Activity E, Activity I, Activity C, Activity H). Figure 6.18 illustrates different methods of converting this block into a program:



1. Steward [22] suggested 'unwrapping' each block "by laying it out end to end the number of times it is to be iterated." This will result in a program with no blocks (i.e. no feedbacks) shown in Fig. 36-a.
2. Austin et al. [44] suggested several strategies for conversion. In the one shown in Fig.36-b, activity durations are allocated independently and activities are programmed with in the block to start simultaneously.
3. Finally, Cho [48] suggested presenting the iterative block as a 'rolled-up' activity within which its tasks are arranged without feedbacks, see Fig. 37-c. A dummy activity is added at the end of the block representing its duration.

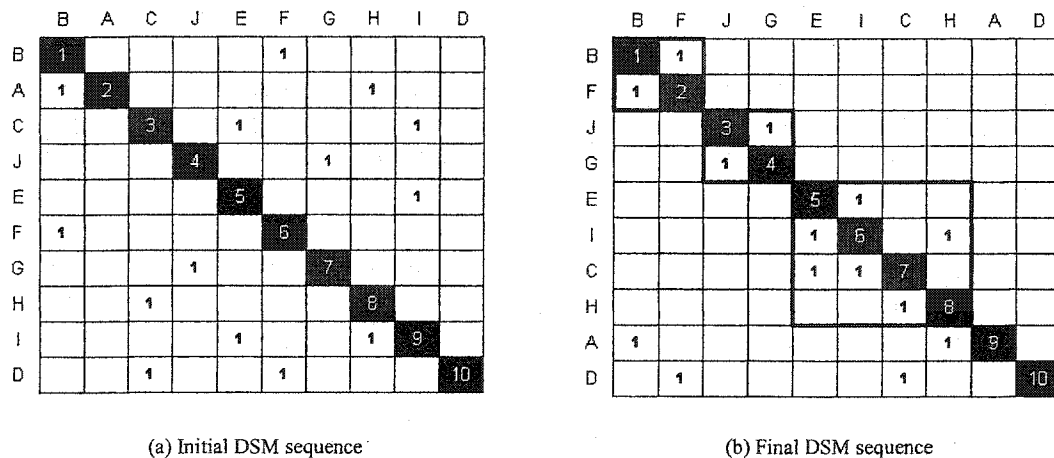
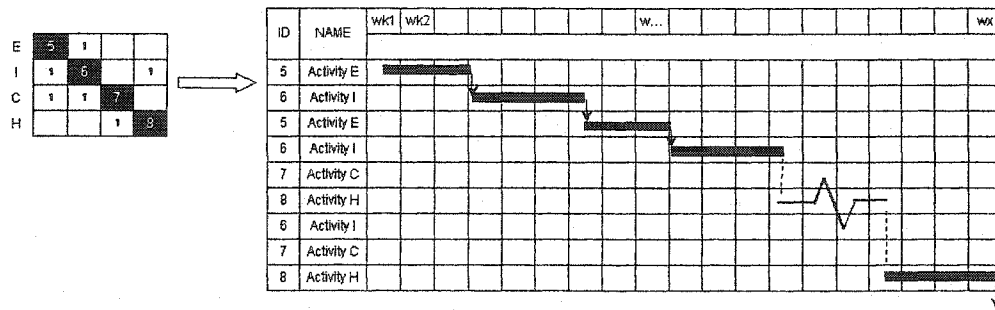
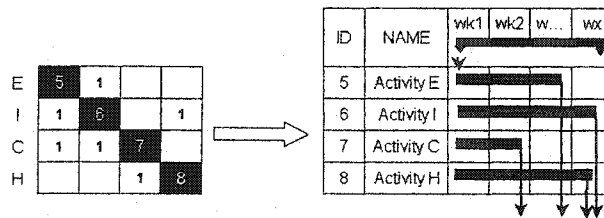


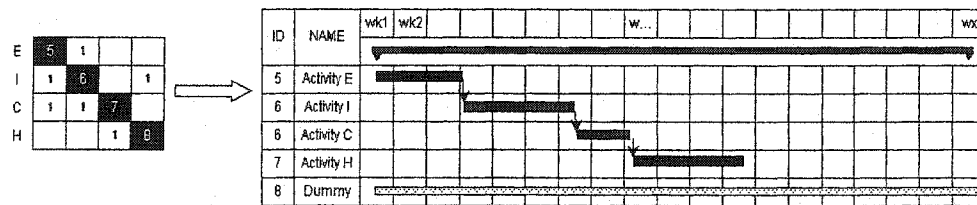
Figure 35. DSM of a Hypothetical Project.



(a) Equivalent Schedule According to Steward [22].



(b) Equivalent Schedule According to Austin et al. [44].



(c) Equivalent Schedule According to Cho [48].

Figure 36. Converting the Final DSM to a Project Schedule.

***Critique of reviewed methodologies***

One main advantage of DSM is the compact presentation of the project. This feature is wasted by unwrapping the blocks and dealing with each iteration as an independent activity. Repeating the same activity on PERT chart expands the project to an extent that

makes it hard to control and follow up. In many cases, partitioning would lead to blocks of large number of activities, for which unwrapping is not a wise idea.

Another form of the same problem with large blocks applies to ADePT where grouping a large number of tasks into one block would also result in meaningless representation. Furthermore, rules by which block duration is determined were not clearly mentioned.

Finally, all three methods implicitly assume that the DSM can be decomposed into totally independent blocks. Thus, in cases where the DSM can be only decomposed into blocks with inter-blocks relationships (i.e. if an activity falls into two or more feedback loops) these methods become inapplicable.

### **6.10.2 The Proposed Conversion Procedure**

Following the optimization process, the resulted optimally re-sequenced DSM, then needs to be converted into a schedule in order to proceed with resource assignment (as will be discussed in Chapter VII). The sequence of the activities, associated loads, and precedence relationships among them are defined by the output from the DSM. However, a methodology is needed to de-couple un-solved feedback couplings. The current research introduces a methodology that involves three main phases to produce an equivalent DSM without feedback couplings, which can be easily represented as a project schedule. These stages are: patterns recognition, collapsing, and tearing.

#### DSM Complexity Index

In some cases, as will be illustrated later, the number of couplings involved in a DSM can be so large to the extent that collapsing and tearing would be meaningless, or in other words, the proposed conversion procedure becomes inapplicable.

To help determine whether or not the architecture can proceed with conversion, a new index is introduced:

$$DSM \text{ Complexity Index} = \frac{\text{No. of Couplings}}{\text{No. of Activities}} \quad (18)$$

The architecture can proceed with conversion only in cases in which the *DSM Complexity Index* is less than 0.4. Of course, the index is used as a general indicator, and some DSMs with higher index value can be converted to a project schedule.

### A. Patterns Recognition

This phase concerns identifying some patterns in the activities of the optimized DSM. The work is inspired by the idea of improving the management of the design process by identifying some patterns in the DSM presented in Kusiak et al. [212]. Their basic idea was to classify some possible patterns in the design process structure and determine a critical activity for each pattern based on the expected behavior of that pattern. Relevant work can be found in [217-219].

Six patterns in the design process structure were defined by Kusiak et al. [212]. The presented methodology here adopts a similar procedure of defining some patterns. But, to cope with the research objectives, patterns classification here is mainly based on some characteristics of feedback couplings. The current research defines five patterns, shown in Fig. 37, these are:

1. I-Pattern (Interaction). A directed 2-cycle graph (a cycle with length 2). Two activities  $A$  and  $B$  are said to be of I-pattern if:
  - i. The couplings  $coup(A, B)$  and  $coup(B, A)$  exist, and
  - ii.  $|x_A - x_B| = 1$ .
 Where  $x_A$  and  $x_B$  are the order of activities  $A$  and  $B$  respectively. As shown in figure (a), such activities are highly-coupled. Tight collaboration has to take place between both of them to reach the desired solution (output).
2. C-Pattern (Cycle). A directed cycle with length = 3. Three activities  $A$ ,  $B$ , and  $C$  constitute a C-pattern if:

- i. The couplings  $coup(A, B)$ ,  $coup(B, C)$ , and  $coup(C, A)$  exist, and
  - ii.  $x_A < x_B < x_C$ .
3. C4-Pattern. a Special case of C-Pattern; a directed cycle with length = 4
  4. L-Pattern (Loop). A special case of the I-pattern. Two activities  $A$  and  $D$  constitute an L-pattern if:
    - i. The couplings  $coup(A, D)$  and  $coup(D, A)$  exist, and
    - ii.  $2 \leq |x_A - x_D| \leq 3$ .
  5. S-Coupling (Single Feedback).

Based on the strength of couplings involved in each of the recognized patterns, activities within iterative sub-cycles are either merged into one block (collapsing), or the feedback coupling is removed (tearing). Thus, the following two phases (collapsing and tearing) are carried out simultaneously for each pattern.

The methodology further involves introducing two types of buffers: block (coupling) buffers and a project buffer. The basic idea of adding buffers is adopted from [220]. For more details refer to [221]. The objectives of the introduced buffers are:

1. Compensate for uncertainty in activities estimated durations.
2. Helps controlling the project.
3. Compensate for iterative load of the coupling removed.
4. To tolerate incomplete information (change in information), such as durations of activities.

### B. Collapsing

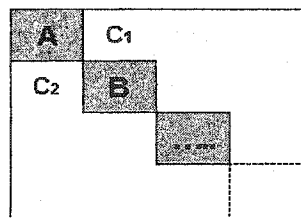
In some cases (patterns) where tight collaboration between two or three activities is required, these activities can be collapsed (merged) into one block. The presented procedure specifies two cases in which this collapsing must take place, these are: case (3) of the I-Pattern and case (4) of the C-Pattern shown in Table 6 and Table 7 respectively. In both cases, couplings involved are characterized by high coupling strength. Which means that information flow between these activities is expected to be of high density

until the required output is reached. And moreover, the output is expected to be sensitive to activities collaboration. Thus, merging these activities into one block will simplify the scheduling process, improve project management, enhance collaboration, and improve final solution quality. Since tight collaboration is forced on activities involved in both cases, processing time is expected to be lower than the calculated one. Thus, the block is assigned only half of the load originally assumed for both activities iterations. As a factor of safety, a percentage of the remove load is added to the 'project buffer.'

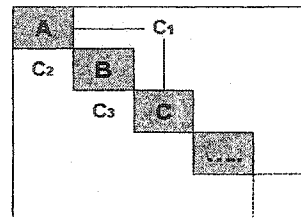
### C. Tearing

In cases where moderate collaboration is required (determined by coupling strengths) coupling are suspended (removed) and a 'block buffer' is added after the last activity in the block investigated. Buffer load is set equal to a portion of the suspended coupling load and another percent is added to the project buffer. An equivalent DSM representation of an I-pattern, C-Pattern, or an L-pattern can be determined according to the cases shown in Tables 6, 7, and 8 respectively. An equivalent DSM for a C4-pattern and an S-coupling are shown in Fig.38, and Fig. 39 respectively.

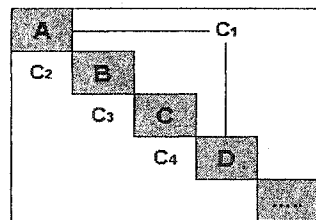
It should be mentioned here that the philosophy of suspending weak feedback couplings was also adopted by both ADePT [35] and DeMAID/GA [39], but neither methodology suggested rules of compensation for the suspended iteration time.



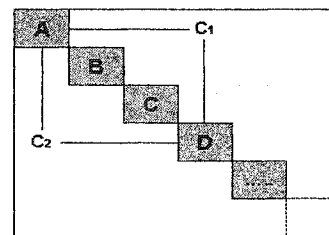
(a) I-Pattern



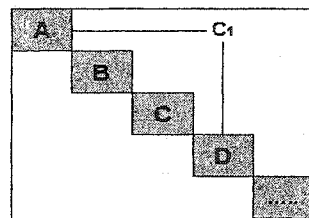
(b) C-Pattern



(c) C4-Pattern



(d) L-Pattern



(e) S-Coupling

Figure 37. Different Patterns.

Table 6. I-Pattern Conversion Rules.

Case No.	Coupling Strength		Equivalent DSM Representation	Add to Project Buffer
	C <sub>1</sub>	C <sub>2</sub>		
1	≤ 3	≤ 3		0.2 load of C <sub>1</sub>
2	≤ 3	> 3		0.3 load of C <sub>1</sub>
3	≥ 6		<p>Block load = 0.5 load of C<sub>1</sub></p>	0.2 load of C <sub>1</sub>

Table 7. C-Pattern Conversion Rules.

Case No.	Coupling Strength			Equivalent DSM Representation	Add to Project Buffer
	C <sub>1</sub>	C <sub>2</sub>	C <sub>3</sub>		
1	≤ 3	≤ 6	≤ 6		0.2 load of C <sub>1</sub>
2	≤ 3	> 6	> 6		0.3 load of C <sub>1</sub>
3	> 3	< 6	< 6		0.4 load of C <sub>1</sub>
4		≥ 6		<p>Block load = 0.5 load of C<sub>1</sub></p>	0.2 load of C <sub>1</sub>



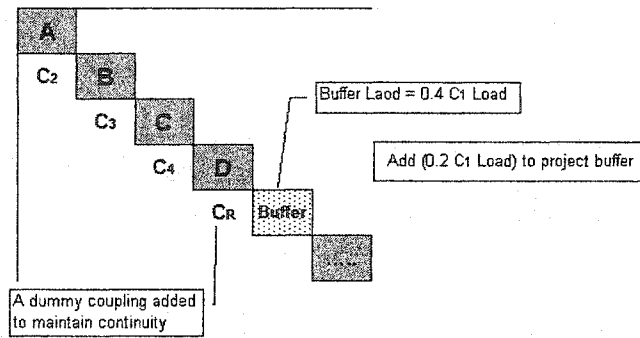


Figure 38. Equivalent DSM for C4-Pattern.

Table 8. L-Pattern Conversion Rules.

Case No.	Coupling Strength		Equivalent DSM Representation	Add to	
	C <sub>1</sub>	C <sub>2</sub>		Block Buffer	Project Buffer
1	≤ 3	≤ 3			0.2 load of C <sub>1</sub>
2	≥ 6	≥ 6		0.4 load of C <sub>1</sub>	0.3 load of C <sub>1</sub>
3	≥ 6	< 6		0.4 load of C <sub>1</sub>	0.2 load of C <sub>1</sub>
4	Other			0.3 load of C <sub>1</sub>	0.3 load of C <sub>1</sub>

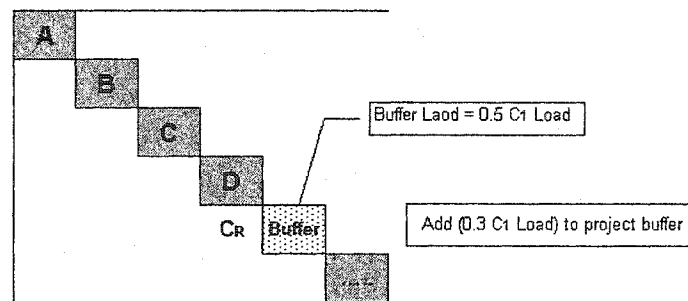


Figure 39. Equivalent DSM for S-Coupling.

#### D. Rules of Conversion

The following are the steps to be followed in order to define an equivalent DSM without feedback couplings.

1. Determine a set of unsolved feedback couplings.
2. Arrange these couplings according to the following order:
  - a. I-pattern
  - b. C-pattern
  - c. C4-pattern
  - d. L -pattern
  - e. S -coupling
3. Ties are broken according to the order of activities; first comes first.
4. Consider couplings, in order, and perform the following:
  - a. Apply conversion rules defined previously
  - b. Update DSM; activities, couplings
  - c. If a buffer is added:

- i. A dummy feed forward coupling with nominal strength is added between that buffer and preceding activity.
  - ii. Any two (or more) consecutive buffers are merged into one buffer with an associated load equals to the total load of these buffers.
  - iii. Buffer load is determined based on the original couplings loads and the original final DSM, not on the DSM resulted from previous step.
5. Unsolved patterns in the set are handled according to the procedures regardless of any modification occurs due to a previous step (i.e. merging of two activities, or adding a buffer).

Following these rules, an equivalent program of the DSM (with no feedback couplings) is generated. Fig. 40 shows the equivalent project schedule for the comparison case (Section 10.1) according to the presented methodology.

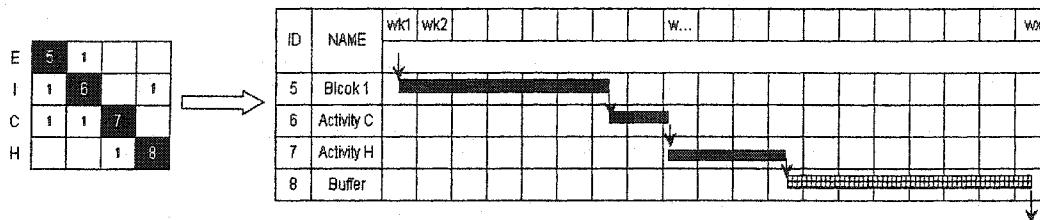


Figure 40. Equivalent Schedule for the Comparison Case According to the Proposed Methodology.

## 6.11 Summary

Since its original introduction by Steward in the 1980's, the DSM has proved itself as an effective tool for analyzing and understanding system architecture especially in product development and, hence, achieving improved performance.

The use of the DSM is the corner stone of the architecture proposed in this dissertation. Following the modeling of the design project in the form of a DSM, this chapter provided in-depth discussions on there of current research contributions:

1. Determining the optimal sequence of DSM activities in either two cases:
  - a. Deterministic activity data
  - b. Stochastic activity data
2. Interfacing simulated annealing with Monte Carlo simulation to handle the second case. The interface required modifying the SA algorithm.
3. Conversion of the optimally re-sequenced DSM into an equivalent DSM with no feedback loops.

## CHAPTER VII

### TCTO HYBRID MODEL

#### 7.1 Introduction

In this chapter, a new time-cost trade-off model is presented. The presented model helps crashing a project with little, or no, additional cost. It is based on the trade-off of resources where, in some cases, it may be possible to transfer men, equipment, or other resources from a non-critical activity to a critical one. So, where large float times are available, cheaper but slower resources can be substituted for those originally planned. Trade-offs of this kind tend to expedite some of the activities on the critical path to save time, and, unfortunately, increase total cost, in addition to relaxing some non-critical activities to reduce total cost.

#### 7.2 Basic Concepts

The concepts on which the model relies on are:

1. Formulation of activity duration as a function of resources assigned to it.
2. Expediting critical activities by assigning high-productivity resources, while relaxing non-critical activities by assigning cheaper low-productivity resource. Thus, savings from relaxed activities can compensate for additional costs required for crashing other activities.

The following simple hypothetical example clarifies the first concept:

Among resources available in a product development firm (or a design firm) is *design engineer*. Assume that the firm has three levels of design engineers based on experience. These are: beginner, experienced, and professional. Each of these levels has its own *productivity* and *hourly cost* as shown in Fig. 41. Of course, as productivity increases, increased hourly cost can be expected.

Resource Type: Design Engineer




	Level	Productivity (load units/hr)	Cost (\$/hr)
	Beginner	5	3
	Experienced	8	5
	Professional	14	10

Figure 41. Resource Levels.

Given an activity with load equals to 1120 units, where load might be a design assignment, any combination of the former resource levels can be assigned to this activity. Thus, given that three levels are available and that only two engineers are available from each level 129 possible combinations (both feasible and infeasible) can be chosen from. Figure 42 shows some of these combinations. As shown, each combination results in a certain activity duration and cost. The general problem, thus, is how to assign the proper resource combination for each activity in the project such that:

1. Project is completed on a given due date,
2. With available resources, and
3. With minimum cost.

The next example provides a more elaborated discussion on the problem.





 1120 load units	Resource Assignment	Case 1	Case 2	Case 3	Case ...
					
Activity Duration (hr)		112	87	80	
Activity Cost (\$)		672	696	800	

Figure 42. Different Assignments Result in Different Activity Durations.

### 7.2.1 Illustration of the Basic Concept

Consider the simple hypothetical project network shown in Fig. 43 consisting of five activities: A, B, C, D, and E. Each activity in the network is associated with a certain load (shown on figure next to activity name) that needs to be done in order to realize the activity.

Assuming that all activities require the same resource type and that there exists three levels of this resource: *fast*, *normal*, and *slow* each associated with a certain productivity rate and hourly cost as shown in Table 9.

As a start, each activity is assigned one normal-level resource. Such assignment will result in project completion time of 30 hours and total cost of \$500. Gantt chart shown in Fig. 44 presents project schedule for the initial assignment and corresponding cost calculations are shown in Table 10.

The critical path, which is the longest path, is simply B-E. Thus, to crash the project, activities on that path need to be expedited. To help clarify the procedure, another form of the Gantt chart is shown in Fig. 45 in which each of the three paths in the network is presented independently. In Fig. 46, the slack amounts associated with activities D and C are shown. For the sake of illustration, it is assumed that the project needs to be crashed by four hours. This can be achieved by expediting activity E through assigning the fast-level resource to it instead of the normal-level one. Figure 47 shows the project after

crashing. A subsequent result of project crashing, of course, is the increase of project cost by \$24 to be \$524.

A second look to Fig. 46 suggests that activity D is not on the critical path and has a large slack. Or, in other words, activity D can be relaxed (to a certain extent) without affecting the project completion time. So, a slow-level resource can be assigned to it instead of the normal-level resource. The new configuration (shown in Fig. 48) now results in a crashed project of 26 hours with lower additional cost; \$513 instead of \$524 – cost calculations shown in Table 11.

Thus, the concept proposed is:

*“While higher-productivity (thus more expensive) resources are assigned to critical activities to expedite the projects, lower-productivity (and cheaper) resources can be assigned to non-critical activities.”*

This would serve in two directions:

1. Reducing the additional cost associated with project crashing.
2. Lowering the demand on high-productivity resources.

It should be noticed, though, that in some assigning a faster resource to an activity might reduce the total cost of the project and shorten its completion time too. For example, if the cost of the fast-level resource in the example was \$12/hr, the cost of crashed activity D would have been \$192 which is lower by \$8 than using a normal-level resource.



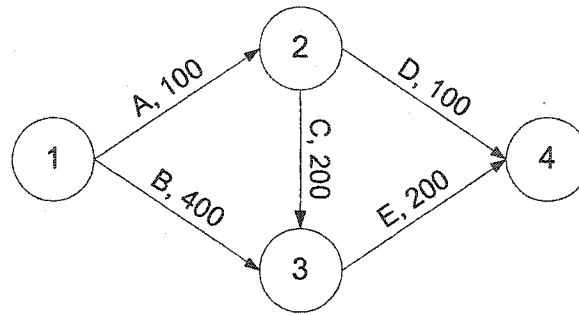


Figure 43. Project Network.

Table 9. Resource-Levels.

Resource-Level	Productivity (units/hr)	Cost (\$/hr)
Fast	25	14
Normal	20	10
Slow	8	3

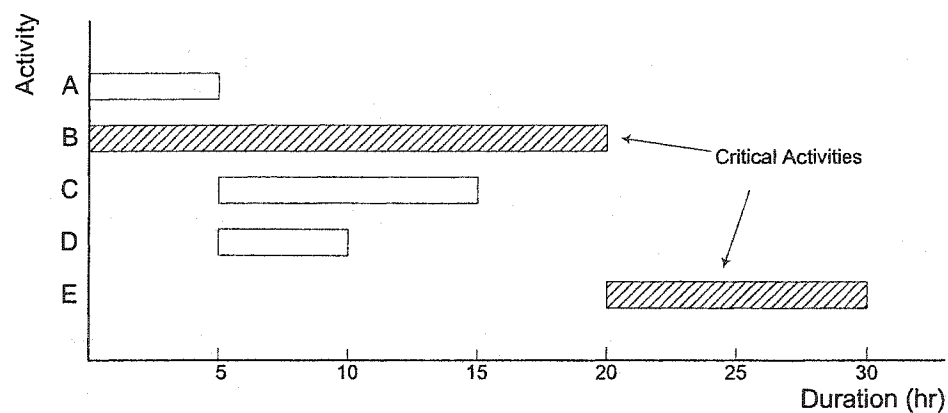


Figure 44. Project Schedule Corresponding to Initial Assignment.

Table 10. Cost Calculations for the Initial Configuration (Assignment).

Activity	Resource-Level	Duration (hr)	Cost (\$)
A	Normal	5	50
B	Normal	20	200
C	Normal	10	100
D	Normal	5	50
E	Normal	10	100

$$Duration = \frac{load}{total\ resource\ productivity}$$

$$= \frac{100}{1 \times 20} = 5$$

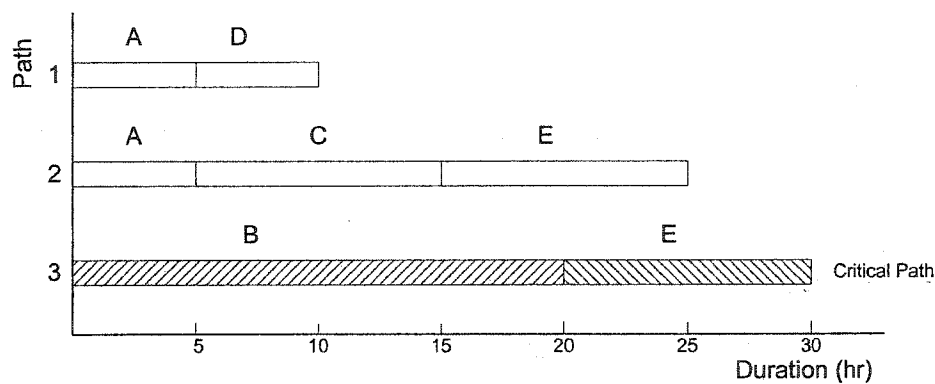


Figure 45. Gantt Chart Showing the Three Paths.

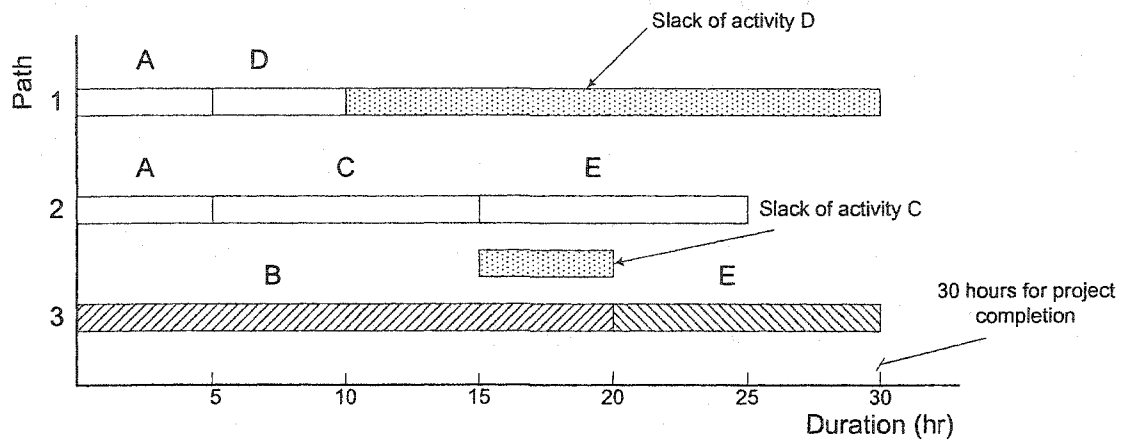


Figure 46. Activities' Slacks.

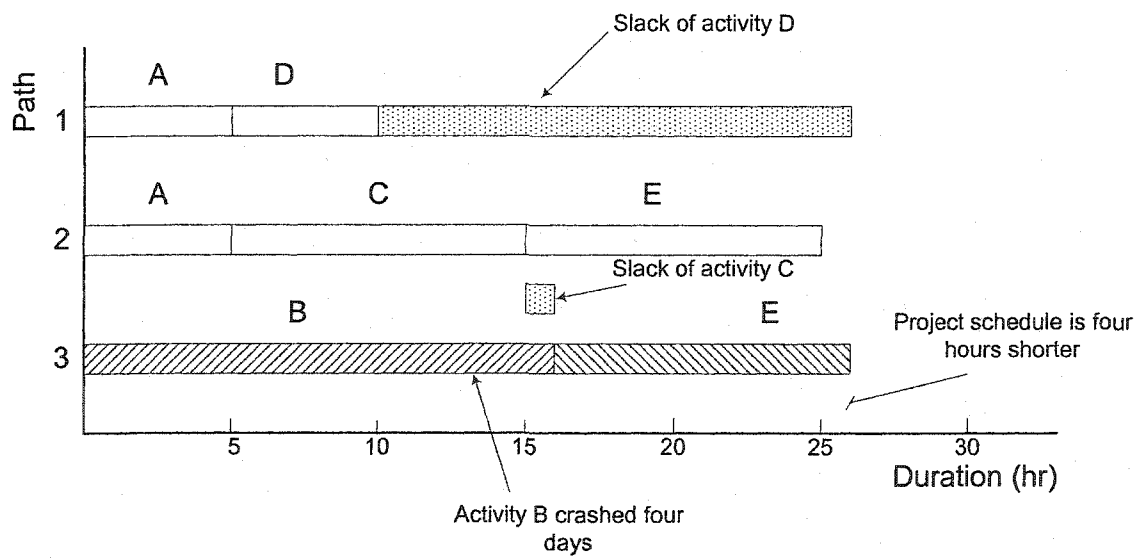


Figure 47. Crashing of Activity E.

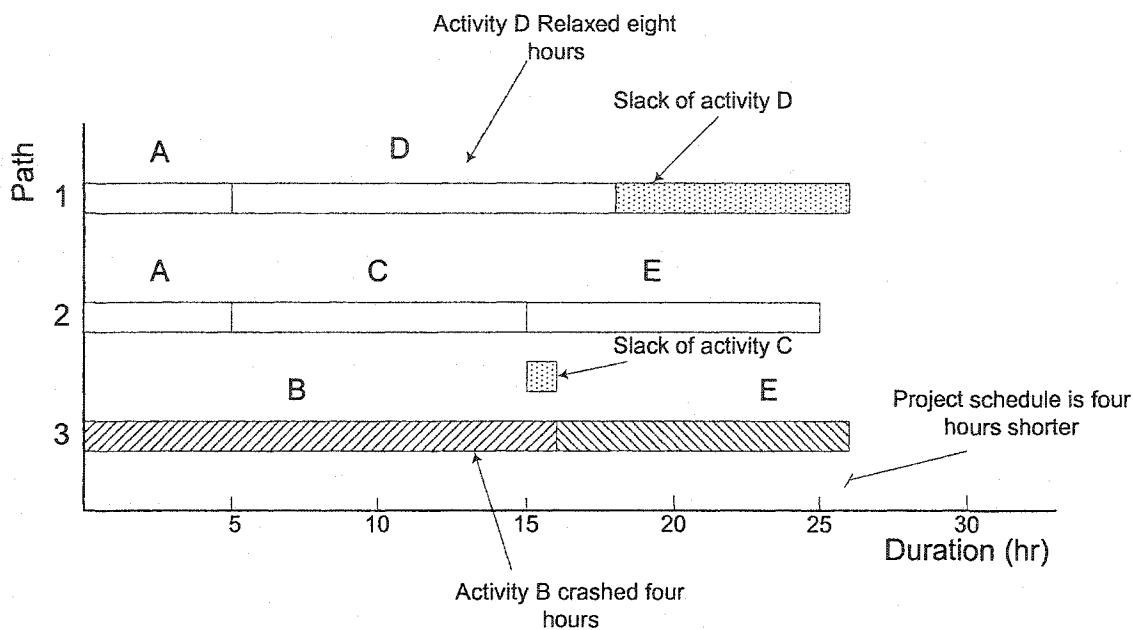


Figure 48. Relaxing of Activity D.

Table 11. Cost Calculations for the Final Configuration (Assignment).

Activity	Resource-Level	Duration (hr)	Cost (\$)
A	Normal	5	50
B	Fast	16	196
C	Normal	10	100
D	Slow	13	39
E	Normal	10	100

$$Duration = \frac{100}{1 \cdot 8} = 12.5 \Rightarrow 13$$

### 7.3 The Proposed Model

The model presented here is a nonlinear-integer programming model. On one hand, the model is integer because the duration of an activity, a function of both its load and the combination of resources assigned to it is assumed to be in whole time units. And, on the other hand, the model is nonlinear since the duration is being calculated using a nonlinear function.

#### 7.3.1 Problem Statement

In the proposed model, the project crashing or time-cost trade-off problem is being tackled from another point of view that tends to be more practical and suitable for design and manufacturing applications. The model tries to merge the known CPM calculations with the assignment problem. For a given project the followings apply:

1. The project is decomposed to its smallest component and each component is represented by an activity.
2. The specification of the project is assumed given in activity-on-arc notation (AoA); a set of activities to be completed according to certain precedence relationships.
3. Each activity has a “load”, e.g. processing/analysis time, that must be performed in order to realize the activity.
4. Different types of resources are available.
5. Each resource has a number of levels each associated with its own production rate and hourly cost.
6. An activity requires only one resource type (depending on the activity’s nature), but a combination of that resource levels can be assigned to the activity.
7. The duration of an activity is, thus, a discrete function of the number of resource levels assigned to it. Or, in other words, an activity shall assume a set of different durations according to different combinations of resource levels assigned to it. And once assigned, there will be no change in resource assignment.

8. The total number of each type of resource level is assumed to be limited, in the sense that, once a resource unit is used (assigned), it cannot be used again in another activity.

So, the objective is:

*“To determine the most efficient (optimized) project schedule(s), by assigning a suitable combination of resources to different activities, for a given project completion time while maintaining minimum cost.”*

### 7.3.2 Assumptions and Limitations

For the sake of simplification, the following assumptions apply to the formulated problem:

1. Looping and dangling of activities are not allowed.
2. Activities once started cannot be interrupted (activity splitting is not allowed).
3. Activity overlapping is not allowed. Thus, an activity cannot start until all its predecessors are completed.
4. An activity requires only one resource type.
5. The amount of load assigned to an activity is constant.
6. Number and productivity of each resource assigned to an activity remain constant throughout its duration.
7. Resources considered here are assumed to be non-renewable.
8. Activity loads, resource productivity, and resource cost are assumed deterministic.

### 7.3.3 Notations

The presented research uses an activity-on-arc notation; in which each activity is represented by an arc in the project network. The rest of symbols used are:

- $S$  : set of activities in the network
- $D_{ij}$  : duration of activity  $(i, j)$
- $load_{ij}$  : load of activity  $(i, j)$

- $m$  : number of nodes in the network,  
 $DPCT$  : desired project completion time  
 $k$  : resource type index,  $k = 1, \dots, K$   
 $K$  : total number of resources  
 $k_l$  : resource level index for resource type ( $k$ ),  $l = 1, \dots, L$   
 $k_L$  : total number of resource levels for resource type ( $k$ )  
 $A_{k_l}$  : available total number of resource level ( $k_l$ )  
 $C_{k_l}$  : hourly cost of resource level ( $k_l$ )  
 $P_{k_l}$  : production rate of resource level ( $k_l$ ) in units/hour  
 $N_{k_l,ij}$  : number of resource level ( $k_l$ ) assigned to activity ( $i, j$ )

#### 7.3.4 Inputs

The initial inputs that are supplied to the model are as follows:

1. A finite set of activities ( $S$ ) representing the project network. Each activity is described by:
  - a. Its tail and head events (nodes), denoted to as ( $i$ ) and ( $j$ ) respectively.  
Thus an Activity will be described as ( $i, j$ ).
  - b. An associated activity load.
  - c. The required resource type.
2. A Finite set of resources with which activities are performed. The set includes:
  - a. Resource types.
  - b. Resource levels for each type, each level has:
    - i. Productivity.
    - ii. Cost rate.
    - iii. Availability.
3. A finite set of precedence constraints to be satisfied.
4. An objective: a desired project completion time.

### 7.3.5 Process

For a given desired project completion time, the proposed model tends to minimize the total cost of the project. In order to achieve this, the model works as follows:

1. A combination of suitable resource levels,  $N_{k_i,ij}$ , are assigned to each activity in the project.
2. Based on this assignment, activities' durations,  $D_{ij}$ , are calculated by dividing the "activity load" by "the total productivity rate of the resource(s) assigned to it" and rounding the duration to the next larger integer value.
3. Activity cost is determined by multiplying its duration by "the total cost rate of the resource level(s) assigned to it."
4. Finally, total project cost is calculated and compared to the current best solution objective.

### 7.3.6 Decision Variables

$x_i$  : the earliest event time of node  $i$ ,  $i = 1, 2, \dots, m-1$

$x_j$  : the earliest event time of node  $j$ ,  $j = 2, 3, \dots, m$

$N_{k_i,ij}$  : the number of resource level ( $k_i$ ) assigned to activity ( $i, j$ )

### 7.3.7 Constraints

Six sets of constraints are imposed on the model:

#### A. Network Constraints

This set of constraints is required to insure that an activity cannot be started until all its preceding activities have been realized. The set describes the structure of the network by specifying the precedence relationships among different network activities. Let  $i$  be the preceding event and  $j$  be the following event, then the precedence relationship constraint can be formulated as:



$$x_j - x_i - D_{ij} \geq 0 \quad (19)$$

The number of such constraints is equal to the number of activities in the network.

### B. Assignment Feasibility

Since the optimization algorithm used to solve the model might tend to assign zero resources for activities to reach a minimum total cost, this constraint set is introduced to guarantee that at least one resource unit (of any level) is assigned to each activity.

$$\sum_{k_i} N_{k_i,ij} \geq 1 \quad \forall s_{ij} \in S \quad (20)$$

### C. Resource Availability Constraints

This set of constraints represents the available number of units available for each resource.

$$\sum_{k_i} N_{k_i,ij} \leq A_{k_i} \quad \forall s_{ij} \in S \quad (21)$$

### D. Project Completion Constraint

This constraint insures that the project schedule was set in such a way that the project would be completed within the desired time span. This is done by specifying that the last event must take place before the desired project schedule deadline date.

$$x_m \leq DPCT \quad (22)$$

### E. Activity Duration Constraints

To satisfy the assumption of integer durations and integer number of resource to be assigned/used, another set of constraints was added that forces the optimization algorithm to have integer values for these variables.

$$N_{k_i,ij} \text{ and } D_{ij} \text{ are integers} \quad (23)$$

#### F. Non-negativity Constraints

Finally, all variables, of course, cannot be assigned negative values.

$$x_i, x_j, N_{k_i,ij}, D_{ij} \geq 0 \quad \forall s_{ij} \in S \quad (24)$$

#### 7.3.8 Objective Function

To minimize the total cost of the project;  $Z$

$$Z = \left( \sum_{s_{ij}} \sum_{k_i} N_{k_i,ij} C_{l_k} \right) D_{ij} \quad (25)$$

where

$$D_{ij} = \frac{\text{load}_{ij}}{\sum_{k_i} P_{k_i} N_{k_i,ij}} \quad (26)$$

and  $D_{ij}$  is rounded to the next higher integer number

#### 7.4 The model: How is it different from the classical model?

Herring & Murphy [222] quoted:

When using CPM procedure, the time-cost trade-off points are assumed to lie on a continuously linear, or piece-wise linear, decreasing convex curve to insure an optimal solution. Further, all activities are assumed independent, in the sense that buying time on one activity does not affect in anyway the availability, cost, or need to buy time in some other activity.

The presented model, thus, differs from classical CPM time-cost trade-off model in five major aspects:

1. As shown in Fig. 49, activity possible durations are represented by separate points not by a continuous straight line.
2. Activities are not independent, since assigning a resource to an activity affects the availability of this resource with respect to other activities.
3. The procedure itself differs from the classical crashing procedure in the sense that it crashes and relaxes and not just crashes activities.
4. In classical/traditional CPM calculations, time and/or cost estimates are assigned to each activity at the start of the analysis. In the proposed model, activity durations are calculated during the solution procedure and are changing from one trial to the other based on the assignment of different resources to different activities.
5. The proposed model is a hybrid model because:
  - a. It is an assignment problem in the sense that:
    - i. It involves determining the most efficient (optimum) assignment of resources to different activities.
    - ii. A resource unit can be assigned to one activity only.
    - iii. The objective is to minimize the total cost while maintaining a specified project completion time.
    - iv. While assignment problems are classified as linear programming problems, the case here is different; it is a nonlinear/ integer programming problem.
    - v. A mathematical programming problem is considered integer when one, or more, of the decision variables has to take on an integer value in the final solution. Furthermore, the proposed model is considered a “pure IP” since all decision variables must have integer solutions.
  - b. It is a project management problem in the sense that:
    - i. A project goal (manufacturing of a certain product) is specified.
    - ii. All project activities are defined.

- iii. A certain precedence relationships profile among these activities exists.

Finally, it should be noted here that the presented model is an NP-hard problem, meaning that for even moderate-sized problems finding an optimal solution is a difficult task due to the exponential size of the solution space. The non-linearity of the objective function, with many expected local optima further adds more complexity to the problem. Moreover, another complexity factor is the integer constraints.

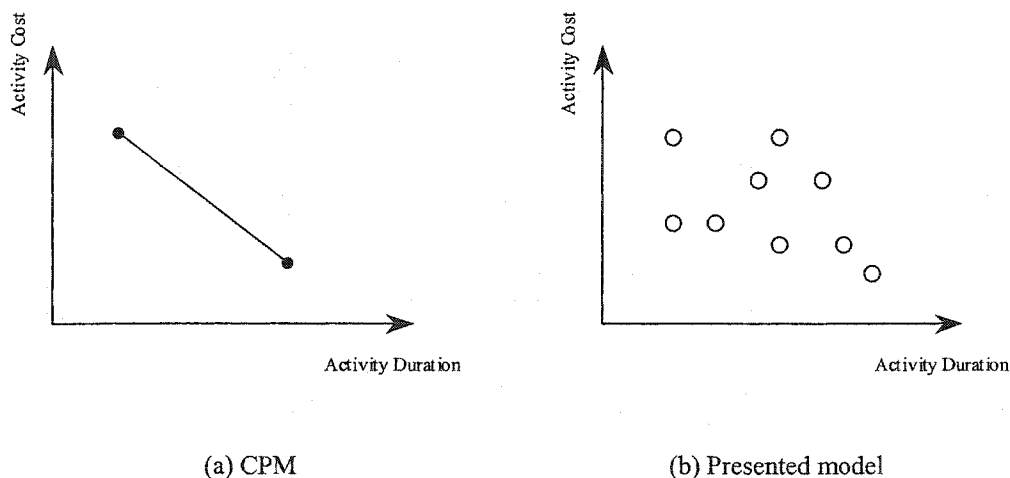


Figure 49. Models comparison.

### 7.5 Solution: Optimization Methods Implemented

In order to solve the presented mathematical model, an Excel sheet that costs the variables and equations of the model was created. Several commercial optimization add-ins were implemented, these are:

1. What's Best!– developed by LINDO systems Inc.
2. Evolver – developed by Palisade.
3. optQuest – developed by Decisioneering Inc.

4. Premium solver platform – developed by Front Line Systems.

None of these tools managed to reach even a feasible solution.

More investigation was performed on using “What’s Best!” by developing a VBA macro that provides WB! with an initial feasible solution many times in order to exhaust the solution space as possible, and hence, gets the global optimum. Figure 50. shows the flow diagram of the macro. The main goal of the macro was to automatically assign random, but reasonable, starting values for the decision variable. In case of non-feasible solution, the macro re-runs What’sBest! again and again until it reaches an optimal solution. This scenario is repeated until 20 local optimal solutions are to be found, and then the minimum one of them is chosen as the global optimum solution. But, unfortunately, experimentation shows that WB! was unable to move from these initial feasible solutions, and rather consider these solutions the optimal ones! Thus, the decision was made to implement simulated annealing instead.

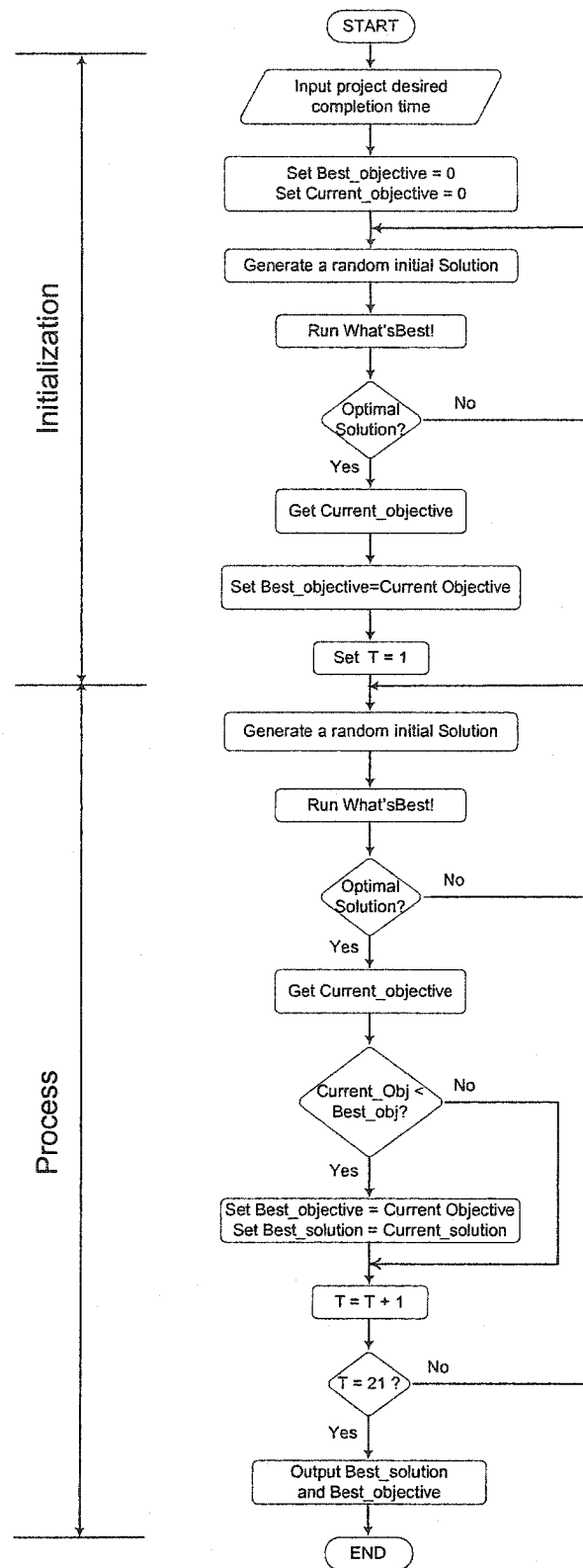


Figure 50. TCTO Optimization Macro Flowchart.

### 7.5.1 Simulated Annealing

SA proved to be a powerful optimization algorithm when used in the DSM optimization part. In this section, the two-stage SA discussed earlier in Chapter VI will be used to solve the presented TCTO model. The followings are implementation details.

#### A. Decision Variables and Solution Representation

The solution representation, as shown in Fig. 51, is an  $m \times n$  matrix where the number of rows ( $m$ ) equals the number of activities, and the number of columns ( $n$ ) is the total number of all resources levels. The decision variables are the number of resource units of a certain type and level assigned to each activity.

Decision Variables

Activities	Resource Levels										
	$1_1$	$1_2$	...	$1_l$	$2_1$	$2_2$	...	...	...	...	$k_l$
$s_{12}$	$N_{1_1,12}$	$N_{1_2,12}$	$N_{1_3,12}$	$N_{1_l,12}$							
$s_{13}$					$N_{2_1,13}$	$N_{2_2,13}$					
...											
...											
...											

Figure 51. Solution Representation.

### B. Generation of the Initial Solution Configuration

An initial solution configuration is generated by randomly assigning resource units to each activity. All constraint sets are examined and the initial solution has to satisfy all constraints, including project duration, to be considered feasible.

### C. Generation of a Feasible Neighboring Solution Configuration

From a current solution configuration, a neighboring solution is generated as follows:

1. Randomly choose a decision variable in the matrix.
2. Randomly choose a perturbation value from the set  $\{-2, -1, 1, 2\}$ .
3. Change the decision variable value by the amount determined in step 2.

It should be noted here that the model is constrained. Thus, to accept a generated solution configuration all constraints have to be satisfied first before pursuing the rest of SA steps.

### D. Objective Function Evaluation

The objective function, which is the total project cost, is determined for each solution directly from Excel calculations.

### E. Cooling Schedule

A relatively fast annealing schedule is implemented, in which:  $T_0 = 80$ ,  $T_f = 10$ , and  $\alpha = 0.95$ .

### F. Stopping Criterion

The system is considered in a meta-stable condition when either of the number of rejected solutions or the number of accepted solution reaches 30. The optimization process is set to stop when the final temperature is reached. But, for the sake of shortening the optimization runs, the optimization process is terminated (assuming that the ground state has been reached) when the processing time at any temperature reaches a predefined threshold value.



## 7.6 Summary

In this chapter, a new time-cost trade-off model for project networks crashing was presented. The new model combines a traditional assignment problem with CPM calculations. It avoids the shortcoming of the traditional crashing problem, concerning its inapplicability to real life situations. While the classical approach tries to obtain the minimum cost associated with a desired project completion time by reducing the time of some activities, this approach, in addition to this objective, minimizes the cost through extending/relaxing non-critical activities, as possible, by increasing the use of the slow type- i.e. cheaper- resource. Moreover, the objective of the classical formulation is to reduce additional cost due to crashing, while in the proposed model the objective is to reduce overall cost of the project or, in other words, to get the minimum possible project total cost.

## CHAPTER VIII

### ARCHITECTURE AND PRODUCT

Figure 52 illustrates the architecture proposed in this dissertation. It consists of three major phases:

1. System decomposition,
2. DSM optimization and analysis, and
3. Project scheduling and time-cost trade-off analysis.

While the methods related to the first phase falls beyond the scope of this research, elaborated discussions on the methods and tools related to the second and the third phases were given throughout the dissertation. The current chapter describes in detail different modules of the architecture and the interactions among these modules.

#### 8.1 Architecture Overview

The structure of the proposed architecture can be further broken down into the following sequential modules:

1. Modeling,
2. Optimization,
3. Structuring,
4. Conversion,
5. Scheduling, and
6. Crashing (TCTO).

While modules one to four constitute the second phase (DSM optimization and analysis), modules five and six represent the third phase (time-cost trade-off). The following sections give a short description of each of these modules.

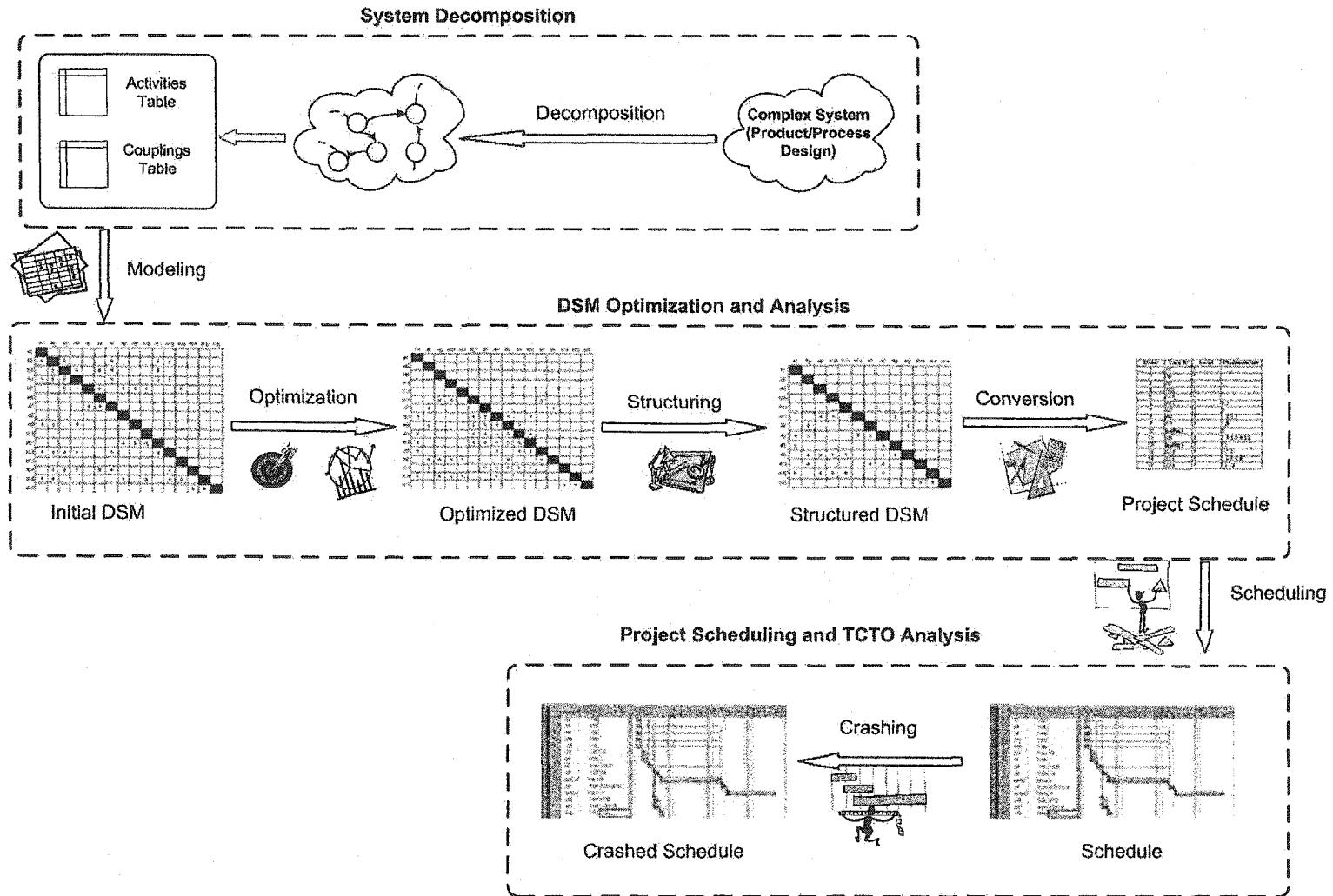


Figure 52. The Proposed Architecture.

### 8.1.1 Modeling

Given a list of activities, their associated loads, and information inputs and outputs from and to each activity (i.e. information couplings), the initial dependency structure matrix (DSM) which provides a compact visualization of the project and a clear understanding of the information flow patterns among different activities is created. The architecture deploys a numerical DSM; in which coupling marks are replaced by numbers (iteration factors). Refer to Section 6.6.

### 8.1.2 Optimization (Re-Sequencing)

As the order of the activities changes, the DSM structure (couplings' directions) changes. Thus, incorporating an optimization technique that is capable of re-sequencing the activities execution order will achieve a reduction in the estimated project total load in addition to an improved design quality. Since the tackled problem falls in the NP-hard class, a conventional calculus-based optimization technique wouldn't be an effective solution tool. Thus, as discussed in Section 6.8, a meta-heuristic algorithm called simulated annealing is implemented to rapidly evaluate many sequences and find the one that minimizes project total load (the objective function) while satisfying all imposed constraints (both precedence and logical).

Thus, the objective of this module is to find the optimum activity sequence based on one of the following objective functions:

1. Total number of feedbacks.
2. Project total iterative load.
3. Project total iterative time.
4. Project total iterative time and cost.

As hinted previously, the objective function calculates the total load (or time and cost) resulted from feedbacks loops not the total project time and cost (refer to Section 6.7). Furthermore, the fourth objective function option represents a multi-objective optimization problem.

The module tolerates stochastic activity loads (presented by a probability distribution not by one-point estimate) by integrating a commercial risk analysis tool (Crystal Ball™) to construct a simulation-based optimization framework. Thus, the optimization module performs one of the following techniques:

- A. In the deterministic case, simulated annealing will be deployed.
- B. In the stochastic case, an integrated simulation-optimization framework - i.e. SA interfaced with Crystal Ball™ - will be deployed. This framework will be presented in Section 2 of the current chapter.

It should be noted here that while an unconstrained optimization is the default case, a constrained optimization problem emerges in cases where logical constraints are imposed. Figure 53 presents the classification of different possible optimization situations solved by the module.

### **8.1.3 Structuring**

Following module 2, the optimally sequenced DSM is to be re-structured according to rules discussed earlier in Section 6.10.2. Typically, this includes patterns recognition, collapsing, and tearing. The objective of the module is to produce an equivalent DSM that contains no feedback couplings

### **8.1.4 Conversion to a Project Schedule**

The equivalent DSM resulted from the structuring module is used to construct a project schedule - a list of activities, their loads, buffers, and precedence relationships – without feedback couplings.

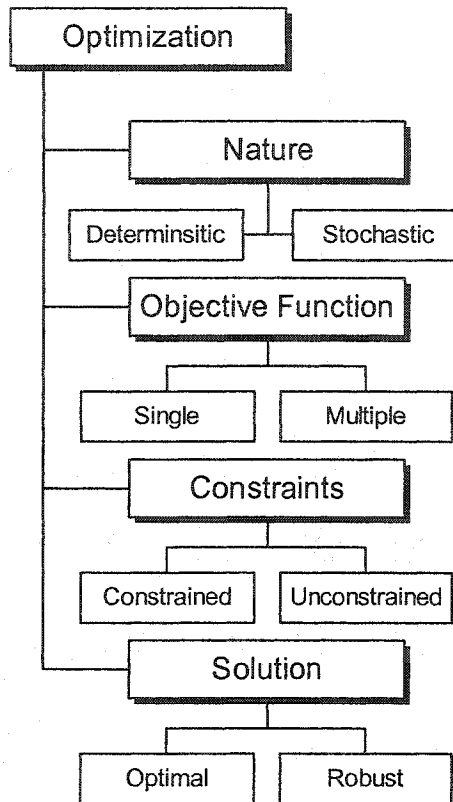


Figure 53. Optimization Cases.

### 8.1.5 Scheduling

The resulted project schedule is, then, transferred to a project management software (MS Project) where it is presented using both Gantt chart and a network diagram.

### 8.1.6 Crashing

Finally, as detailed in Chapter VII, resources are assigned to project activities to meet a specified due date.

## 8.2 Product

### 8.2.1 DSM Optimization and Analysis

A software called optDSM has been developed as an implementation part of this dissertation. optDSM is an Excel add-in that aimed towards carrying out the first phase of the framework. To achieve its goal, the tool integrates:

1. Visual Basic for Applications (VBA).
2. Mathematical programming (modeling).
3. Simulated annealing.
4. Commercial software of risk analysis (Crystal Ball™).

The construction and different functions of optDSM are presented in Appendix A.

The research utilized VBA programming language to develop optDSM infrastructure and different functions. VBA (Microsoft Visual Basic® for Applications) is a powerful development technology. The use of VBA in this research allowed customization and integration of off-the shelf software (MS Excel, Crystal Ball, and MS Project) to serve the research objectives rather than developing the whole solution architecture from scratch. For more details on VBA refer to [223].

The corner stone of optDSM is MS Excel, which is characterized by: (1) its great range of built-in mathematical and scientific functions, (2) availability of logical statements and decisions, (3) its capability to communicate with other applications within MS Windows environment, and (4) the availability of different add-ins that seamlessly add evermore functionality to MS Excel. Thus, a mathematical model for the DSM calculations is developed through an Excel spreadsheet.

While simulated annealing (SA) was used as the optimization technique for deterministic cases, a simulation-based optimization framework (in which SA is interfaced with Crystal Ball) was needed to handle stochastic cases. The flowchart presented in Fig. 54 illustrates

the structural operability of this framework, which constitutes a major component of optDSM.

The process of this framework goes as follows. Starting with some initial solution configuration, the following process is repeated. At each iteration, the *objective function evaluation* module receives a new solution configuration from the *optimizer*. The *simulation* of the model takes place with this configuration. *Expected* value of the objective function is obtained directly from *Crystal Ball<sup>TM</sup>*. This value is fed to the *optimizer* which return a new solution configuration. A new iteration starts. The process proceeds until the *simulated annealing* stopping criterion is reached.



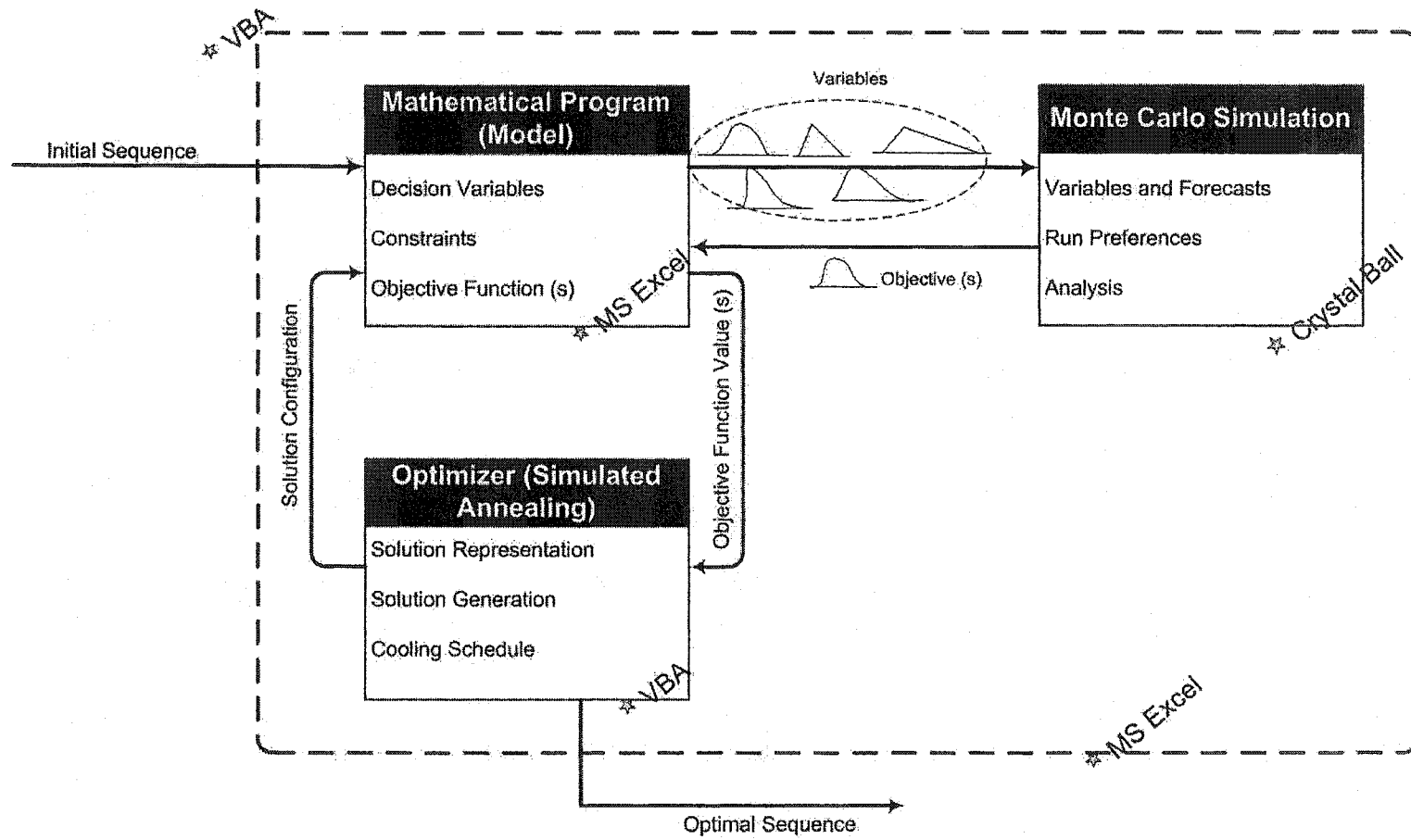


Figure 54. Simulation-based Optimization Framework.

### 8.2.2 Time-Cost Trade-Off

The mathematical model formulated in Chapter VII is implemented through an Excel sheet. To perform the optimization process, a VBA macro, shown in Fig. 55, that implements simulated annealing was created. The process starts by generating a random feasible starting solution. Then the following is repeated: at each iteration, the *assignment module* in the Excel sheet receives a new solution configuration from the *optimization module*; the feasibility of the proposed solution is checked; if the solution is infeasible, it is rejected; if the solution is feasible, time and cost calculations of the model take place with this configuration; precedence and project completion constraints are checked for feasibility; if feasible, the value of the objective function is fed to the *optimizer* which return a new solution configuration; a new iteration starts. The process proceeds until the *simulated annealing* stopping criterion is reached.

Due to the complicated non-linearity, in addition to integer constraints, that exists in the model. Thus, despite of the powerful capabilities of SA, the macro was designed to:

1. Perform several optimization runs, each with a different initial feasible solution
2. Reject any non-feasible solutions generated.

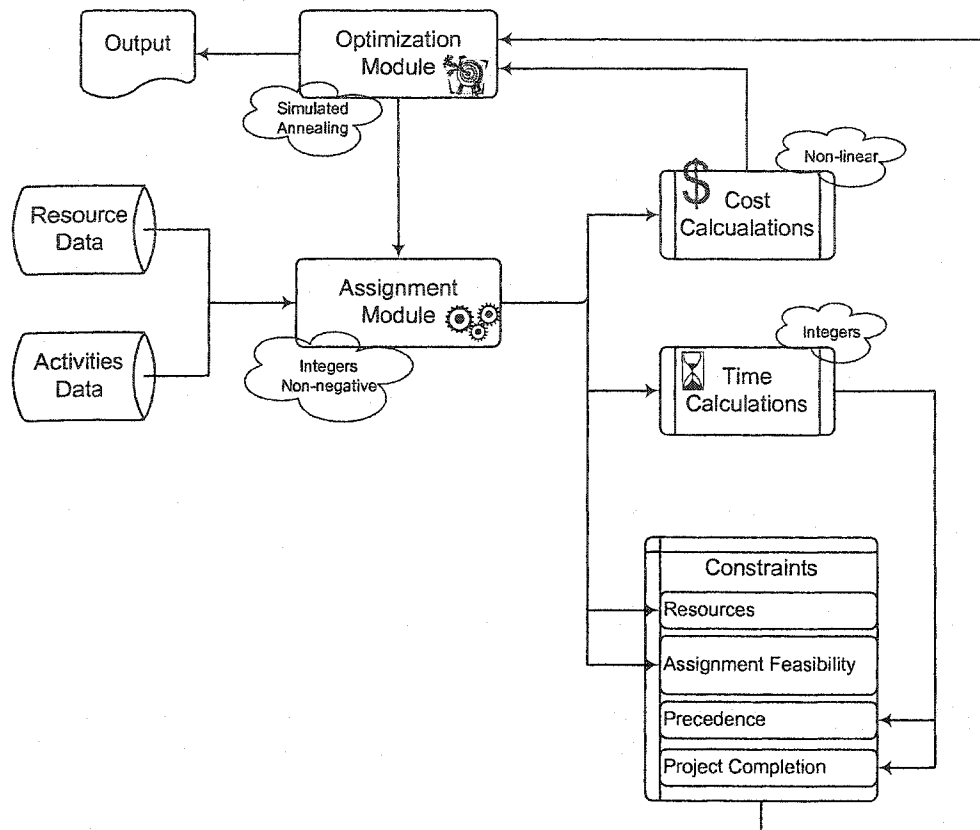


Figure 55. TCTO Module.

#### 8.4. Summary

The main objective of the presented research is to develop an optimization-based architecture that helps guiding the project manager efforts for managing the design process in complex integrated product development projects. The current chapter described in details different modules of the architecture and the interactions among these modules. While table 12 summarizes the two main phases of the architecture, their functions, the conceptual tools used and the implementation software, Table 13 presents different options of the architecture.

Table 12. Summary of Architecture Functions.

Phase	DSM Optimization and Analysis			Time-Cost Trade-Off		
	Function	Conceptual Tools	Implementation Software	Function	Conceptual Tools	Implementation Software
Details	Project Representation	DSM	MS Excel	Mathematical Modeling		MS Excel
	Mathematical Modeling		MS Excel	Optimization	SA	VBA
	Optimization	SA	VBA	Application Integration		VBA
	Uncertainty Analysis	Monte Carlo Simulation	Crystal Ball			
	Stochastic Optimization	Simulation- based Optimization	VBA			
	Application Integration		VBA			

Table 13. Summary of Architecture Options.

		Option #	1	2	3	4	5	6	7	
DSM Optimization and Analysis	INPUTS	Tasks	Load (S)	Load (D)	Time (S)	Time (D)	Time & Cost (S)	Time & Cost (D)		
		Information Flow	Resource types, levels, productivities, cost rates, and availability		Couplings Couplings Strength (optional) Logical Constraints (optional)					
	PROCESS		Quantify couplings strength							
		Optimization (Minimize)	Load OR Feedbacks		Time Or Feedbacks		Time Or Time & Cost Or Feedbacks		Feedbacks	
Simulation-based		Yes	No	Yes	No	Yes	No	No		
TCIO	OUTPUTS	DSM Structuring	Yes (for proceeding)		Optional					
		Conv. To Program								
		Resource Assignment	Yes		No					
		Project Schedule (crashed at minimum cost)	Optimum DSM (minimum feedbacks, time, and/or cost) Analyzed DSM (concurrency, sequence) Gantt chart (Project schedule)							

D: Deterministic  
S: Stochastic

## CHAPTER IX

### CASE STUDIES AND RESULTS

#### 9.1 Project 1: Benchmarking

To test the performance of optDSM and to compare it with two other tools (AGENDA and DeMAID) optDSM was applied to the VTOL analysis code used by Kopra et al. [224].

While DeMAID has no explicit objective, AGENDA's objective is to reduce the 'total length of feedback' of the system. The published DeMAID solution has a total feedback length of 275, while the solution obtained by AGENDA, shown in Fig. 56, has a total feedback length of 133 [47].

In optDSM terms, AGENDA's solution has 33 feedbacks, and 167 load units - assuming that one load unit is assigned to each activity. When optDSM was used to solve the problem, it reached an equivalent solution (33 feedbacks and 167 load units) shown in Fig. 57. Due to the difficulty of the DSM optimization problem since many local optimum solutions may exist, the sequence obtained by optDSM was slightly different than the one obtained by AGENDA (comparison shown in Table 14).

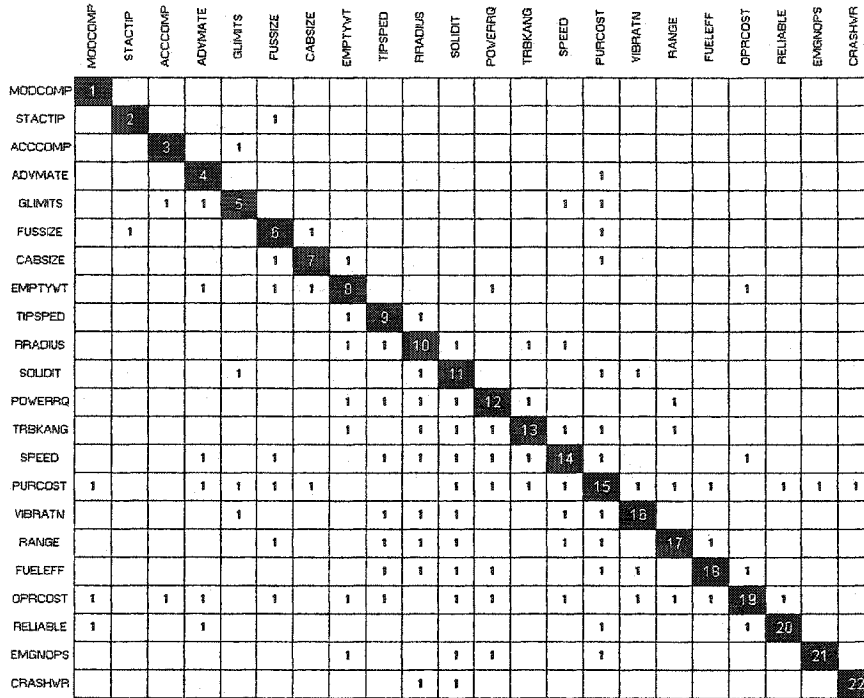


Figure 56. Solution Obtained by AGENDA.

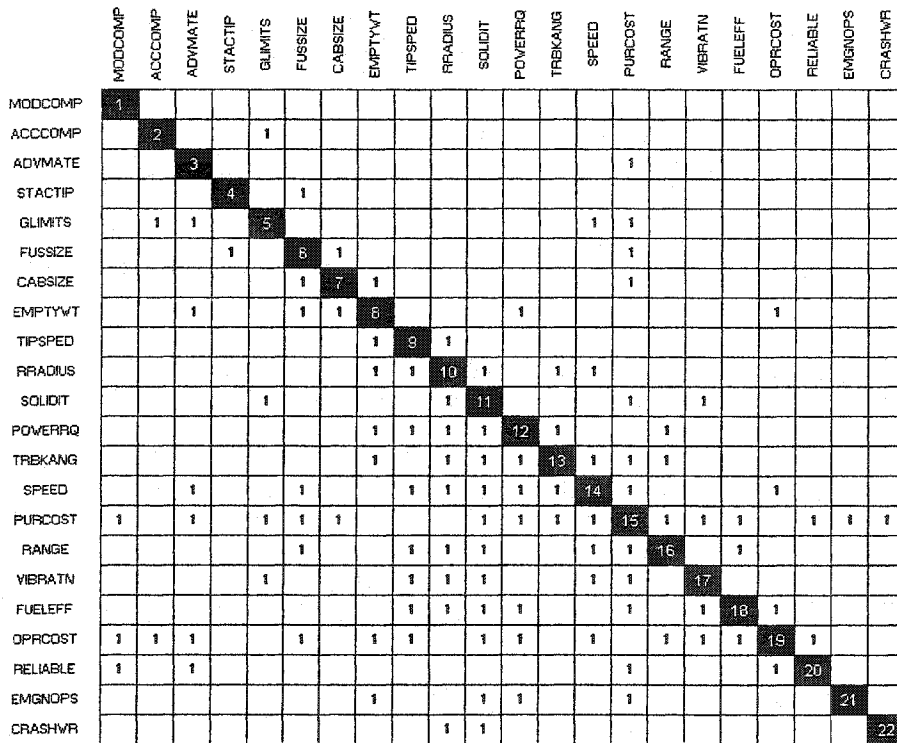


Figure 57. Solution Obtained by optDSM.

Table 14. optDSM Sequence vs. AGENDA Sequence.

Task	Order	
	AGENDA	optDSM
MODCOMP	1	1
STACTIP	2	4
ACCCOMP	3	2
ADVMATE	4	3
GLIMITS	5	5
FUSSIZE	6	6
CABSIZE	7	7
EMPTYWT	8	8
TIPSPED	9	9
RRADIUS	10	10
SOLIDIT	11	11
POWERRQ	12	12
TRBKANG	13	13
SPEED	14	14
PURCOST	15	<b>15</b>
VIBRATN	16	<b>17</b>
RANGE	17	<b>16</b>
FUELEFF	18	18
OPRCOST	19	19
RELIABLE	20	20
EMGNOPS	21	21
CRASHWR	22	22

Numbers in **bold** represent different activity order.



## 9.2 Project 2: A Conceptual Design Project – Deterministic Case

The project presented here is adopted from Rogers et al. [42]. This project (consisting of 22 activities) was taken from a larger conceptual design project. Figure 58 represents the process flowchart for this project. “The main problems with this type of chart are that it is difficult to determine where to begin the design activity and which processes are iterative [39].”

As a start, the sequence of the activities has been randomly ordered. Table 15 shows different activities, their order, and an arbitrary time and cost (units depend on the user) associated with each of them. The DSM for this ordering, shown in Fig. 59, reveals the existence of 39 couplings (23 feedbacks and 16 feed forward). Coupling strengths are defined and then used to estimate the required number of iterations for convergence as discussed previously. Table 16 shows all couplings and their associated strength.

In the following sections, the DSM sequence will be optimized based on three different objective functions:

1. Case (1): Minimum feedbacks
2. Case (2): Minimum load
3. Case (3): Minimum total time and cost

It should be noted that the optimization of the current DSM is a relatively easier task than optimizing the DSM of project (1) since it has a lower complexity factor (1.77 compared to 4.99).

Despite of SA proven robustness, three different optimization runs will be performed for each case. These runs have different initial solution, initial temperature, and final temperature. Notice, though, that these runs are considered short runs in SA practice.

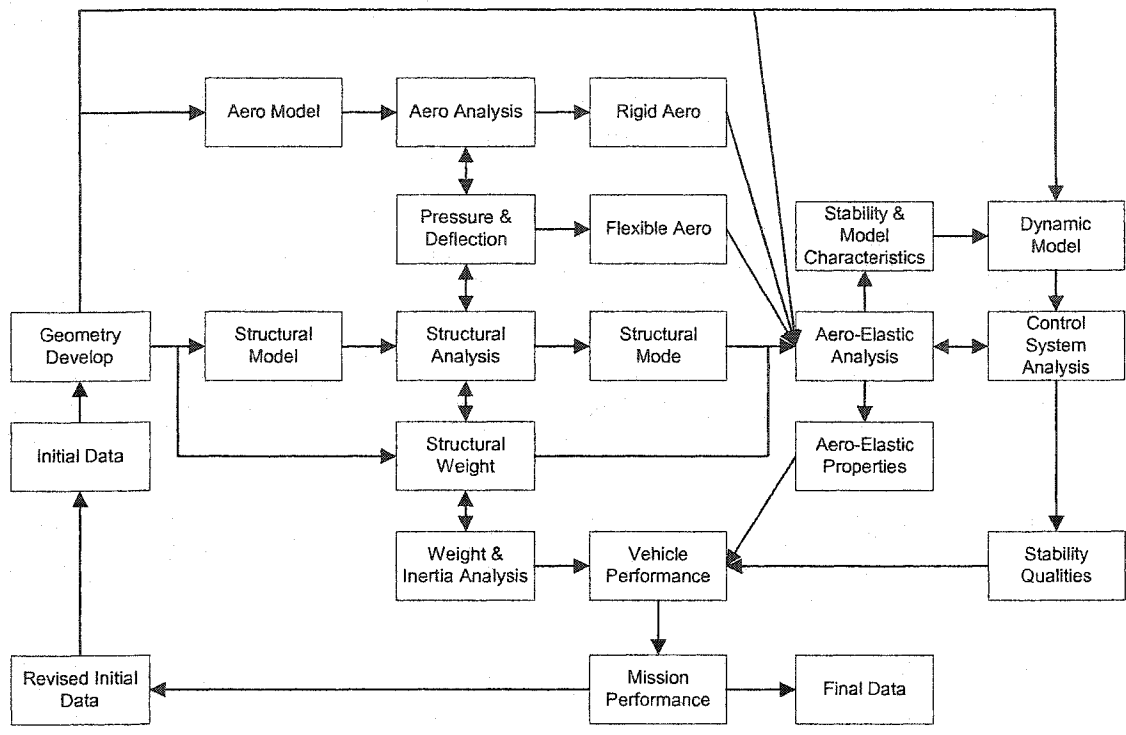


Figure 58. Process Flowchart.

	INITDAT	DYMMODL	STDMOCH	STRMOOL	HANDQUL	STRMODE	GEOMDEY	AOSRVO	STRDYNA	CSYSANAL	FAPERDCH	RVSEDAT	MISPERF	VEHEPERF	FAERDCH	AERDAIL	PRESEDF	STRANAL	STRCTVT	VIANAL	AERDMOL	FINLDAT		
INITDAT	1											5												
DYMMODL		2	3				4																	
STDMOCH			3						8															
STRMOOL				4			2																	
HANDQUL					5					7														
STRMODE						6												6						
GEOMDEY							7																	
AOSRVO								8	6															
STRDYNA						2	6		9	2	8				4							7		
CSYSANAL		8							6	10														
FAPERDCH											11						8							
RVSEDAT												12	3											
MISPERF													13	6										
VEHEPERF					3			7		2				14	9							4		
FAERDCH															15	6								
AERDAIL																16	7					8		
PRESEDF																	17	9						
STRANAL				4														7	18	6				
STRCTVT																			8	19	2			
VIANAL							4														8	20		
AERDMOL								6															21	
FINLDAT													5											22

Figure 59. Initial DSM Sequence.

Table 15. Project 2: Activities List.

Order	Description	ID	Time	Cost
1	Initial Data	INITDAT	40	20
2	Dynamic Model	DYNMODL	30	30
3	Stability & Model Characteristics	STDMOCH	40	20
4	Structure Model	STRMODL	10	50
5	Stability Qualities	HANDQUL	10	50
6	Structure Mode	STRMODE	10	50
7	Geometry Development	GEOMDEV	50	10
8	Aero Elastic Properties	AROSRVO	40	20
9	Aero Elastic Analysis	STRDYNA	50	10
10	Control System Analysis	CSYSANAL	20	40
11	Flex. Aero Characteristic	FAREROCH	20	40
12	Revise Initial Data	RVSEDAT	30	30
13	Mission Performance	MISPERF	30	30
14	Vehicle Performance	VEHEPERF	20	40
15	Rigid Aero Characteristic	RAEROCH	30	30
16	Aero Analysis	AEROANL	20	40
17	Pressure & Deflection	PRESDEF	30	30
18	Structure Analysis	STRANAL	40	20
19	Structure Weight	STRCTWT	50	10
20	Weight & Inertia Analysis	WIANAL	40	20
21	Aero Model	AEROMDL	20	40
22	Final Data	FINLDAT	20	40

Table 16. Project 2: Couplings List.

Coupling No.	From	To	Strength
1	INITDAT	GEOMDEV	es
2	RVSEDAT	INITDAT	n
3	MISPERF	RVSEDAT	vw
4	GEOMDEV	AEROMDL	es
5	GEOMDEV	STRMODL	ew
6	AEROMDL	AEROANL	es
7	PRESDEF	AEROANL	vs
8	AEROANL	PRESDEF	s
9	STRANAL	PRESDEF	es
10	PRESDEF	STRANAL	vs
11	STRCTWT	STRANAL	s
12	STRMODL	STRANAL	w
13	STRANAL	STRCTWT	es
14	WIANAL	STRCTWT	ew
15	GEOMDEV	WIANAL	w
16	STRCTWT	WIANAL	es
17	AEROANL	RAEROCH	s
18	PRESDEF	FAREROCH	es
19	STRANAL	STRMODE	s
20	WIANAL	VEHEPERF	w
21	RAEROCH	VEHEPERF	es
22	FAREROCH	VEHEPERF	ew
23	AROSRVO	VEHEPERF	vs
24	HANDQUL	VEHEPERF	vw
25	VEHEPERF	MISPERF	s
26	STRDYNA	STDMOCH	es
27	GEOMDEV	STRDYNA	s
28	RAEROCH	STRDYNA	w
29	FAREROCH	STRDYNA	es
30	STRMODE	STRDYNA	ew
31	WIANAL	STRDYNA	vs
32	CSYSANAL	STRDYNA	ew
33	STRDYNA	AROSRVO	s
34	GEOMDEV	DYNMODL	w
35	STDMOCH	DYNMODL	es
36	DYNMODL	CSYSANAL	es
37	STRDYNA	CSYSANAL	s
38	CSYSANAL	HANDQUL	vs
39	MISPERF	FINLDAT	n

### 9.2.1 Case (1)

In this case:

- All activities assume equal load of one unit.
- Both SA acceptance and rejection limits are set to 50.
- The objective is to minimize: “number of feedback couplings”

The results of the three runs are shown in Table 17. As shown, the minimum number of feedbacks – equals to 9 - was achieved in Run (3). The DSM corresponding to this solution is shown in Fig. 60.

Table 17. Case (1): Results.

Run #	1	2	3
<b>SA Settings</b>			
Initial Temp.	55	60	60
Final Temp.	10	10	5
Cooling Factor	0.98	0.95	0.95
<b>Initial Configuration</b>			
Load	660	660	660
Feedbacks *	23	23	23
(Time + Cost)	39600	39600	39600
<b>1<sup>st</sup> Meta-Stable Solution</b>			
Load	689	915	698
Feedbacks *	16	15	16
(Time + Cost)	41340	54900	41880
<b>Optimal Solution</b>			
Load	769	681	568
Feedbacks *	10	10	9
(Time + Cost)	46140	40860	34080
<b>% reduction w.r.t. Initial Configuration</b>			
Load	(16.52)	(3.18)	13.94
Feedbacks *	56.52	56.52	60.87
(Time + Cost)	(16.52)	(3.18)	13.94
<b>% reduction w.r.t. 1<sup>st</sup> meta-stable solution</b>			
Load	(11.61)	25.57	18.62
Feedbacks *	37.50	33.33	43.75
(Time + Cost)	(11.61)	25.57	18.62
<b>Solution Time (sec)</b>			
Evaluated Solutions			
Total	4363	1797	2539
Accepted	4250	1750	2450

\* Objective Function

	INITDAT	GEOMDEV	VIANAL	DYNAMOOL	RAEROCH	PREDEF	STRCTVT	STRMODL	CSYSANAL	AEROMDL	STRANAL	FAEROCHI	RYS DAT	STRMODE	STRDYNA	AROSRVO	HANDQUL	STDMOCH	VEHEPERF	AERDANL	MISPERF	FINLDAT	
INITDAT	1												5										
GEOMDEV	8	2																					
VIANAL		4	3				8																
DYNAMOOL		4		4														8					
RAEROCH					5															8			
PREDEF						6						8									8		
STRCTVT			2				7				8												
STRMODL		2						8															
CSYSANAL				8					9						8								
AEROMDL		8								10													
STRANAL						7	6	4			11												
FAEROCHI						8						12											
RYS DAT													13									3	
STRMODE											6			14									
STRDYNA		6	7		4			2			8		2	15									
AROSRVO														6	16								
HANDQUL								7									17						
STDMOCH															8								
VEHEPERF			4		8							2				7	3				19		
AERDANL						7				8												20	
MISPERF																					6		21
FINLDAT																						5	22

Figure 60. Case (1): DSM with Minimum Number of Feedbacks.

The results obtained from the other two runs (1 and 2) reveal a very important conclusion. Notice that although the number of feedbacks was improved (10 instead of 16), the other two measures – load, time and cost – did not. On the contrary, their values increased. Thus, minimizing the number of feedback coupling does not necessarily imply a lower total time and cost. Finally, Fig.61 represents the meta-stable values of the three measures (number of feedbacks, load, and time and cost) at different temperatures.

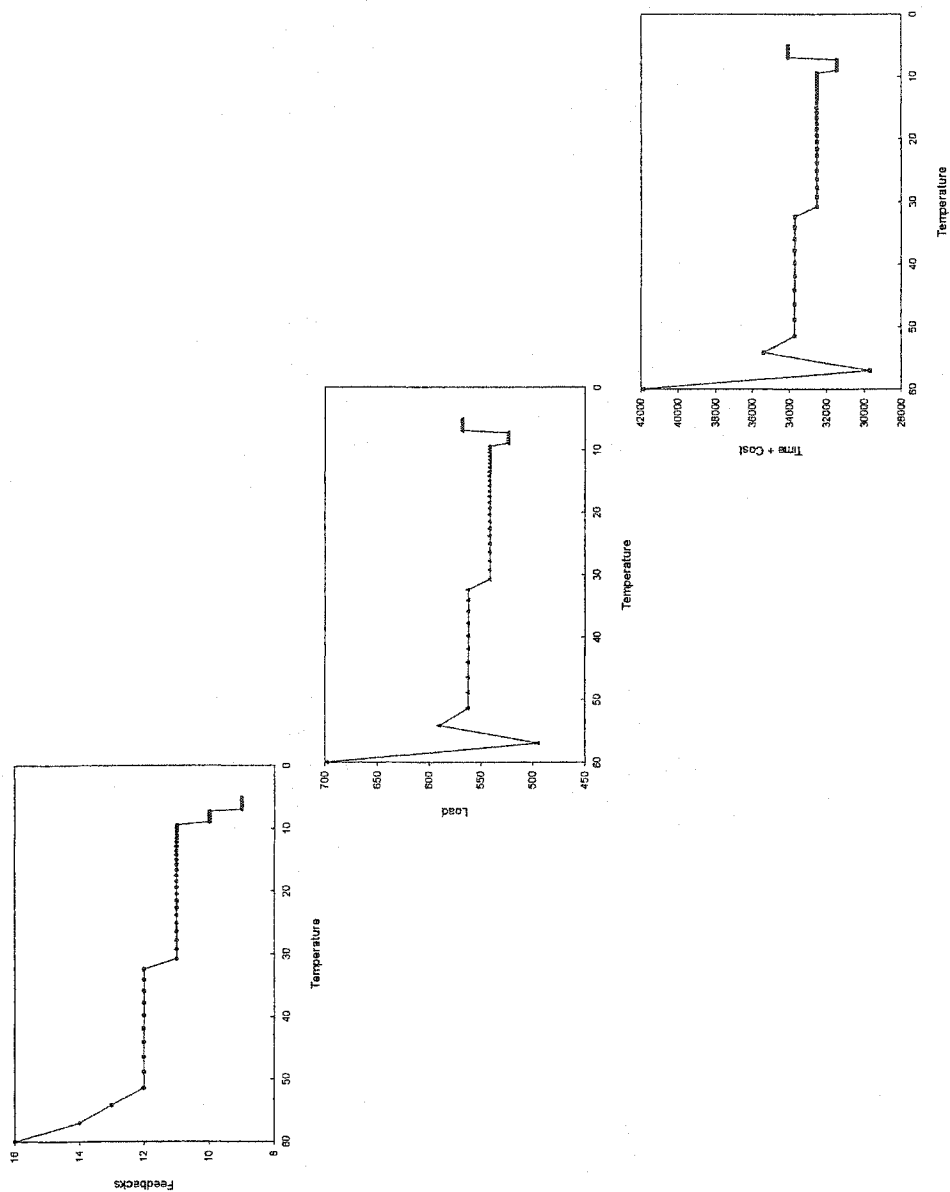


Figure 61. Case (1): Meta-stable Objective Functions at Different Temperatures.

### 9.2.2 Case (2)

In this case:

- All activities assume equal load of one unit.
- Both SA acceptance and rejection limits are set to 50.
- The objective is to minimize: “total load”
- Activity loads are deterministic.

The results of the three runs are shown in Table 18. As shown, the minimum load – equals to 130 units - was achieved in Run (1). The DSM corresponding to this solution is shown in Fig. 62. The meta-stable values of the three measures (number of feedbacks, load, and time and cost) at different temperatures are shown in Fig. 63.

Table 18. Case (2): Results.

Run #	1	2	3
<b>SA Settings</b>			
Initial Temp.	55	60	60
Final Temp.	10	10	5
Cooling Factor	0.98	0.95	0.95
<b>Initial Configuration</b>			
Load	660	660	660
Feedbacks	23	23	23
(Time + Cost)	39600	39600	39600
<b>1<sup>st</sup> Meta-Stable Solution</b>			
Load	436	574	605
Feedbacks	15	14	20
(Time + Cost)	26160	34440	36300
<b>Optimal Solution</b>			
Load	130	160	136
Feedbacks	7	9	7
(Time + Cost)	7800	9600	8160
<b>% reduction w.r.t. Initial Configuration</b>			
Load	80.30	75.76	79.39
Feedbacks	69.57	60.87	69.57
(Time + Cost)	80.30	75.76	79.39
<b>% reduction w.r.t. 1<sup>st</sup> meta-stable solution</b>			
Load	70.18	72.13	77.52
Feedbacks	53.33	35.71	65.00
(Time + Cost)	70.18	72.13	77.52
Solution Time (sec)	212.14	89.18	127.08
<b>Evaluated Solutions</b>			
Total	6273	2673	3356
Accepted	2156	1040	1027

\* Objective Function

	INITDAT	RVSEDAT	GEOMDEV	STRMODL	AEROMDL	STRCTVT	STRANAL	PRESDEF	AEROANL	WIANAL	STRMODE	RAEROCH	FAEROCH	STRDYNA	STDMOCH	DYNMODL	CSYSANL	HANDQUL	AROSRVO	VEHEPERF	MISPERF	FINLDAT	
INITDAT	1	5																					
RVSEDAT		2																				3	
GEOMDEV	8		3																				
STRMODL			2	4																			
AEROMDL			8		5																		
STRCTVT						6	8				2												
STRANAL			4			6	7	7															
PRESDEF							8	8	6														
AEROANL					8		7	9															
WIANAL		4				8				10													
STRMODE						6					11												
RAEROCH								6				12											
FAEROCH							8						13										
STRDYNA			5							7	2	4	8	14				2					
STDMOCH														8	15								
DYNMODL			4												8	16							
CSYSANL														6	8	17							
HANDQUL																	7	18					
AROSRVO												6								19			
VEHEPERF									4		8	2						3	7		20		
MISPERF																					6	21	
FINLDAT																						5	22

Figure 62. Case (2): DSM with Minimum Total Load.

A closer look at the results reveal that, in contrast to Case (1), a minimum total load would result in a minimum number of feedbacks and minimum time and cost. The number of feedbacks corresponding to minimum total load is even lower than the number obtained in Case (1) – 7 compared to 9.



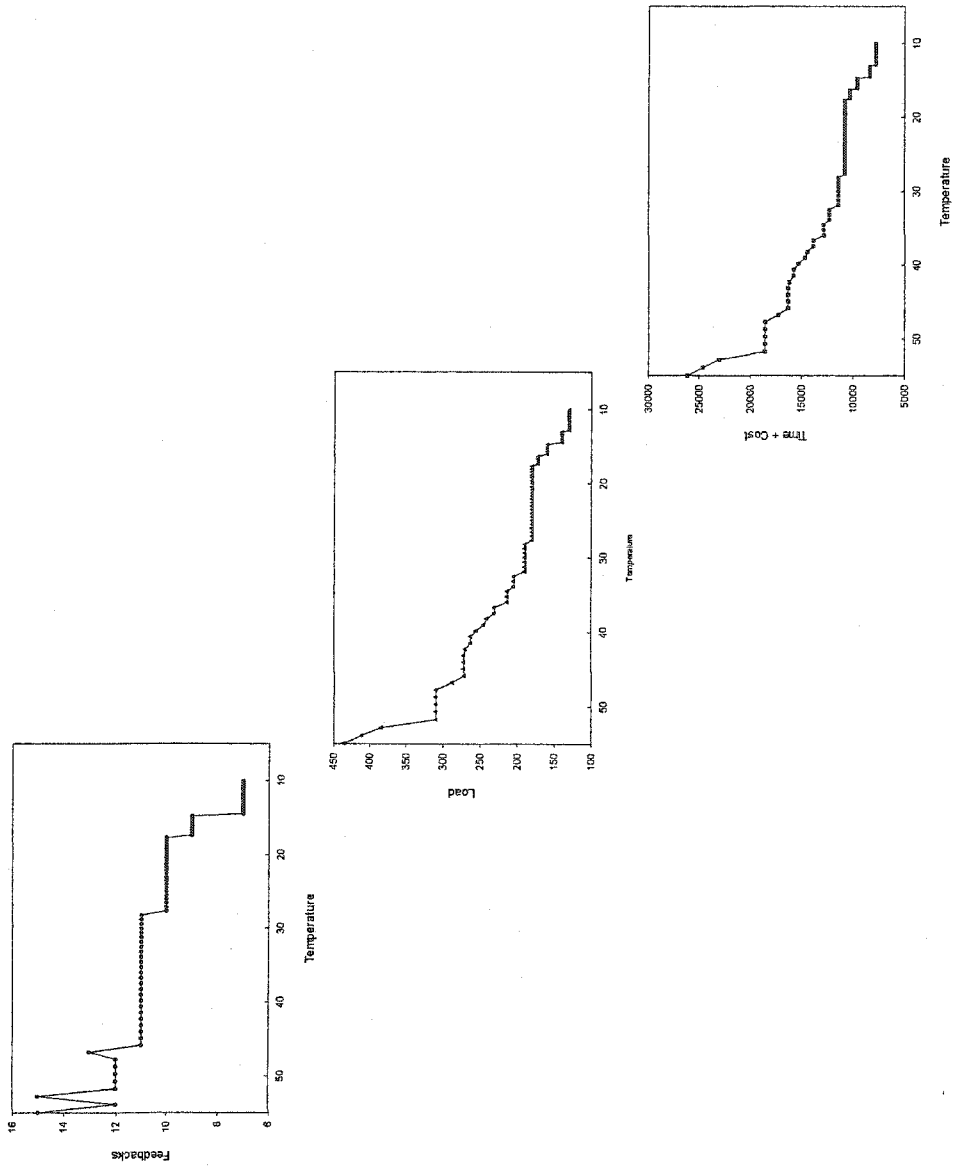


Figure 63. Case (2): Meta-stable Objective Functions at Different Temperatures.

### 9.2.3 Case 3

Finally, in this case:

- All activities assume equal load of one unit.
- Both SA acceptance and rejection limits are set to 50.
- The objective is to minimize: “total time + cost”
- Activity time and cost are deterministic

The results of the three runs are shown in Table 20. As shown, the minimum objective function – equals to 8100 units - was achieved in Run (2). The DSM corresponding to this solution is shown in Fig. 64.

The meta-stable values of the three measures (number of feedbacks, load, and time and cost) at different temperatures are shown in Fig.65. It can be noticed that a minimum time and cost would result in lower load and number of feedbacks. In Fig.66, the objective function values corresponding to all accepted solutions are shown. The figure illustrates the basic concept on which SA avoids being trapped in a local minima; the acceptance of some solutions with objective functions higher than the current optimal one.

Table 19. Case (3): Results.

Run #	1	2	3
<b>SA Settings</b>			
Initial Temp.	55	55	60
Final Temp.	10	10	10
Cooling Factor	0.98	0.98	0.95
<b>Initial Configuration</b>			
Load	660	660	660
Feedbacks *	23	23	23
(Time + Cost)	39600	39600	39600
<b>1<sup>st</sup> Meta-Stable Solution</b>			
Load	305	451	526
Feedbacks	13	14	14
(Time + Cost) *	18300	27060	31560
<b>Optimal Solution</b>			
Load	136	135	136
Feedbacks	8	7	8
(Time + Cost) *	8160	8100	8160
<b>% reduction w.r.t. Initial Configuration</b>			
Load	79.39	79.55	79.39
Feedbacks	65.22	69.57	65.22
(Time + Cost) *	79.39	79.55	79.39
<b>% reduction w.r.t. 1<sup>st</sup> meta-stable solution</b>			
Load	55.41	70.07	74.14
Feedbacks	38.46	50.00	42.86
(Time + Cost) *	55.41	70.07	74.14
<b>Solution Time (sec)</b>			
Total	176.14	226.90	86.63
<b>Evaluated Solutions</b>			
Total	4397	5664	2433
Accepted	147	1438	728

\* Objective Function

	INITDAT	RYSEDAT	GEOIMDEV	STRMODL	AEROMDL	STRCTVT	STRANAL	VIANAL	PRESEDF	AERDANL	RAEROCH	FAEROCH	STRMODE	STRDYNA	AROSRVO	STDMOCH	DYNAMOVL	CSYSANAL	HANDQUL	VEHEPERF	MISPERF	FINLDAT
INITDAT	1	5																				
RYSEDAT		2																				3
GEOIMDEV	8		3																			
STRMODL		2	4																			
AEROMDL		8		5																		
STRCTVT						6	8	2														
STRANAL			4		6	7		7														
VIANAL		4			8		8															
PRESEDF						8		9	6													
AERDANL				8				7	10													
RAEROCH									8	11												
FAEROCH								8			12											
STRMODE						6						13										
STRDYNA			6				7			4	8	2	14					2				
AROSRVO														6	15							
STDMOCH														8		15						
DYNAMOVL			4													8	17					
CSYSANAL													6				8	18				
HANDQUL																		7	19			
VEHEPERF							4			8	2			7					3	20		
MISPERF																				6	21	
FINLDAT																					5	22

Figure 64. Case (3): DSM with Minimum Total Time and Cost.

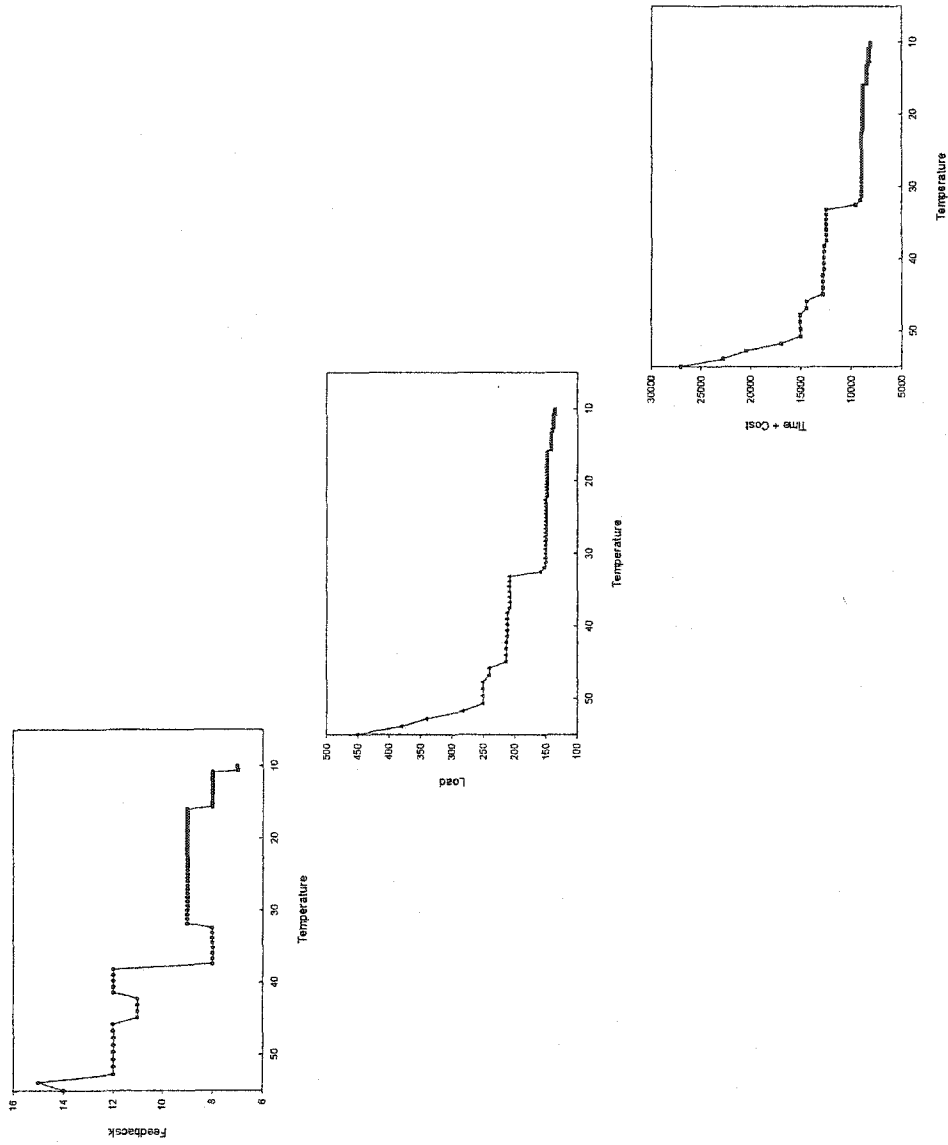


Figure 65. Case (3): Meta-stable Objective Functions at Different Temperatures.

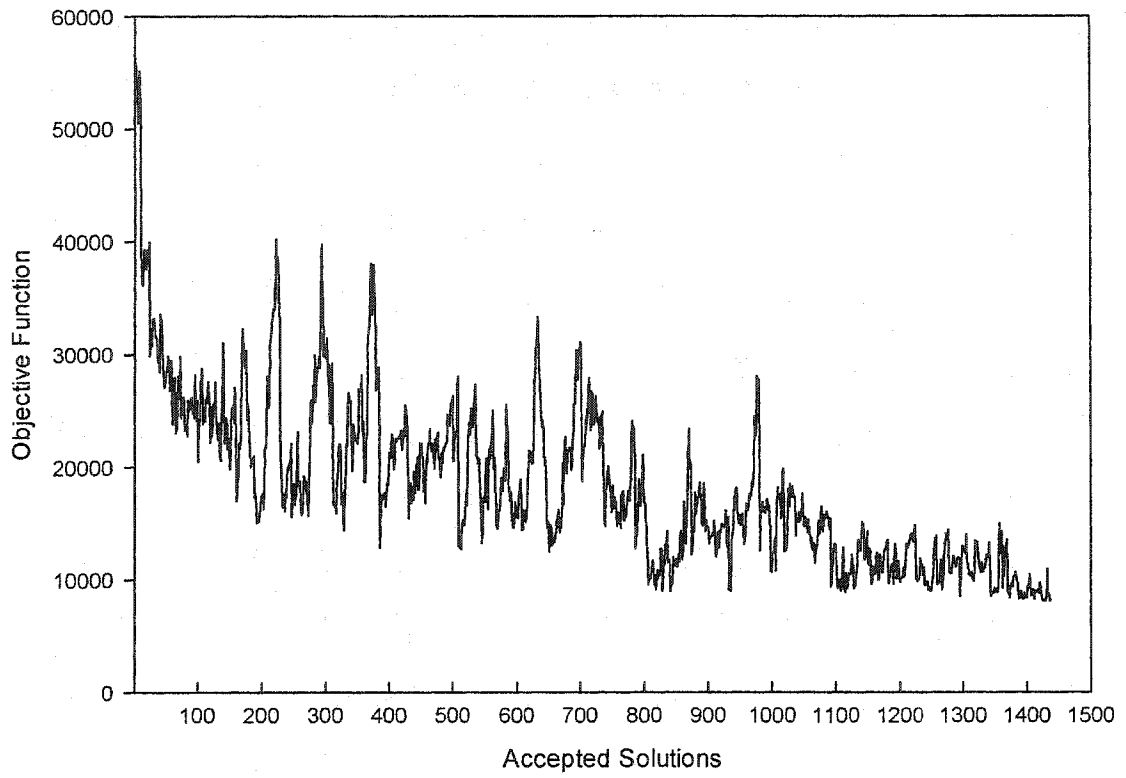


Figure 66. Case (3): Accepted Solutions.

### 9.3 Project 3: A Conceptual Design Project – Stochastic Case

This project is the same one presented in the previous section, but with a main difference, that is: activities assume stochastic load, shown in Table 20. Probability distributions used and associated parameters are shown in Fig. 67.

Other settings are:

- Both SA acceptance and rejection limits are set to 50.
- The objective is to minimize: “total load”

Both comparison rules (min-mean-max, and utility function method) presented in Chapter VI will be implemented in Sections 3.1 and 3.2 respectively.

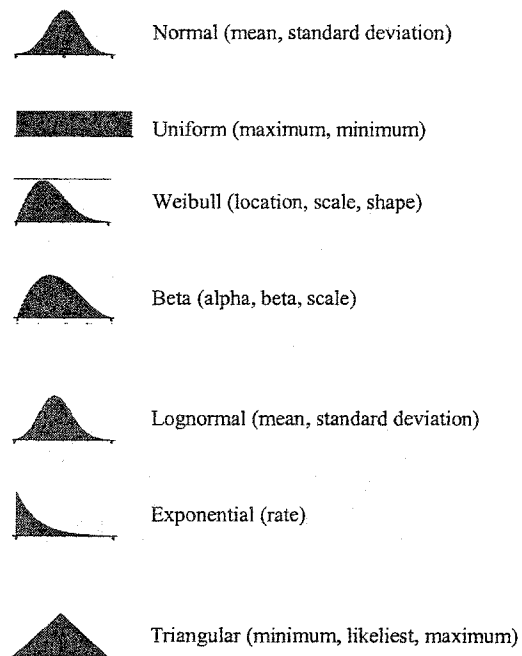


Figure 67. Probability Distributions' Parameters.

Table 20. Project 3: Activities List.

Order	Description	ID	Load
1	Initial Data	INITDAT	Normal (40, 4)
2	Dynamic Model	DYNMODL	Uniform (27, 33)
3	Stability & Model Characteristics	STDMOCH	Weibull (40, 1, 2)
4	Structure Model	STRMODL	Beta (2, 3, 10)
5	Stability Qualities	HANDQUL	Lognormal (10, 1)
6	Structure Mode	STRMODE	Normal (10, 1)
7	Geometry Development	GEOMDEV	Normal (50, 5)
8	Aero Elastic Properties	AROSRVO	Normal (40, 4)
9	Aero Elastic Analysis	STRDYNA	Normal (50, 5)
10	Control System Analysis	CSYSANAL	Normal (20, 2)
11	Flex. Aero Characteristic	FAREROCH	Normal (20, 2)
12	Revise Initial Data	RVSEDAT	Normal (30, 3)
13	Mission Performance	MISPERF	Exponential (0.03)
14	Vehicle Performance	VEHEPERF	Beta (2, 3, 20)
15	Rigid Aero Characteristic	RAEROCH	Beta (2, 3, 30)
16	Aero Analysis	AEROANL	Beta (2, 3, 20)
17	Pressure & Deflection	PRESDEF	Weibull (30, 1, 2)
18	Structure Analysis	STRANAL	Uniform (36, 44)
19	Structure Weight	STRCTWT	Beta (2, 3, 50)
20	Weight & Inertia Analysis	WIANAL	Uniform (36, 44)
21	Aero Model	AEROMDL	Beta (2, 3, 20)
22	Final Data	FINLDAT	Triangular (18, 20, 22)



### 9.3.1 Min-Mean-Max (M3) Results

In this section, the min-mean-max comparison rules for optimization with stochastic activity loads are implemented. Table 21 shows the results from two runs, each with different initial solution, and different cooling schedule. The DSM corresponding to the second run is shown in Fig.68. Figures 69 and 70 represents the probability distribution curves for the initial and optimal solutions respectively. Finally, Fig.71 illustrates the minimum, mean, and maximum values corresponding to the objective function meta-stable values.

To study the robustness of the solution obtained, Fig.72 was created. As shown, while the M3 rules led to minimum mean and minimum min, the maximum values were not minimized. Thus, the final solution is less robust than its preceding solution. Again, Fig.73 illustrates how the modified SA manages to avoid local optima.

Table 21. M3: Results.

Run #	1	2
<b>SA Settings</b>		
Initial Temp.	55	60
Final Temp.	2	2
Cooling Factor	0.95	0.95
<b>Initial Configuration</b>		
Min	14472.25	14472.25
Mean	17179.33	17179.33
Max	23044.19	23044.19
Standard Deviation	1188.11	1188.11
<b>1<sup>st</sup> Meta-Stable Solution</b>		
Min	9172.61	17860.41
Mean	10663.56	20698.67
Max	15708.47	24415.73
Standard Deviation	943.4	945.05
<b>Optimal Solution</b>		
Min	3789.36	3377.17
Mean	4453.2	4181.34
Max	6260.28	6763.51
Standard Deviation	319.32	391.77
<b>% reduction w.r.t. Initial Configuration</b>		
Min	73.82	76.66
Mean	74.08	75.66
Max	72.83	70.65
Standard Deviation	73.12	67.03
<b>% reduction w.r.t. 1<sup>st</sup> meta-stable solution</b>		
Min	58.69	81.09
Mean	58.24	79.80
Max	60.15	72.30
Standard Deviation	77.10	82.89
Solution Time (sec)	1795.7	4018.9
<b>Evaluated Solutions</b>		
Total	4198	4578
Accepted	298	558

	INITDAT	RVSEDAT	STRMODL	GEOMDEV	AEROMDL	STRANAL	STRCTVT	PRESDEF	AEROANL	VIANAL	FAPEROC	STRMODE	FAEROCH	STRDYNA	MISPERF	AROSRYO	VEHEPERF	STDMOCH	DYNAMOOL	CSYSANAL	HANDQUL	FINLDAT
INITDAT	1	5																				
RVSEDAT		2													3							
STRMODL			2																			
GEOMDEV	8			4																		
AEROMDL					5																	
STRANAL			4			6	6	7														
STRCTVT							8	7			2											
PRESDEF								8	6													
AEROANL					8				7	9												
VIANAL				4			8				10											
FAPEROC								8				11										
STRMODE						6							12									
FAEROCH									6					13								
STRDYNA				6						7	8	2	4	14							2	
MISPERF															15		6					
AROSRYO														6		16						
VEHEPERF										4	2		8				7	17				3
STDMOCH														8					18			
DYNAMOOL				4														8		19		
CSYSANAL														6						8	20	
HANDQUL																					7	21
FINLDAT															5							22

Figure 68. M3: DSM Corresponding to the Optimal Solution.

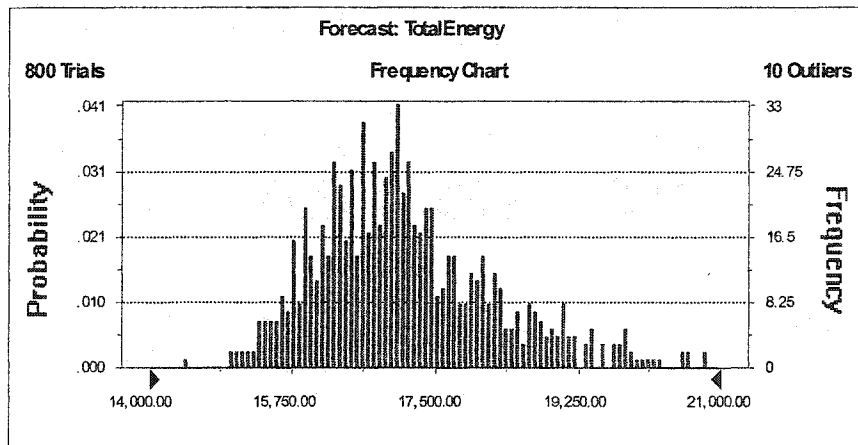


Figure 69. M3: Probability Distribution of the Initial Solution.

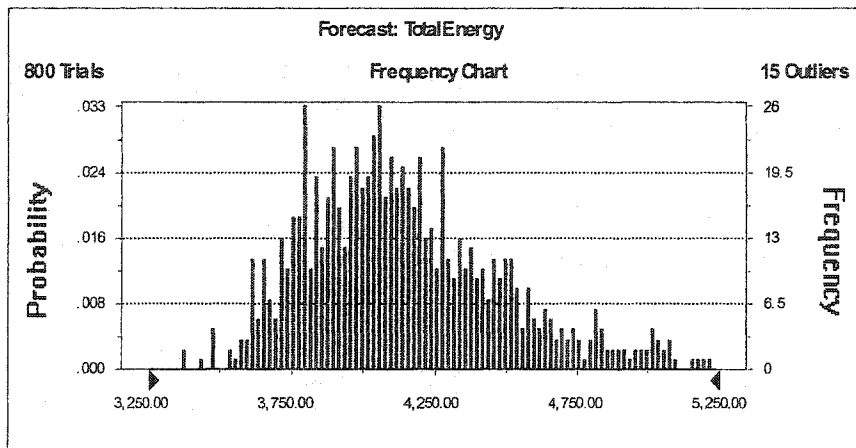


Figure 70. M3: Probability Distribution of the Optimal Solution.

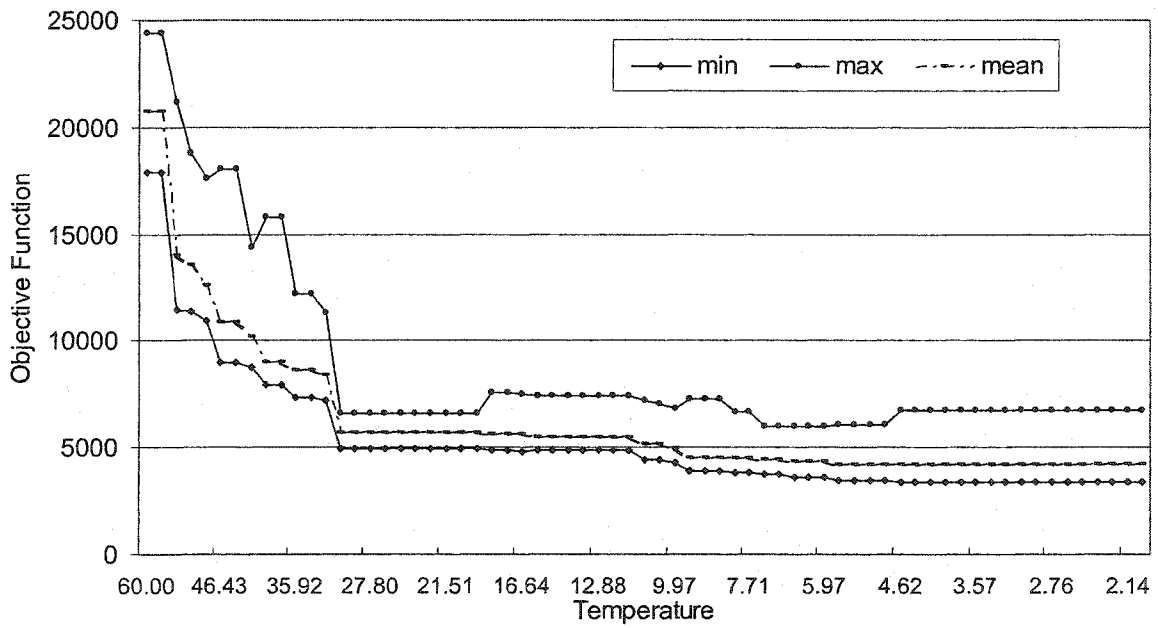


Figure 71. M3: Meta-stable Objective Function Values.

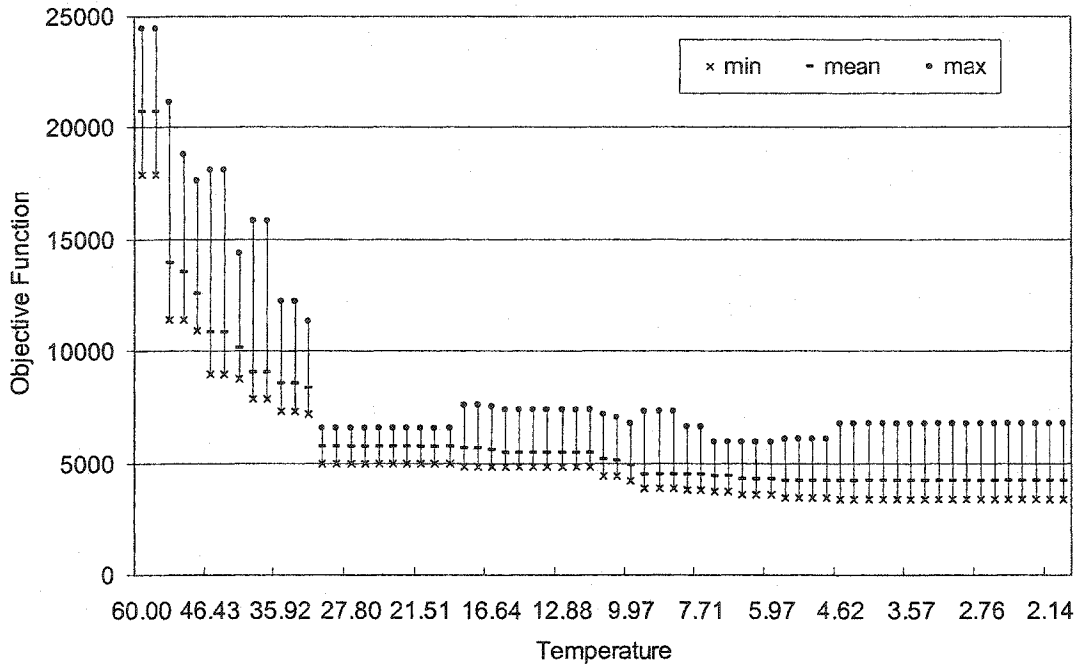


Figure 72. M3: Solution Robustness.

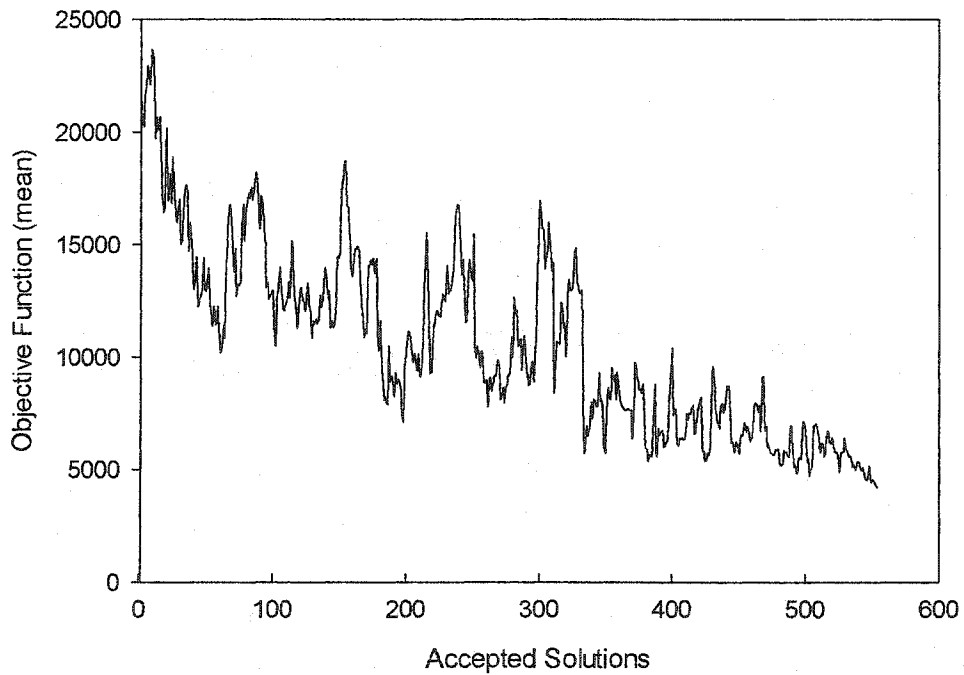


Figure 73. M3: Accepted Solutions.

### 9.3.2 Utility Function Method Results

An alternative method to the M3 is the Utility method. Table 22 shows the results from two runs, each with different initial solution, and different cooling schedule. The DSM corresponding to the second run is shown in Fig.74. Figure 75 illustrates the mean, range, and maximum values corresponding to the objective function meta-stable values.

Table 22. Utility Function Method: Results.

Run #	1	2
<b>SA Settings</b>		
Initial Temp.	50	60
Final Temp.	2	2
Cooling Factor	0.95	0.95
<b>Initial Configuration</b>		
Mean	17179.33	17179.33
Variance	1411605.37	1411605.37
Range	5864.86	5864.86
Max	23044.19	23044.19
<b>1<sup>st</sup> Meta-Stable Solution</b>		
Mean	17468.62	11421.86
Variance	484163.07	479527.13
Range	4558.88	4333.92
Max	20147.21	14330.93
<b>Optimal Solution</b>		
Mean	4266.85	4109.9
Variance	32923.62	33778.41
Range	1051.01	985.68
Max	4858.35	4673
<b>% reduction w.r.t. Initial Configuration</b>		
Mean	75.16	76.08
Variance	97.67	97.61
Range	82.08	83.19
Max	78.92	79.72
<b>% reduction w.r.t. 1<sup>st</sup> meta-stable solution</b>		
Mean	75.57	64.02
Variance	93.20	92.96
Range	76.95	77.26
Max	75.89	67.39
<b>Solution Time (sec)</b>		
	7608.5	14010.9
<b>Evaluated Solutions</b>		
Total	3991	5227
Accepted	887	1289

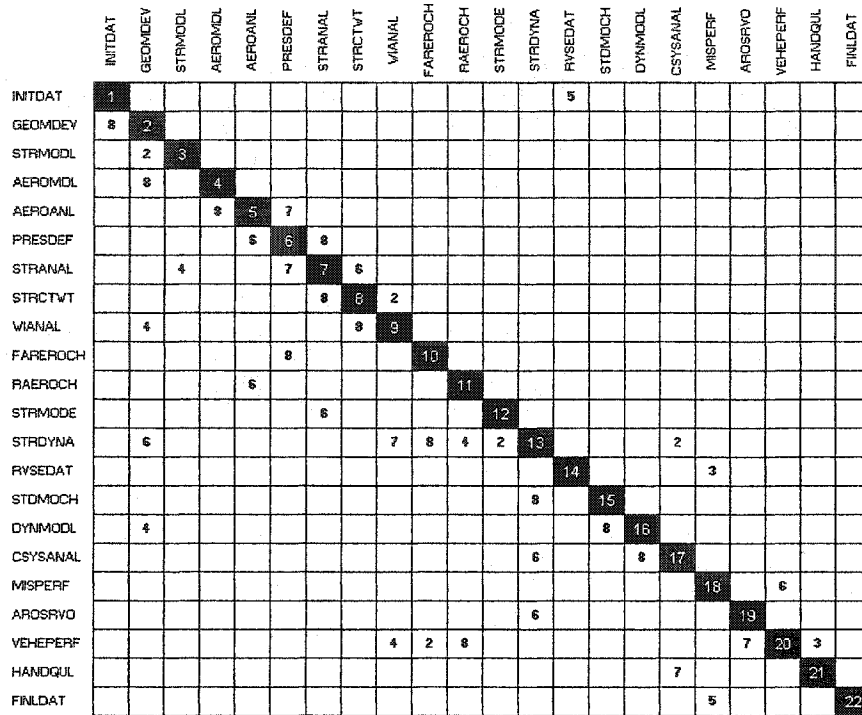


Figure 74. Utility Function Method: DSM.

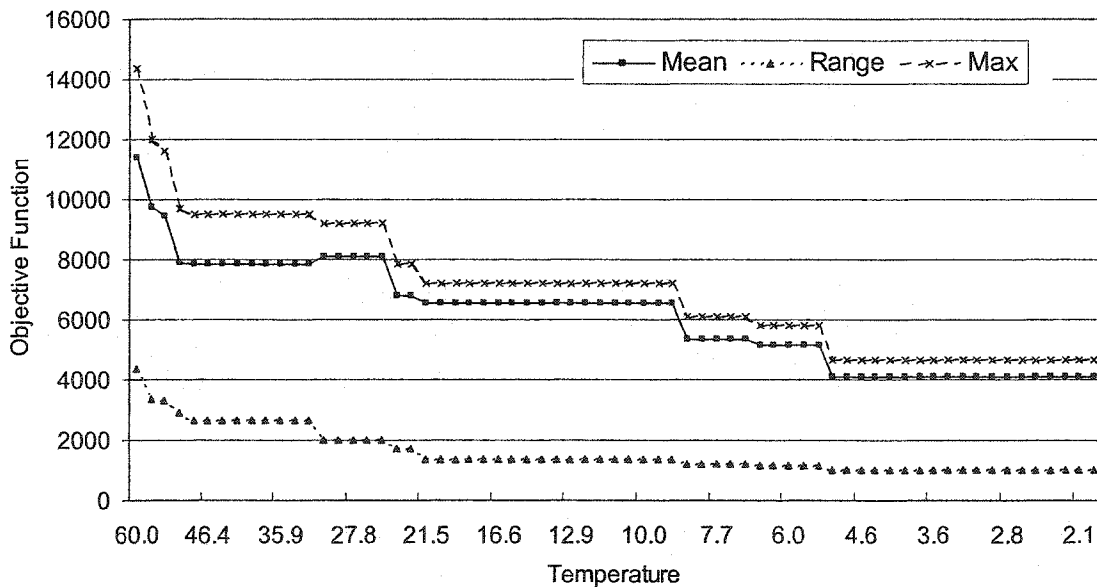


Figure 75. Utility Function Method: Meta-stable Objective Function Values.

To study the robustness of the solution obtained, let's consider Fig.76. As shown, the implementation of the utility function methods led to an optimal and robust solution.

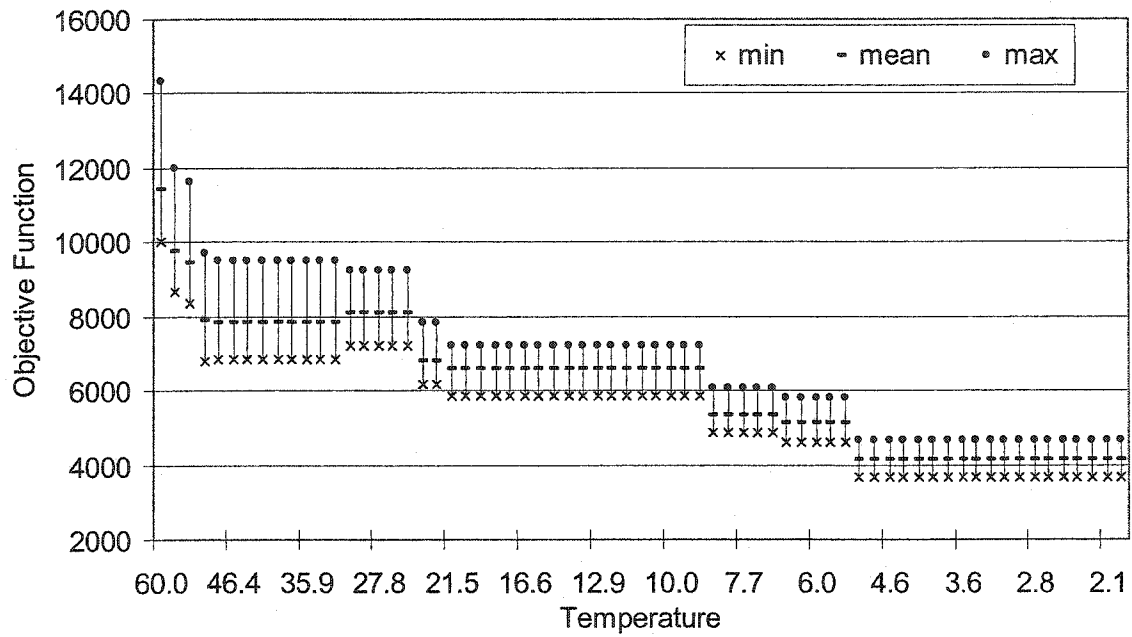


Figure 76. Utility Function Method: Solution Robustness.



## 9.4 Project 4: Illustrative Project

The purpose of the current and last case is to illustrate how the optimized DSM is converted into a program. A hypothetical project is considered. The project consists of 16 activities (Table 23 lists these activities and their associated load), and contains 38 couplings (shown in Table 24), and finally two logical constraints (shown in Table 25) are imposed on the project.

Table 23. Project 4: Activities List.

Order	Activity	
	ID	Load
1	K	50
2	B	40
3	C	30
4	D	20
5	E	40
6	F	30
7	G	50
8	H	60
9	I	10
10	J	20
11	A	30
12	L	40
13	M	50
14	N	20
15	P	60
16	O	30

Table 24. Project 4: Couplings List.

No.	From	To	Strength
1	A	H	n
2	A	P	s
3	A	O	w
4	B	D	es
5	B	C	n
6	B	M	vs
7	B	G	s
8	C	G	s
9	C	O	w
10	D	H	vw
11	D	P	ew
12	D	C	n
13	E	G	n
14	E	O	s
15	F	D	s
16	F	N	vs
17	G	A	vs
18	G	D	es
19	G	M	n
20	H	N	w
21	H	P	n
22	H	C	vw
23	H	M	s
24	I	H	s
25	I	G	n
26	J	C	n
27	J	M	vw
28	K	D	n
29	L	P	s
30	M	G	s
31	N	P	vw
32	N	M	w
33	N	G	s
34	N	O	n
35	O	A	es
36	O	D	s
37	P	H	s
38	P	N	w

Table 25. Project 4: Logical Constraints.

No.	Activity	Precedes
1	F	L
2	B	I

### 9.4.1 Optimization

Figure 77 represents the DSM corresponding to an initial activity sequence. The initial configuration results in 15 feedbacks and a total load of 15700 units. Figure 78 represents the DSM resulted from minimizing the total feedbacks. This DSM has only 7 feedbacks and a total load of 13040 units. Figure 79 shows the DSM corresponding to the sequence of minimum load. This DSM has 10 feedbacks and total load of 7540. The DSM was obtained using Geometric cooling schedule, with  $T_0 = 50$ ,  $T_f = 20$ ,  $\alpha = 0.98$ , acceptance limit = 60, and rejection limit = 60.

	K	B	C	D	E	F	G	H	I	J	A	L	M	N	P	O
K	1															
B		2														
C		5	3	5				3		5						
D	5	8		4		6	8									6
E					5											
F						6										
G		6	6		5		7		5				6	6		
H				3				8	6	5						6
I									9							
J										10						
A						7					11					8
L												12				
M		7					5	6	3				13	4		
N						7		4						14	4	
P				2				5			6	8		3	15	
O			4		6						4			5		16

Figure 77. Project 4: Initial DSM.

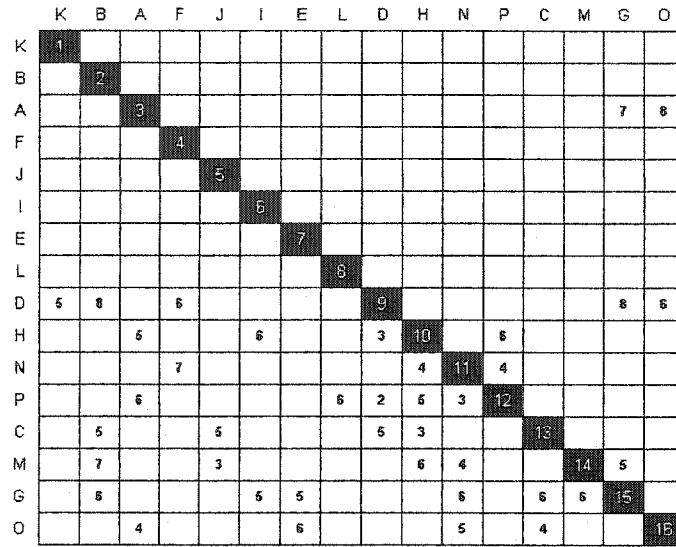


Figure 78. Project 4: DSM with Minimum Number of Feedbacks.

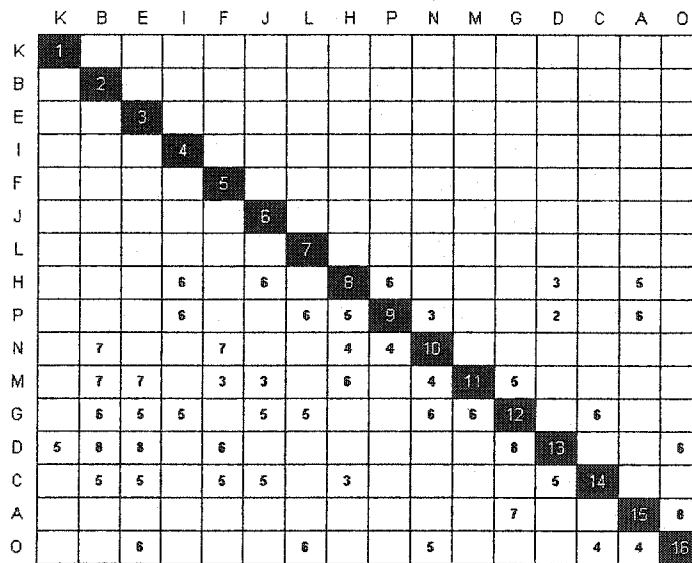


Figure 79. Project 4: DSM with Minimum Total Load.

### 9.4.2 Conversion to a Project Schedule

Based on the optimal DSM with minimum load, the rules for conversion to a project schedule are summarized in Table 26.

Table 26. Conversion to a Project Schedule.

Step #	Details			Figure
	Activities	Pattern	Equivalent	
1	H & P	I $C_1=6$	Merge activities into 'Block 1' Block load = $0.5 \times 720 = 360$ units Addition to project buffer = $0.2 \times 720 = 144$ units	80
2	Block 1 & N	I $C_1=3, C_2 > 3$	Suspense the feedback coupling Addition to project buffer = $0.3 \times 240 = 72$ units	81
3	M & G	I $C_1=5, C_2=6$	Merge activities into 'Block 2' Block load = $0.5 \times 500 = 250$ units Addition to project buffer = $0.2 \times 500 = 100$ units	82
4	A & O	I $C_1=8$	Merge activities into 'Block 3' Block load = $0.5 \times 480 = 240$ units Addition to project buffer = $0.2 \times 480 = 96$ units	82
5	Block 1 & D	C Length > 3	Suspense the feedback coupling Insert 'Buffer 1' with load = $0.4 \times 780 = 312$ units Addition to project buffer = $0.2 \times 780 = 156$ units	83
6	Block 2 & C	C Length = 3	Suspense the feedback coupling Insert 'Buffer 2' with load = $0.5 \times 600 = 300$ units Addition to project buffer = $0.2 \times 600 = 120$ units	84
7	Block 3 & D	S	Suspense the feedback coupling Insert 'Buffer 3' with load = $0.4 \times 660 = 264$ units Addition to project buffer = $0.3 \times 660 = 198$ units	85
8	Block 3 & Block 1	S	Suspense the feedback coupling Insert 'Buffer 4' with load = $0.4 \times 1560 = 300$ units Addition to project buffer = $0.3 \times 1560 = 468$ units	85
9	Buffer 3 & Buffer 4		Merge as Buffer 3 with load = $264 + 300 = 564$ units	85

	K	B	E	I	F	J	L	Block1	N	M	G	D	C	A	O
K	1														
B		2													
E			3												
I				4											
F					5										
J						6									
L							7								
Block1				6		6	6	8	3			3		6	
N		7			7			4	10						
M		7	7		3	3		6	4	11	5				
G		6	5	5		5	5		6	6	12		6		
D	5	8	8		6						8	13			6
C		5	5		5	5		3				5	14		
A										7				15	8
O			6				6		5				4	4	16

Figure 80. Conversion to a Project Schedule: Step 1.

	K	B	E	I	F	J	L	Block1	N	M	G	D	C	A	O
K	1														
B		2													
E			3												
I				4											
F					5										
J						6									
L							7								
Block1				6		6	6	8				3		6	
N		7			7			4	10						
M		7	7		3	3		6	4	11	5				
G		6	5	5		5	5		6	6	12		6		
D	5	8	8		6						8	13			6
C		5	5		5	5		3				5	14		
A										7				15	8
O			6				6		5				4	4	16

Figure 81. Conversion to a Project Schedule: Step 2.

	K	B	E	I	F	J	L	Block1	N	Block 2	D	C	Block 3
K	1												
B		2											
E			3										
I				4									
F					5								
J						6							
L							7						
Block1				6		6	6	8			3		6
N		7			7			4	10				
Block 2		7	7	5	3	5	5	6	6	11		6	
D	5	8	8		6					8	13		6
C		5	5		5	5		3			5	14	
Block 3			6				6		5	7		4	15

Figure 82. Conversion to a Project Schedule: Steps 3 and 4.

	K	B	E	I	F	J	L	Block1	N	Block 2	D	Buffer1	C	Block 3
K	1													
B		2												
E			3											
I				4										
F					5									
J						6								
L							7							
Block1				6		6	6	8						6
N		7			7			4	10					
Block 2		7	7	5	3	5	5	6	6	11			6	
D	5	8	8		6					8	13			6
Buffer1											5			
C		5	5		5	5		3				5	14	
Block 3			6				6		5	7			4	15

Figure 83. Conversion to a Project Schedule: Step 5.

	K	B	E	I	F	J	L	Block1	N	Block2	D	Buffer1	C	Buffer2	Block3
K	1														
B		2													
E			3												
I				4											
F					5										
J						6									
L							7								
Block1				6		6	6	8							6
N		7			7			4	10						
Block2		7	7	5	3	5	5	6	6	11					
D	5	8	8		6					8	13				6
Buffer1											5				
C		5	5		5	5		3				5	14		
Buffer2													5		
Block3			6				6		5	7				4	15

Figure 84. Conversion to a Project Schedule: Step 6.

	K	B	E	I	F	J	L	Block1	N	Block2	D	Buffer1	C	Buffer2	Block3	Buffer3
K	1															
B		2														
E			3													
I				4												
F					5											
J						6										
L							7									
Block1				6		6	6	8								
N		7			7			4	10							
Block2		7	7	5	3	5	5	6	6	11						
D	5	8	8		6					8	13					
Buffer1											5					
C		5	5		5	5		3				5	14			
Buffer2													5			
Block3			6				6		5	7				4	15	
Buffer3															5	

Figure 85. Conversion to a Project Schedule: Steps 7, 8, and 9.

### 9.4.3 Time-Cost Trade-Off

Now that the DSM has been converted to a program, as shown in Table 27, the analysis can proceed to the last phase of the architecture; time-cost trade-off.

For simplicity, it is assumed that all activities share/require the same resource type, and that resource has three level shown in Table 28. The TCTO module presented earlier was implemented. The TCTO curve for the project is shown Fig.86.

Consider the case in which required project completion is set to 56 hours, the resources assignments, resources usage, and corresponding schedule (Gantt chart) are shown in Table 20, Table 30, and Fig.87 respectively. Finally, the meta-stable values of the objective function (project cost) are shown in Fig.88.

Table 27. Activities List of the Equivalent Schedule.

Activity	Load	Precedes
K	50	D
B	40	N, Block 2, D, C
E	40	Block 2, D, C
I	10	Block 1, Block 2
F	30	N, Block 2, D, C
J	20	Block 1, Block 2, C
L	40	Block 1, Block 2, Block 3
Block1	360	N, Block 2, C
N	20	Block 2, Block 3
Block 2	250	D, Block 3
D	20	Buffer 1
Buffer1	312	C
C	30	Buffer 2
Buffer 2	300	Block 3
Block 3	240	Buffer 3
Buffer 3	564	Project Buffer
Project Buffer	1258	-



Table 28. Resource Data.

Resource Level	Productivity (units/hr)	Cost rate (\$/hr)	Availability
Slow	20	7	40
Normal	30	10	40
Fast	40	20	40

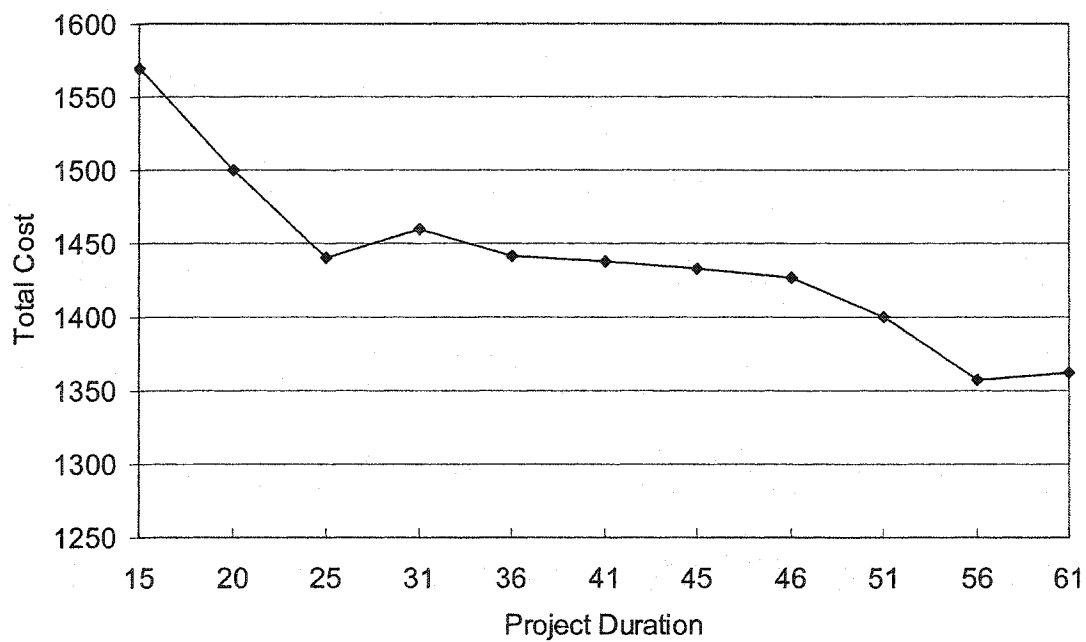


Figure 86. TCTO Curve.

Table 29. Resource Assignment for 56 hours.

Activity	Load	Resource Assignment			Duration	Cost
		Slow	Normal	Fast		
K	50	1	0	0	3	21
B	40	1	0	0	2	14
E	40	1	1	0	1	17
I	10	1	0	0	1	7
F	30	0	1	0	1	10
J	20	1	0	0	1	7
L	40	0	1	1	1	30
Block1	360	0	3	0	4	120
N	20	0	0	1	1	20
Block 2	250	0	3	4	1	110
D	20	0	1	0	1	10
Buffer1	312	1	5	0	2	114
C	30	2	0	0	1	14
Buffer 2	300	0	1	0	10	100
Block 3	240	5	2	2	1	95
Buffer 3	564	0	7	2	2	220
Project Buffer	1258	2	0	0	32	448

Table 30. Resource Usage.

Resource Level	Used	Available
Slow	15	40
Normal	25	40
Fast	10	40

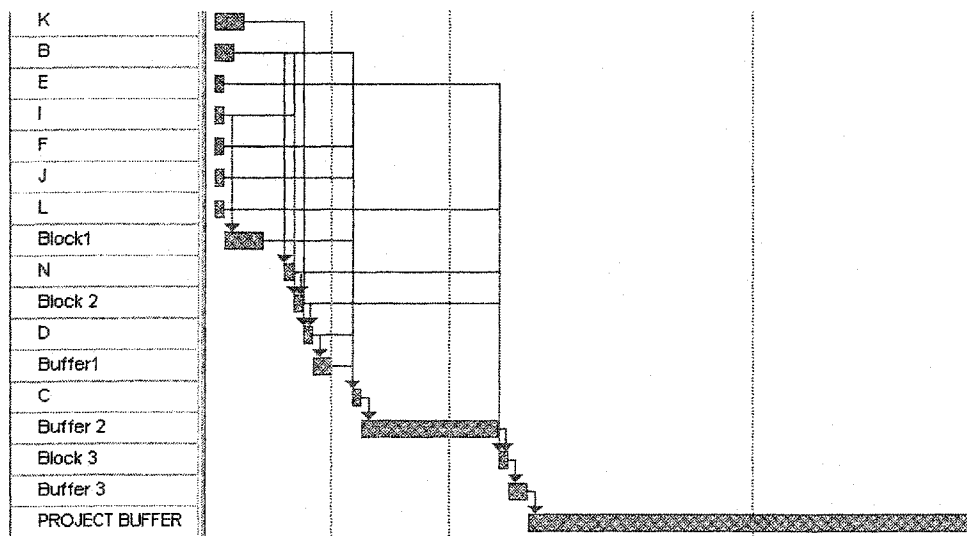


Figure 87. Gantt Chart (for 56 hours Project Duration).

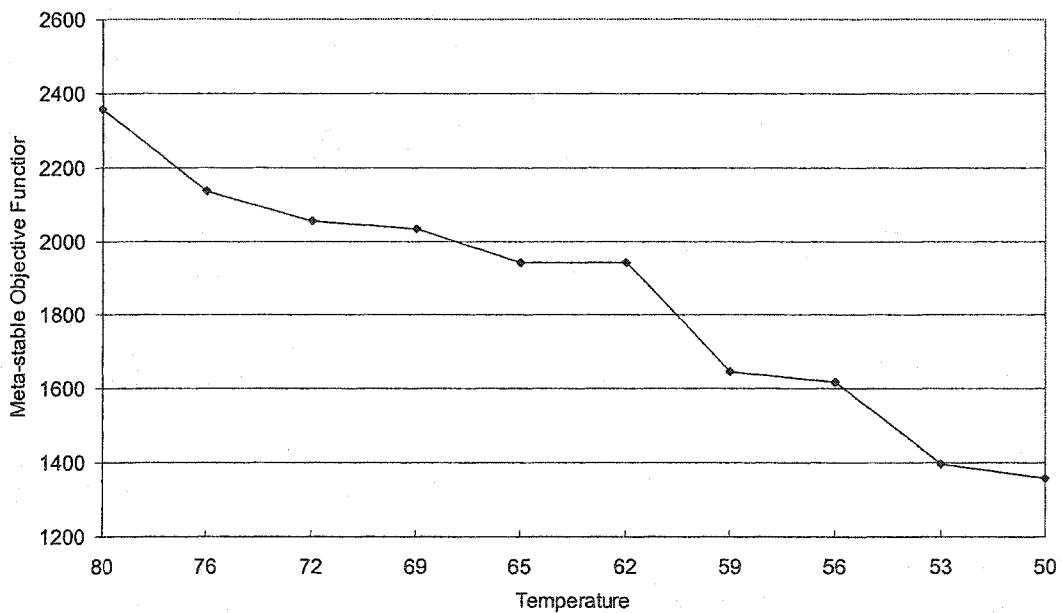


Figure 88. Meta-stable Objective Function; Cost (for 57 hours Project Duration).

## 9.5 Summary

In order to illustrate the process of the presented architecture, four case studies (projects) were presented in this chapter. The objective of considering the first case study was to benchmark the performance of optDSM. Observing the characteristics of the optimization process based on different objective functions was the goal of the second case study. The third one illustrated the incorporation of uncertainty into the model and the application of the simulation-based optimization framework. Finally, the fourth case study was to illustrate in details different steps of the presented architecture applied to a hypothetical project. A summary of these cases (projects) is given in Table 31.

Table 31. Case Projects Summary.

Project No.	Description	Case No.	Details
1	Benchmarking Project representing VTOL analysis. Consists of 22 activities and has 119 couplings.		Equal load of one unit assigned to all activities The objective is: minimizing the total load
2	A circuit taken from a large conceptual design project. Consists of 22 activities and has 39 couplings	1	Equal load of one unit assigned to all activities The objective is: minimizing the no. of feedback couplings.
		2	Equal load of one unit assigned to all activities The objective is: minimizing the total load
		3	The objective is: minimizing the total load
3	Same as (1), but with stochastic activity loads.	1	M3 method implemented
		2	UF method implemented
4	A hypothetical project. Consists of 16 activities and has 38 couplings. 2 logical constraints are imposed.		To illustrate the complete operation of the proposed architecture: a. Optimization for minimum load b. Conversion of the optimized DSM to an equivalent program. c. Implementation of the TCTO model.

## CHAPTER X

### SUMMARY AND FUTURE WORK

A product development project (PDP) fundamentally differs from a construction (or a manufacturing) project in two major aspects: (1) while the later is activity-based, the first is information-based; and (2) a typical PDP is characterized by its highly coupled, interdependent activities, which must converge iteratively to an acceptable design solution.

A PDP is typically a complex system. The main approach to handling such system involves: decomposing it into sub-systems and furthermore into smaller components; and defining the relationships among these components. Following these steps, the system will be decomposed into possibly several hundreds of activities (components) and thousand of variable interchanges among these activities. The sequence of performing these activities strongly affects the time (and hence the cost) needed to realize the whole project.

This dissertation has presented an optimization-based architecture that helps guiding the project manager efforts for managing the design process in complex integrated product development projects.

Following a sufficient background of the tools integrated through the architecture in Chapters II, III, IV, and. Chapters VI and VII provide detailed discussions on the research methodology and on research areas of contribution. Different modules of the architecture and the interactions among these modules were discussed in detail in Chapter VIII. Finally, Chapter IX attempted to present the performance of the architecture by applying it to several projects, followed by conclusions and directions to future research in Chapter X. The presented architecture was further implemented through a developed excel add-in called “optDSM” presented in Appendix A.

The presented work contributes to five areas of research:

1. Dependency Structure Matrix (DSM).
2. Optimization with the Simulated Annealing Algorithm (SA).
3. Simulation-based Optimization.
4. Time-Cost Trade-Off in Project Networks.

In the following sections, a summary of each of the contributions mentioned above is provided. Future research directions are also indicated.

### 10.1 The Dependency Structure Matrix

Improving system performance can be achieved through efficient re-engineering of its structure. The Dependency Structure Matrix (DSM) provided an effective tool for system structure understanding. The first research contribution aimed towards finding an optimal activity sequence of the DSM representing a design project in terms of load, time, and cost. To achieve this goal, a mathematical program (model) representing the DSM structure was developed and a meta-heuristic optimization algorithm called Simulated Annealing (SA) was implemented to solve this model.

Since its original introduction by Steward in the 1980's, the DSM has proved itself as an effective tool for analyzing and understanding system architecture especially in new product development projects and, hence, achieving improved performance. The use of the DSM is the corner stone of the architecture proposed in the current research. Following the modeling of the system (project) in a DSM format, finding its optimal activity sequence represented the first contribution of the research presented.

- This research implemented a numerical DSM in which coupling marks are replaced with numbers (iteration factors) indicating the strength of the coupling.
- The DSM was then represented by a mathematical program (model) so that it can be mathematically optimized. The model, further, allowed:
  - Imposing logical constraints on activity sequence.
  - The choice among four objective functions.

- The use of stochastic activity data (load, time and/or cost).
- Although the optimization process considered planned iterations only, unplanned iterations can be compensated for in the scheduling phase through buffers introduced in the scheduling phase.
- The model assumed sequential execution of activities and did not allow for activity concurrency or overlapping. Further investigation can consider cases in which activities can start without receiving all required input information.
- Furthermore, cases in which an activity can provide some output to other activities before it finishes can also be incorporated in a future model.
- The fashion by which an activity is redone when it falls in more than one feedback loop can be modified to reflect more applicable situations.
- The introduction of logical constraints to the model provided control means over the activity sequence of the DSM. Moreover, such constraints can be used when considering multi-DSM (or multi-project) cases. In such cases, these constraints can be set to maintain some logic activity order, or to represent resource constraints. So, multi-DSM can be optimized through the same optimization module developed in the dissertation.

## 10.2 Simulated Annealing

To carry out the optimization process, a meta-heuristic algorithm called Simulated Annealing (SA) was implemented. SA is a stochastic optimization algorithm that provides global or near-global optimal solutions for a wide variety of hard combinatorial optimization problems. SA was used in this research to rapidly examine a large number of configurations and choose the optimal one.

- The SA proposed and implemented in this research (referred to as ‘two-stage SA’) was modified from the original SA by adding a stage that keeps record of the best solution throughout the optimization process.

- Since SA is a robust algorithm, initial solution was chosen to be generated randomly in the implementation presented. Other options that can be considered are:
  - Allowing the user to provide the initial solution.
  - Generating the initial solution using a partitioning algorithm.
- Only one cooling schedule – geometric - was used in this research. Other schedules (such as multi-stage cooling and geometric re-heating) can be used and have their results compared.
- In the implemented SA, the system is assumed to reach a meta-stable state when either of two defined counter reaches its threshold (specified by the user). And the optimization process was set to stop when the final temperature is reached. Although these settings provided adequate results, other options can be further implemented such as allowing the optimization process to stop when no improvement of the objective function for a long time is noticed.
- When generating a neighboring solution, only feasible solutions (which satisfies logical constraints if any) were accepted for evaluation. A future research may consider providing a mechanism that allows some infeasible moves to be considered.

### **10.3 Simulation-based Optimization**

One unavoidable pitfall in the estimation of activity time and cost is uncertainty which arises from many different sources. Although uncertainty cannot be eliminated, incorporating it in the model can reduce its effect. Thus, the estimate of activity time and cost can be in the form of a probability distribution not as a single point value. The second research area concerned handling uncertainty in activity loads, time, and/or cost. A simulation-based optimization framework that integrates simulated annealing with a commercial risk analysis software called Crystal Ball<sup>TM</sup> was developed as a part of the proposed architecture to optimally re-sequence the DSM activities given stochastic activity data.



- Monte Carlo (MC) simulation was used to determine the statistical measures of the objective function at each optimization iteration. As known, MC is a time consuming method. A future research, thus, may consider implementing an analytical model to determine these measures with accepted margin of error for situations where an exact minimum solution is not required and a satisfactory estimation is accepted.

#### **10.4 Handling Stochastic Objective Function**

Since simulated annealing was originally developed to handle deterministic objective functions, the third research area involved modifying the SA algorithm to be able to handle stochastic objective functions (multi-point estimate) rather than deterministic ones (one-point estimate). The goal here involved determining a robust solution rather than an optimum minimum one. This was achieved by modifying the acceptance and rejection rules of the SA algorithm. Two methods (sets of rules) were proposed and tested, these are:

- Min-Mean-Max (M3) method:
  - Can be considered as a simple expert system.
  - It should be noted, though, that the cases presented in the dissertation are not exhaustive. They can be modified and other cases can be added.
  - The degree to which the decision maker accepts/tolerates risk is reflected through the choice of the ratios used in the acceptance rules (flow charts) presented. Demonstrates how project manager experience can be translated into a set of rules to modify an existing optimization algorithm and make it suitable to the area of application.
  - Demonstrates how project manager experience can be translated into a set of rules to modify an existing optimization algorithm and make it suitable to the area of application.

- Utility Function (UF) method:
  - Decision-making is based, mainly, on appraisal; different alternatives are appraised and compared against each other to choose the one with best expected outcome. The UF method provided a means by which alternative solutions can be compared based on the expected value of four of their attributes. These attributes, representing four statistical measures of the objective function distribution, are: mean, variance, range, and maximum.
  - By defining the weights associated with these attributes in the UF, the decision maker can direct the optimization process to find either a minimum solution or a robust one.
  - The method is characterized by:
    1. Being simple and direct.
    2. Ease of attributes management (fine-tuning).
    3. Being familiar to most decision makers and analysts.
    4. Easy to modify.
    5. Easy to implement.

### **10.5 DSM Conversion to a Project Schedule**

For more than fifty years, critical path methods (CPM) have provided efficient tools for planning, scheduling, and controlling constructions and manufacturing projects. But since CPM does not tolerate for feedback loops usually found in design project, another tool that explicitly handle these loops was developed, that is DSM. But, on the other hand, for the DSM to serve as a means of control of the design projects (continual re-planning, re-scheduling, and follow-up) activities in the optimally re-sequenced DSM need to be represented against a time scale. In other words, the DSM has to be converted into a project schedule. The fourth contribution of this research was providing a methodology for the conversion of the optimally sequenced DSM into an equivalent DSM that contains no feedback couplings. Once an equivalent DSM is obtained, a project schedule can be developed and the use of scheduling methods becomes feasible.

- The presented research applied a methodology in which project scheduling is done in isolation from DSM analysis (two successive phases). A future research may consider the possibility of performing the CPM – or a modified CPM - calculations within the DSM paradigm.
- The presented research suggested a three-stage methodology for this conversion. The methodology was inspired by the work of several researchers and can be the basis for more generalized rules. It consists of three sequential yet integrated stages: patterns recognition, collapsing, and tearing.
- The basic idea of the methodology is to identify some activity patterns; each pattern has only one feedback coupling, from two to four activities, and at least one feed forward coupling, in the DSM. Based on the strength of the couplings involved in each pattern (block), the block is converted to an equivalent program that contains no feedback loops.
- The methodology further involves introducing two types of buffers to the generated program: block (or coupling) buffers, and a project buffer.
- Future research may include more elaborated conversion rules and can provide procedure for buffer generation and management.

## 10.6 TCTO Hybrid Model

Finally, the fifth and final area presented a new time-cost trade-off model for project networks. The new model is a hybrid model that joins the resource assignment problem with project crashing. The presented model is based on the trade-off of resources where, in some cases, it may be possible to transfer men, equipment, or other resources from a non-critical activity to a critical one. Thus, it helps crashing a project with little, or no, additional cost.

- The presented model differs from classical CPM time-cost trade-off model in six major aspects discussed in Section 6.4.
- Since the model is in its initial development phase, the pattern used for resource availability constraints was simplified. It was assumed that once a

machine is assigned to an activity, it cannot be used again. In future work, an algorithm for resource-constrained projects algorithm could be incorporated.

- Furthermore, the model could be extended to include fixed costs.
- Also, the introduction of additional logical constraints can be further investigated.

## 10.7 optDSM

The former areas of research were applied through a developed excel add-in called “optDSM”. The tool was developed by the author using Visual Basic for Application (VBA) programming language. Among its several modules, optDSM has the ability to interface with Crystal Ball™ to carry out the optimization process in cases where activity loads assumes stochastic values. The main functions of optDSM are:

1. Modeling of the project under consideration in the form of DSM.
2. Finding the optimum sequence of DSM activities based on a user selected objective function.
3. Producing a DSM equivalent to the optimized one but without feedback couplings.
4. Converting the structured DSM into a project schedule.

While the first two functions are fully operational, the later two are still under development.

- Quality of solution obtained. In spite of the impossibility to benchmark in practice (since the true global optima is not known) the performance of optDSM was evaluated through the optimization of a famous example and comparing the results to a published one. It was found that optDSM reached an equivalent solution to the published one of AGENDA. Refer to Section 9.1.
- Why Excel? Excel has several benefits:
  1. It is powerful modeling tool to implement mathematical programs (models):
    - a. Quick to build and easy to modify (flexibility).

- b. Has a great range of built-in mathematical and scientific functions.
  - c. Availability of logical statements and decisions.
  - d. Capability of host large, and complex models.
  - e. Automatically provides 'what-if' tools which makes it suitable for combinatorial optimization problems.
2. With VBA, it can be an integrated platform that meets different analysis requirements by communicating with other applications within Micro Soft windows environment.
  3. Availability of different add-ins (for optimization, simulation, statistical analysis, etc.).
  4. A familiar and easy to understand interface for both technical and non-technical persons.
- Execution time. The total execution (run) time depends on several factors:
    - Number of couplings in the DSM.
    - Presence of logical constraints.
    - Optimization nature: if stochastic, another two factors are introduced:
      - i. The number of Monte Carlo simulation runs per optimization iteration.
      - ii. Sensitivity analysis requirements.

### 10.9 Lessons learned

1. Special-purpose optimization tools still have advantage over general-purpose tools.
2. Optimization models should allow the use to monitor and fine tune the optimization process during processing.
3. Fine-tuning of optimization algorithm parameters is a very important factor of the quality of solution.
4. In some cases, constraints relaxation can help the optimization process finding a better initial solution. For example, the solution procedure for the TCTO model included options for:

- a. Relaxation of the resources constraints so that the algorithm can generate a feasible solution in a shorter time, with the optimization process advances, the algorithm tends to lower the consumption of resources (to lower costs), as a result, resource usage will go down to below the real availability limits.
- b. Relaxation of “desired project completion time” (DPCT) constraint by allowing the user to specify an upper and lower limit. For example, if DPCT was originally required to be 49 hours, the user can relax this constraint to be in the range (47-50) of (45-51) and so on. Because, in the proposed TCTO model, it might be difficult (or even infeasible) to achieve exactly 49 hours and a better solution can be achieved for 48 hours (shorter duration).

## REFERENCES

- [1] Fiksel, J., 1991, *Design for Environment: Creating Eco-Efficient Products & Processes*, McGraw-Hill, N.Y.
- [2] Chakravarty, A. K., 2001, "Overlapping Design and Build Cycles in Product Development," *European Journal of Operational Research*, **134**, pp. 392-424.
- [3] Kotler, P., 7<sup>th</sup> ed. 1991, *Marketing Management: Analysis, Planning, Implementation and Control*, Prentice-Hall, Englewood Cliffs, NJ.
- [4] Stalk, G., 1988, "Time: The Next Source of Competitive Advantage," *Harvard Business Review*, **66**, July-August 41-51.
- [5] Blackburn, J. D., 1991, "New Product Development: The New Time Wars," In: Blackburn, J.D., (Ed.), *Time-Based Competition*. Business One Irwin Publishers, Homewood, IL.
- [6] House, C.H., and Price, R. L., 1991, "The Return Map: Tracking Product Teams," *Harvard Business Review*, January-February, 92-101.
- [7] National Research Council, 1991, "Improving Engineering Design: Designing for Competitive Advantage," National Academy Press, Washington, DC.
- [8] Peet, W. J., and Hladik, K. J., 1989, "Organizing for global product development," *Electronic Business*, **15** (5), pp. 62-64.
- [9] Yassine, A., Chelst, K., and Falkenburg, D., 1999, "Engineering Design Management: An Information Structure Approach", *International Journal of Production Research*, **37**(13), pp. 2957-2975.
- [10] Lindemann, U., Bichlmaier, C, Stetter, R., Viertlböck, M., 1999, "Enhancing the Transfer of Integrated Product Development in Industry," In: Lindemann, U.; Birkhofer, H.; Meerkamm, H.; Vajna, S. (Eds.): *Proc. of the 12th Intern. Conference on Engineering Design ICED 1999*, Vol. 1, München, 24.-26.08.1999. München: TU 1999, S. 373-376. (Schriftenreihe WDK 26)
- [11] Lindemann, U., Stetter, R., and Viertlböck, M., 2001, "A Pragmatic Approach for Supporting Integrated Product Development," *Transactions of the Society for Design and Process Science*, **5**(2), pp. 39-51.
- [12] Armstrong, S. C., 2001, *Engineering and Product Development Management: The Holistic Approach*, Cambridge University Press.
- [13] Port, O., 1990, "A smarter way to manufacture," *Business Week*, April 30, 1990, pp.110-117.
- [14] Nevins, J. L., and Whitney, D. E. (eds), 1989, *Concurrent Design of Products and Processes: A Strategy for the Next Generation in Manufacturing*, McGraw-Hill, New York.
- [15] O'Grady P., Ramers D. and Bowen J., 1988, "Artificial Intelligence Constraint Nets Applied to Design for Economic Manufacture and Assembly," *Computer Integrated Manufacturing Systems*, **1**(4), 204-210.
- [16] Olesen, J. and Keldmann, T., 1993, "Design for Environment - A Framework," 9th International Conference on Engineering Design, The Hague, Netherlands, HEURISTA, pp. 747-754.
- [17] Andreasen, M. M., and Hein, L., 1987, *Integrated Product Development*, IFS Publications Ltd, Bedford.

- [18] Andreasen, M. M., and Hein, L., 1998, "Innovating the Product Development Organisation," in Frankenberger, E., Badke-Schaub, P., and Birkhofer, H. (Eds.): *Designers: the Key to Successful Product Development*, Springer, pp. 183-195.
- [19] Daellenbach, H. G., 1994. *Systems and Decision Making: A Management Science Approach*. John Wiley and Sons, New York.
- [20] Baldwin, A. N., Austin, S. A., Hassan, T. M., and Thorpe, A., 1998, "Planning Building Design by Simulating Information Flow," *Automation in Construction*, 8, pp. 149-163
- [21] Johnson, E. W., Castillo, L. A., and Brockman, J. B., 1996, "Application of a Markov Model to the Measurement, Simulation, and Diagnosis of an Iterative Design Process," *Proceedings of the 33<sup>rd</sup> annual conference on Design Automation Conference*, Las Vegas, Nevada, United States, pp.185 – 188. ACM Press, New York, NY, USA
- [22] Steward, D. V., 1981, *Systems Analysis and Management: Structure, Strategy, and Design*, PBI, NEW York.
- [23] Browning, T. R., 2001, "Modeling the Customer Value of Product Development Processes," *Proceedings of the 11<sup>th</sup> Annual International Symposium of INCOSE*, Melbourne, Australia, July 1-5
- [24] Browning, T. R., and Eppinger, S. D., 2002, "Modeling Impacts of Process Architecture on Cost and Schedule Risk in Product Development," *IEEE Transaction on Engineering Management*, 49(4), pp.443-458.
- [25] Von Hippel, E., 1990, "Task Partitioning: An Innovation Process Variable," *Research Policy*, 19, pp. 407-418.
- [26] Browning, T. R., 2001, "Applying the Design Structure Matrix to System Decomposition and Integration Problems: A Review and New Directions," *IEEE Transactions on Engineering Management*, 48 (3), pp.292-306.
- [27] Browning, T. R., 1998, "Modeling and Analyzing Cost, Schedule, and Performance in Complex System Product Development," Ph.D. dissertation, MIT, Cambridge, MA.
- [28] Browning, T. R., 1999, "The Design Structure Matrix," in *Technology Management Handbook*, R. C., Dorf. Ed., Boca Raton, Fl: Chapman & Hall/CR CnetBASE, pp. 103-111.
- [29] Denker, S., Steward, D. V., and Browning, T. R., 1999, "Planning Concurrency and Managing Iteration in Projects," *Project Management Journal*, 32 (3): 31-38, 2001.
- [30] Kerzner, H., 1989, *Project Management*, Van Nostrand Reinhold, New York.
- [31] Steward, D. V., 1981, "The Design Structure System: A Method for Managing the Design of Complex Systems," *IEEE Transactions on Engineering Management*, 28, pp. 71-74.
- [32] Gebala, David A. and Eppinger, Steven D., 1991, "Methods for Analyzing Design Procedures", *Proceedings of the ASME Third International Conference on Design Theory and Methodology*, pp. 227-233.
- [33] Warfield, John N., 1973, "Binary Matrices in System Modeling," *IEEE Transactions on Systems, Man, and Cybernetics*, 3, pp. 441-449.
- [34] Kusiak, N. Larson, and J. Wang, 1994, "Reengineering of Design and Manufacturing Processes," *Computers and Industrial Engineering*, 26 (3), pp. 521-536.
- [35] Austin, S., Baldwin, A., Li, B., and Waskett, P., 2000, "Application of the Analytical Design Planning Technique to Construction Project Management," *Project Management Journal*, 31 (2), pp. 48-59.



- [36] Rogers, J. L., 1989, "Knowledge-Based Tool for Multilevel Decomposition of a Complex Design Problem," NASA TP-2903.
- [37] Rogers, J. L., 1989, "DeMaid – A Design Manager's Aid for Intelligent Decomposition User's Guide," NASA TM-101575.
- [38] Rogers, J. L. and Padula S. L., 1989, "An Intelligent Advisor for the Design Manager," Proceeding, the First International Conference on Computer Aided Optimum Design of Structures, Southampton, UK, pp. 169-177.
- [39] Rogers, J. L., 1996, "DeMaid/GA – An Enhanced Design Manager's Aid for Intelligent Decomposition," 6<sup>th</sup> AIAA/USAF/NASA/OSSMO Symposium on Multidisciplinary Analysis and Optimization, Seattle, WA, September 4-6, 1996. AIAA Paper No. 96-4157.
- [40] Rogers, J. L., and Barthelemy, J-F, 1992, "Enhancements to the Design Managers Aid for Intelligent Decomposition (DeMAID)," Proceedings, 4th AIAA/Air Force/NASA/OAI Symposium on Multidisciplinary Analysis & Optimization. Cleveland, Ohio.
- [41] Rogers, J. L., and Bloebaum, C. L., 1994, "Ordering Design Tasks Based on Coupling Strengths," Proceeding, the 5th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Panama City, Florida. AIAA paper No. 94-4326. Also NASA TM 109137.
- [42] Rogers, J. L., McCulley, C. M., and Bloebaum, C. L., 1996, "Integrating a Genetic Algorithm Into a Knowledge-Based System for Ordering Complex Design Processes," Proceedings, the 96 Artificial Intelligence on Design Conference, Stanford University, CA. Also NASA TM - 110247.
- [43] Austin, S., Baldwin, A., Li, B., & Waskett, P., 2000, "Analytical design planning technique (ADePT): A Dependency Structure Matrix Tool to Schedule the Building Design Process," Construction Management and Economics, **8**, 173-182.
- [44] Austin, S., Baldwin, A., Li, B., & Waskett, P., 1999, "Analytical Design Planning Technique: A Model of the Detailed Building Design Process," Design Studies, **20** (3), 279-296.
- [45] Austin, S., Baldwin, A., Li, B., & Waskett, P., 1999, "Analytical Design Planning Technique for Programming the Building Design Process" Proceedings of the Institution of Civil Engineers, Structures and Building, **134** (2), 111-118.
- [46] Baldwin, A. N., Austin, S. A., Hassan, T. M., and Thorpe, A., 1998, "Planning Building Design by Simulating Information Flow," Automation in Construction, **8**, pp. 149-163
- [47] Altus, S. S., Kroo, I. M., and Gage, P. J. 1996, "A Genetic Algorithm for Scheduling and Decomposition of Multidisciplinary Design Problems," Transactions of the ASME, **118** (4), pp. 486-489.
- [48] Cho, S-H, 2001, "An Integrated Method For Managing Complex Engineering Projects Using the Design Structure Matrix and Advanced Simulation," M.S. Thesis, MIT.
- [49] Eppinger, S. D., and Smith, R. P., 1997, "A Predictive Model of Sequential Iteration in Engineering Design," Management Science, **43** (8), pp. 1104-1120.
- [50] Smith, R. P., and Eppinger, S. D., 1997, "Identifying Controlling Features of Engineering Design Iteration," Management Science, **43** (3), pp. 276-293.

- [51] Carrascosa, M., Eppinger, S.D., and Whitney, D.E., 1998, "Using the Design Structure Matrix to Estimate Product Development Time," Proceedings of DETC'98, 1998 ASME Design Engineering Technical Conferences, Sept. 13-16, 1998, Atlanta, Georgia, USA.
- [52] Tang, D., Zheng, L., Li, Z., Li, D., and Zhang, S., 2000, "Re-engineering of the Design Process for Concurrent Engineering," *Computers & Industrial Engineering*, **38**, pp. 479-491.
- [53] Chun-Hsien Chen, Shih Fu Ling, Wei Chen, 2003, "Project Scheduling for Collaborative Product Development Using DSM," *International Journal of Project Management*, **21** (4), pp. 291-299.
- [54] Shi-Jie (Gary) Chena, Li Lin, 2003, "Decomposition of Interdependent Task Group for Concurrent Engineering," *Computers & Industrial Engineering*, **44** (3), pp. 435-459.
- [55] Rao, S. S., 3<sup>rd</sup> edition 1996, *Engineering Optimization: Theory and Practice*, John Wiley & Sons, Inc, New York.
- [56] Pike, R. W., 1986, *Optimization for Engineering Systems*, Von Nostrand Reinhold Company, New York.
- [57] Reklaitis, G. V., Ravindran, A., and Ragsdell, K. M., 1983, *Engineering Optimization: Methods and Applications*, John Wiley & Sons, Inc, New York.
- [58] Osman, I. H., & Kelly, J. P., 1996, Meta-heuristics: an Overview, O In: I. H. Osman, & J. P. Kelly, Meta-heuristics: theory and application. Boston: Kluwer Academic Publishers.
- [59] Metropolis, W., Roenbluth, A., Rosenbluth, M., Teller, A., and Teller, E., 1953, "Equation of the State Calculations by Fast Computing Machines," *Journal of Chemical Physics*, **21**, 1087-1092.
- [60] Kirkpatrick, S., Gelatt, C. D. Jr., and Vecchi, M. P., 1982, "Optimization by Simulated Annealing," IBM Research Report, RC 9355
- [61] Kirkpatrick, S., Gelatt, C.D., and Vecchi, P.M., 1983, "Optimization by Simulated Annealing," *Science*, **220** (4598), PP.671-680.
- [62] Cerny, V., 1985, "Thermodynamical Approach to the Traveling Salesman Problem: an Efficient Simulation Algorithm," *Journal of Optimization Theory and Applications*, **45** (1), PP.41-51.
- [63] Adil, G. K., D. Rajamni, and Strong, D., 1997, "Assignment Allocation and Simulated Annealing Algorithms for Cell Formulation," *IIE Transactions*, **29** (1), 53-67.
- [64] Gemmill, D. D., and Tsai, Y. W., 1997, "Using a Simulated Annealing Algorithm to Schedule Activities of Resource-Constrained Projects," *Project Management Journal*, **28** (4), PP.8-20.
- [65] Kim, J. U., and Kim, Y. D., 1996, "Simulated Annealing and Genetic Algorithms for Scheduling Products with Multi-level Product Structure," *Computers and Operations Research*, **23** (9), pp. 857-868.
- [66] Kuik, R., and Salomon, M., 1990, "Multi-level Lot Sizing Problems: Evaluation of a Simulated Annealing Heuristic," *European Journal of Operational Research*, **45** (1), pp.25-37.
- [67] Kuik, R., Salomon, M., van Wassenhove, L. N., and Maes, J., 1993, "Linear Programming, Simulated Annealing and Tabu Search Heuristics for Lot Sizing in Bottleneck Assembly Systems," *IIE Transactions*, **25** (1), pp.62-72.

- [68] Khan, Z., Prasad, B., and Singh, T., 1997, "Machining Condition Optimization by Genetic Algorithms and Simulated Annealing," *Computers and Operations Research*, 24 (7), pp. 647-657.
- [69] Koulamas, C., Antony, S. R., and Jean, R., 1994, "A survey of Simulated Annealing Applications to Operations Research Problems," *Omega*, 22 (1), pp.41-56.
- [70] Vanderbilt, D., & Louie, S. G., 1984, "A Monte Carlo Simulated Annealing Approach to Optimization Over Continuous Variables," *Journal of Computational Physics*, 56, pp.259- 271.
- [71] Parks, G. T., 1990, "An intelligent Stochastic Optimization Routine for Nuclear Fuel Cycle Design," *Nuclear Technology*, 89, pp.233-246.
- [72] Lin, S., 1965, "Computer solutions of the traveling salesman problem," *Bell System Technical Journal*, 44, pp.2245-2269.
- [73] Bounds, D. G., 1987, "New Optimization Methods From Physics and Biology," *Nature*, 329, pp.215-218.
- [74] Kirkpatrick, S., 1984, "Optimization by Simulated Annealing - Quantitative Studies," *Journal of Statistical Physics*, 34 (5/6), pp.975-986.
- [75] Randelman, R. E., and Grest, G.S., 1986, "N-City Traveling Salesman Problem-Optimization by Simulated Annealing," *Journal of Statistical Physics*, 45, pp.885-890.
- [76] Van Laarhoven, P. J. M., and Aarts, E. H. L., 1987, *Simulated Annealing: Theory and Applications*, D. Reidel publishing Company, Holland.
- [77] Elmohamed, M.A. S., Fox, G., and Coddington, P., 1998, "A Comparison of Annealing Techniques for Academic Course Scheduling," NPAC Technical Report SCCS-777.
- [78] Johnson, D., and L. McGeoch, 1997, "The Traveling Salesman Problem: A Case Study in Local Optimization," in *Local Search in Combinatorial Optimization*, E. H. Aarts and J. K. Lenstra (eds.), Wiley and Sons.
- [79] Ingber, L., 1993, " Simulated Annealing: Practice versus Theory," *Computer Mathematical Modeling*, 18 (11), pp.29-57.
- [80] A. Charnes and M. Wolfe, 1989, " Extended Pincus Theorems and Convergence of Simulated Annealing," *International Journal of Systems Science*, 20 (8), pp.1521-1533.
- [81] Encarta® World English Dictionary [North American Edition] © & (P) 2001 Microsoft Corporation. All rights
- [82] Naylor, T. J., Balintfy, J. I., Burdick, D. S., and Chu, K., 1966, "Computer Simulation Techniques," Wiley, New York.
- [83] Moore, L. J., and Clayton, E. R., 1<sup>st</sup> ed. 1976, "GERT Modeling and Simulation: Fundamentals and Applications," Petrocelli/Harter, New York.
- [84] Law, A. M., and Kelton, W. D., 3<sup>rd</sup> ed. 2000, *Simulation Modeling and Analysis*, McGraw-Hill, Boston.
- [85] Rubinstein, R. Y., and Melamed, B., 1998, *Modern Simulation and Modeling*, John Wiley & Sons, Inc, New York.
- [86] Chase, R. B., and Aquilano, J. J., 7<sup>th</sup> ed.1995, *Production and Operations Management: Manufacturing and Services*, Irwin, Chicago.
- [87] Stevenson, W. J., 6<sup>th</sup> ed.1999, *Production/Operations Management*, Irwin/McGraw-Hill, New York
- [88] Good, I.J. (1994). Reliability always depends on probability of course, *Journal of Statistical Computation and Simulation* 52, pp.192-193.

- [89] Smith, E.P., 2002, "Uncertainty Analysis," in A.H. El-Shaarawi and W.W. Piegorsch, eds, *Encyclopedia of Environmentrics*, John Wiley & Sons, Ltd, Chichester. pp. 2283-2297.
- [90] Metropolis, N., and Ulam, S., 1949, "The Monte Carlo Method," *Journal of the American Statistical Association*, **44** (247), pp. 335-41.
- [91] Sobol, I. M., 1974, *The Monte Carlo Method*, The University of Chicago Press, Chicago.
- [92] Suri, R., 1985, "An Overview of Evaluative Models for Flexible Manufacturing Systems," *Annals of Operations Research*, **3**, pp. 13-21.
- [93] Paul, R. J., and Chaney, T. S., 1998, "Simulation Optimization Using a Genetic Algorithm," *Simulation Practice and Theory*, **6** (6), pp. 601-611.
- [94] Andradottir, S., 1992, "A Stochastic Approximation Algorithm with Varying Bounded," Technical Report 92-5, Department of Industrial Engineering, University of Wisconsin-Madison, Madison, WI.
- [95] Andradottir, S., 1992, "Discrete Optimization in Simulation: A Method and Application," In *Proc. 1992 Winter Simulation Conf.* (Edited by J. J. Swain, D. Goldsman, R. C. Crain and J. R. Wilson), pp. 483-486.
- [96] Evans, G.W., Stuckman, B., and Mollaghasemi, M., 1991, "Multicriteria Optimization of Simulation Models," *Proceedings of the 23rd conference on Winter simulation 1991*, Phoenix, Arizona, United States B.L. Nelson, W.D. Kelton, G.M. Clark (Eds.), pp.894-900.
- [97] Stuckman, B., Evans, G., and Mollaghasemi, M., 1991, "Comparison of Global Search Methods for Design Optimization Using Simulation," *Proceedings of the 23<sup>rd</sup> conference on Winter Simulation*, Phoenix, Arizona, United States B.L. Nelson, W.D. Kelton, G.M. Clark (Eds.), pp.937-944.
- [98] Azadivar, F., 1992, "A Tutorial on Simulation Optimization," *Proceedings of the 24<sup>th</sup> Conference on Winter Simulation* Arlington, Virginia, United States (Edited by J. J. Swain, D Goldsman, R. C. Crain and J. R. Wilson), pp. 198-204.
- [99] Akbay, K. S., 1996, "Using Simulation Optimization to Find the Best Solution," *IIE Solutions*, **28** (5), pp.24-29.
- [100] Greenwood, A.G., Rees, L.P., Crouch, I.W.M., 1993, "Separating the Art and Science of Simulation Optimization: A Knowledge-based Architecture Providing Machine Learning," *IIE Transactions*, **25** (6), pp. 70-84
- [101] Arsham, H., 1996, "Stochastic Optimization of Discrete Event Systems Simulation," *Microelectronics and Reliability*, **36** (10), pp. 1357-1368.
- [102] Meketon, M. S., 1987, "Optimization in Simulation: A Survey of Recent Results," *Proceedings of the 1987 Winter Simulation Conference* (Edited by A. Thesen, H. Grant and W. David Kelton), pp. 58-67.
- [103] Yunker, J. M., and Tew, J. D., 1994, "Simulation Optimization by Genetic Search," *Mathematics and Computers in Simulation*, **37** (1), pp. 17-28.
- [104] Azzaro-Pantel, C., Bernel-Haro, L., Baudet, P., Domenech, S., and Pibouleau, L., 1998, "A Two-Stage Methodology for Short-Term Batch Plant Scheduling: Discrete-Event Simulation and Genetic Algorithm," *Computers Chem. Engng*, **22** (10), pp. 1461-1481.

- [105] Azadivar, F., and Tompkins, G., 1999, "Simulation Optimization with Qualitative Variables and Structural Model Changes: A Genetic Algorithm Approach," *European Journal of Operational Research*, **113** (1), pp. 169-182.
- [106] Glover, F., Kelly, J. P., and Laguna, M., 1996, "New Advances and Applications of Combining Simulation and Optimization," *Proceedings of the 1996 Winter Simulation Conference* J. M. Charnes, D. J. Morrice, D. T. Brunner, and J. J. Swain (Eds.), pp. 144-152
- [107] Laguna, M., 1997, "Metaheuristic Optimization with Evolver, Genocop and OptQuest," *EURO/INFORMS Joint International Meeting 1997 Plenaries and Tutorials*, J. Barcelo (Ed.), pp. 141-150
- [108] Bulgak, A. A., and Sanders, J. L., 1988, "Integrating a Modified Simulated Annealing Algorithm with the Simulation of a Manufacturing System to Optimize Buffer Sizes in Automatic Assembly Systems," *Proceedings of the 20th conference on Winter simulation*, San Diego, California, United States. pp. 684 - 690
- [109] Gelfand, S.B., and Mitter, S.K., 1989, "Simulated Annealing with Noisy or Imprecise Energy Measurements," *Journal of Optimization Theory and Applications* **62** , pp. 49-62.
- [110] Roenko, N., 1990, "Simulated Annealing under Uncertainty," Technical Report, Inst. F. Operations Research, Univ. Zurich, 1990.
- [111] Haddock, J, and Mittenthal, J., 1992, "Simulation Optimization Using Simulated Annealing," *Computers & Industrial Engineering*, **22** (4), pp. 387-395
- [112] Alkhamis, T. M., Ahmed, M. A., and Tuan, V. K., 1999, "Simulated Annealing for Discrete Optimization with Estimation," *European Journal of Operational Research* **116** (3), pp.530-544.
- [113] So, D.G., and Dowsland, K.A., 1993, "Simulated Annealing: An Application to Simulation Optimization," Presented at OR35 Conference, University of York, September 1993.
- [114] Gutjahr, W.J., and Pflug, G.C., 1996, "Simulated Annealing for Noisy Cost Functions," *Journal of Global Optimization*, **8** (1), pp. 1-13.
- [115] Ahmed, M. A., Alkhamis, T.M., and Hasan, M., 1997, "Optimizing Discrete Stochastic Systems Using Simulated Annealing and Simulation," *Computers & industrial Engineering*, **32** (4), pp. 823-836.
- [116] Ahmed, M. A., and Alkhamis, T. M., 2002, "Simulation-based optimization using simulated annealing with ranking and selection," *Computers & Operations Research*, **29**, pp.387-402.
- [117] Ho, Y. C., 1984, "Perturbation Analysis Methods for Discrete Event Dynamically Systems and Simulations," *Proceedings of the 1984 Winter Simulation Conference*, pp. 171-173.
- [118] Ho, Y. C. and Cao, X. R., 1991, *Perturbation Analysis of Discrete Event Dynamic Systems*, Kluwer Academic, Dordrecht.
- [119] Jacobson, S. H. and Schruben, L. W., 1989, "Techniques for simulation response optimization," *Operations Research Letters*, **8**, pp.1-9.
- [120] Schruben, L. W., 1986, "Simulation Optimization Using Frequency Domain Methods," *Proceedings of the 1986 Winter Simulation Conference* (Edited by J. R. Wilson, J. O. Henriksen and S. D. Roberts), pp. 366-369.

- [121] Rubinstein, R. Y., 1989, "Sensitivity analysis and performance extrapolation for computer simulation methods," *Operations Research*, **37**, pp.72-81.
- [122] Barton, R. R., and Ivey, J. S. Jr, 1996, "Nelder-Mead Simplex Modifications for Simulation Optimization," *Management Science*, **42** (7), pp.954-973.
- [123] Glynn, P. W., 1987, "Likelihood Ratio Gradient Estimation: An Overview," *Proc. 1987 Winter Simulation Conference* (Edited by A. Thesen, H. Grant and W. David Kelton), pp. 366-375.
- [124] Pukkala, T., and Miina, J., 1997, "A Method for Stochastic Multiobjective Optimization Of Stand Management," *Forest Ecology and Management*, **98**, pp.189-203
- [125] Guariso, G., Hitz, M., and Werthner, H., 1996, "An Integrated Simulation and Optimization Modeling Environment for Decision Support," *Decision Support Systems*, **16**, pp. 103-117.
- [126] Biles, W. E., 1975, "A Response Surface Method for Experimental Optimization of Multiresponse Processes," *Industrial Engineering Chemistry: Process Design and Development*, **14**, pp.152-158.
- [127] Smith, D. E., 1976, "Automatic Optimum-Seeking Program for Digital Simulation," *Simulation*, **27**, pp.27-32.
- [128] Wilson, J. R., 1987, "Future direction in response surface methodology for simulation," *Proceedings of the 1987 Winter Simulation Conference* (Edited by A. Thesen, H. Grant and W. David Kelton), pp. 378-381.
- [129] Boyle, C. R., and Shin, W. S., 1996, "An Interactive Multiple-Response Simulation Optimization Method," *IIE Transactions*, **28** (6), pp. 453-463.
- [130] Lee, Y-H, Shin, H-M, and Yang, B-H, 1996, "An Approach For Multiple Criteria Simulation Optimization with Application to Turning Operations," *Journal of Computers and Industrial Engineering*, **30** (3), pp.375-386.
- [131] Greenwood, A. G., Rees, L. P., and Siochi, F. C., 1998, "An Investigation of the Behavior of Simulation Response Surfaces," *European Journal of Operational Research*, **110** (2), pp. 282-313.
- [132] Rees, L. P., Greenwood, A. G., and Siochi, F. C., 2002, "A Best-First Search Approach for Determining Starting Regions in Simulations Optimization," *IIE Transactions*, **34** (3), pp. 283-295.
- [133] Robbins, H. and Monroe, S., 1951, "On a Stochastic Approximation Technique," *Annals of Mathematical Statistics*, **22**, pp.400-407.
- [134] Kiefer, J. and Wolfowitz, J., 1952, "Stochastic Estimation of the Maximum Of A Regression Function," *Annals of Mathematical Statistics*, **23**, pp.462-466.
- [135] Ruppert, D., 1985, "Newton-Raphson Version of the Multivariate Robbins-Monro Procedure," *Annals of Statistics* **13**, pp.236-245.
- [136] Arsham, H., Feuerverger, A., McLeish, D. L., Kreimer, J., and Rubinstein, R. Y., 1989, "Sensitivity Analysis and the "What If" Problem in Simulation Analysis," *Mathematical and Computer Modeling*, **1** (2), pp.193-219.
- [137] Andradottir, S., 1991, "A Stochastic Approximation Algorithm with Bounded Iterates," *Technical Report 91-2*, Department of Industrial Engineering, University of Wisconsin-Madison, Madison, WI.
- [138] Yan, D. and Mukai, H., 1992, "Stochastic Discrete Optimization," *Journal on Control and Optimization*, **30** (3), pp.594-612,

- [139] Gong, W., Ho, W. and Zhai, W., 1992, "Stochastic Comparison Algorithm for Discrete Optimization with Estimation," Proceedings of the 31st Conference on Decision and Control, Tucson, AZ.
- [140] Rossetti, M. D., Clark, G., 1998, Evaluating a Queuing Approximation for the Machine Interference Problem with Two Types of Stoppages via Simulation Optimization," *Computers & Industrial Engineering*, 34 (3), pp. 655-668
- [141] Michael, C. F., and Stacy, D. H., 1997, "Optimization of Discrete Event Systems via Simultaneous Perturbation Stochastic Approximation," *IIE Transactions*, 29 (3), pp. 233-243.
- [142] Kleinman, N. L., Spall, J. C., and Naiman, D. Q., 1999, "Simulation-based Optimization with Stochastic Approximation Using Common Random Numbers," *Management Science*, 45 (11), pp. 1570-1578.
- [143] Subramanian, D., Pekny, J.F., and Reklaitis, G. V., 2000, "A Simulation-Optimization Framework for Addressing Combinatorial and Stochastic Aspects of an R&D Pipeline Management Problem," *Computers and Chemical Engineering*, 24, pp. 1005-1011
- [144] Alberto, I., Azcarate, C., Mallor, F., and Mateo, P.M., 2002, "Optimization with Simulation and Multiobjective Analysis in Industrial Decision-Making: A Case Study," *European Journal of Operational Research*, 140 (2), pp. 373-383.
- [145] Badiru, A. B., and Whitehouse, G. E., 1989, "Computer Tools, Models and Techniques for Project Management," TAB Professional and Reference Books, PA.
- [146] Gido, J., and Clements, J. P., 1999, "Successful Project Management: A Practical Guide for Managers," South-Western College Publishing, Canada.
- [147] PMI Standards Committee, 1996, "A Guide to the Project Management Body of Knowledge", Project Management Institute, PA, USA.
- [148] Rabun, A., and Sommers, J., 1998, "Microsoft Project 98 Support Course," Microsoft Corporation, Redmond, WA.
- [149] Badiru, A. B., 2<sup>nd</sup> ed. 1996, "Project Management in Manufacturing and High Technology Operations," John Wiley & Sons, INC., New York.
- [150] Heuer, G., 1976, "Project Management in Mechanical Engineering: Planning and Controlling of Industrial Intentions," *Vdi-Z*, 118 (7), pp.304-8.
- [151] Brown, A.M., 1984, "Project Management for the Design and Supply of Power Station Mechanical and Electrical Plant," *IEE Proceedings-C Generation Transmission & Distribution*, 131 (6), pp.236-9.
- [152] Feldermann, J., 1993, "Project Management in Production Scheduling Experience of a Large Scale Company in Starting Series Type Production of New Products," *Vdi-Z*, 135 (8), pp.40-42.
- [153] Englund, R. L., and Graham, R. J., 1999, "From Experience: Linking Projects to Strategy," *Journal of Production Innovation Management*, 16, pp. 52-64
- [154] Boznak, R., 1988, "Achieving a Competitive Manufacturing Advantage Through Effective Multi-Project Management," Proceedings, International Industrial Engineering Conference, pp. 285-290.
- [155] Churchill, G.F., 1988, "Quality Assurance - An Effective Project Management Technique," *International Journal of Project Management*, 6 (4), pp. 241-244.

- [156] Beghini, G., and Romanin J. G., 1991, "Project Management and Production Planning: Study for an Integrated Solution," Proceedings, Engineering Systems with Intelligence: Concepts, Tools and Applications, Kluwer Academic Publishers, pp.549-54.
- [157] Maio, A. D., Verganti, R., and Corso, M., 1994, "A Multi-Project Management Framework for New Product Development," European Journal of Operational Research, **78** (2), pp. 178-191.
- [158] Ambrosy, S., Assmann, G., Bindbeutel, K., Cuiper, R., Feldmann, C., and Schmalzl, B., 1996, "Integrated Product and Process Model for Planning and Design," *Zwf Zeitschrift Fuer Wirtschaftlichen Fabrikbetrieb*, **91** (12), pp.607-11.
- [159] Ryba, M., and Baitinger, U. G., 1996, "An Integrated Concept for Design Project Planning and Design Flow Control," Proceedings of the Conference with EURO-VHDL'96 and Exhibition on European Design Automation, pp. 98 – 103.
- [160] Taylor III, B. W., 5<sup>th</sup> ed. 1996, *Introduction to Management Science*, Prentice Hall, New Jersey.
- [161] Batson, R. G., 1987, "Critical Path Acceleration and Simulation in Aircraft Technology Planning," IEEE Transactions on Engineering Management, Vol. EM-34 (4), pp. 244-251.
- [162] Graves, S.B., 1987, "Optimal R and D Expenditure Streams: An Empirical View," IEEE Transactions on Engineering Management, Vol.EM-34 (1), pp.42-48.
- [163] Haffner, E.W., Graves, R.J., 1988, "Managing New Product Time to Market Using Time-Cost Trade-Off Methods," *Omega*, **16** (2), pp.117-124.
- [164] Chapman, C. B., Cooper, D. F. and Page, M. J., 1987, *Management for Engineers*. John Wiley & Sons, New York.
- [165] Kelly Jr, J. E., 1961 "Critical Path Planning and Scheduling: Mathematical Basis," *Operations Research*, **9** (3), pp. 296-320.
- [166] Siemens, N., 1971 "A Simple CPM Time-Cost Trade-Off Algorithm," *Management Science*, **17** (6), pp. 354-363.
- [167] Goyal, S. K., 1975, "A Note On a Simple Cpm Time-Cost Trade-Off Algorithm," *Management Science*, **21** (6), pp. 718-722.
- [168] Berman, E. B., 1964, "Resource Allocation in a PERT Network Under Continuous Activity Time-Cost Functions," *Management Science*, **10** (4), 734-745.
- [169] Falk, J. E., and Horowitz, J. L., 1974, "Critical Path Problems with Concave Cost-Time Curves," *Management Science*, **19** (4), pp. 446-455.
- [170] Fulkerson, D., 1961, "A Network Flow Computation for Project Cost Curves," *Management Science*, **7** (2), pp. 167-178.
- [171] Meyer, W. L. and Shaffer, L. R. , 1963, "Extensions of the Critical Path Method Through the Application of Integer Programming," Department of Civil Engineering, University of Illinois.
- [172] Kuyumcu, A., and Garcia-Diaz, A., 1994, "A Decomposition Approach To Project Compression with Concave Activity Cost Functions," *IIE Transactions*, **26** (6), pp. 63-73.
- [173] Babu, A.J.G., and Suresh, N. , 1996, "Project Management With Time, Cost, and Quality Considerations," *Journal of operational Research*, **88**, pp. 320-327.
- [174] Pulat, P. S., and Horn, S. J., 1996, "Time-Resource Tradeoff Problem," IEEE Transactions on Engineering Management, **43** (4), pp. 411-417.
- [175] Demeulemeester, E., De Reyck, B., Foubert, B., Herroelen, W., and Vanhoucke, M., 1998, "New Computational Results on the Discrete Time/Cost Trade-Off Problem in



- Project Networks," *The Journal of the Operational Research Society*, Oxford, **49** (11), pp 1153-1163.
- [176] AbdelSalam, H. M., and Bao, H. P., 2001, "A Modified Time-Cost Trade-Off Model for Manufacturing Applications," *Virginia Journal of Science*, **52** (2), pp.91 . (Abstract)
- [177] Phillips, D. T., Ravindran, A., and Solberg, J. J., 1976, *Operations Research: Principles and Practice*, John Wiley & Sons, Inc., New York.
- [178] Render, B., and Stair, R. M., 1991, *Introduction to Management Science*, Allyn & Bacon, Massachusetts.
- [179] Williams, H. P., 4<sup>th</sup> ed.1999, *Model Building in Mathematical Programming*, John Wiley & Sons, LTD, New York.
- [180] Tufekci, S., 1982, "A Flow-Preserving Algorithm for the Time-Cost Trade-Off Problem," *IIE Transactions*, **14** (2), pp.109-113.
- [181] Rosenblatt, M.J., and Roll Y., 1985, "A Future Value Approach to Determining Project Duration," *IIE Transactions*, **17** (2), pp.164-167.
- [182] Law, J.S., Hsing-Wei, C., 1987, "Models to Predict Efficiency of Two Network Flow Based Algorithms on the Time-Cost Trade-Off Problem," *Computers & Industrial Engineering*, **12** (2), pp.91-97.
- [183] Deckro, R.F., and Hebert, J.E., 1989, "Resource Constrained Project Crashing," *Omega*, **17** (1), pp.69-79.
- [184] Nair, K.P.K., Prasad, V.R., and Aneja, Y.P., 1993, "Efficient Chains in a Network with Time-Cost Trade-Off Function on Each Arc," *European Journal of Operational Research*, **66** (3), pp.392-402.
- [185] Skutella, M., 1998, "Approximation of Algorithms for the Discrete Time-Cost Trade Off Problem," *Mathematics of Operations Research*, **23** (4), pp. 909-929.
- [186] Gander, J.P., 1985, "Cooperative Research, Government Involvement, and Timing of Innovations," *Technological Forecasting & Social Change*, **28** (2), pp.159-72.
- [187] Levy, J.B., and Tayi, G.K., 1989, "Analysis of Project Scheduling Strategies in a Client-Contractor Environment," *Naval Research Logistics*, **36** (1), pp.69-87.
- [188] Reda, R., and Carr, R. I., 1989, "Time-Cost Trade-Off Among Related Activities," *Journal of Construction Engineering and Management*, **115**, pp. 475-486.
- [189] Icmeli, O., Erenguc, S.S., and Zappe, C.J., 1993, "Project Scheduling Problems: A Survey," *International Journal of Operations & Production Management*, **13** (11), pp.80-91.
- [190] De Reyck, B., and Herroelen, W., 1996, "On the Use Of The Complexity Index As a Measure of Complexity in Activity Networks," *European Journal of Operational Research*, **91** (2), pp.347-366.
- [191] Hajdu, M., 1996, "PDM Time Cost Trade-Off: Activities Are Splittable or Non-Splittable," *Mathematische Operationsforschung und Statistik, Series Optimization*, Vol.38, No.2, pp.155-71.
- [192] De, P., Dune, E.J., Ghosh, J.B., and Wells, C.E., 1997, "Complexity of the Discrete Time-Cost Tradeoff Problem for Project Networks," *Operations Research*, **45** (2), pp.302-306.
- [193] Vanhoucke, M., Demeulemeester, E., and Herroelen, W., 2002, "Discrete Time/Cost Trade-Offs in Project Scheduling with Time-Switch Constraints," *The Journal of Operational Research Society*, **53** (7), pp. 741-751.

- [194] Hamacher, H., and Tufekci, S., 1983, "A lexicographical time-cost tradeoff problem," *Proceedings, Operations Research Verfahren*, No.45, pp.257-68.
- [195] Kanda, A., and Rao, U.R.K., 1984, "A Network Flow Procedure for Project Crashing with Penalty Nodes," *European Journal of Operational Research*, 16 (2), pp.174-82.
- [196] Shouman, M.A., El-nour, A., Ebrahim, S., and El-mahalawy M., 1991, "A Mixed Integer Linear Programming Model for a Time Cost Trade-Off," *Advances in Modeling & Simulation*, 22 (4), pp.53-63.
- [197] Erenguc, S.S., Tufekci, S., and Zappe, C.J., 1993, "Solving Time/Cost Trade-Off Problems with Discounted Cash Flows Using Generalized Benders Decomposition," *Naval Research Logistics*, 40 (1), pp.25-50.
- [198] Liu, L.; Burns, S. A., and Feng, C.-W. , 1995, "Construction Time-Cost Trade-Off Analysis Using LP/IP Hybrid Method," *Journal of Construction Engineering and Management*, 121, pp. 446-454.
- [199] Demeulemeester, E.L., Herroelen, W.S., and Elmaghraby S.E., 1996, "Optimal Procedures for the Discrete Time/Cost Trade-Off Problem in Project Networks," *European Journal of Operational Research*, 88 (1), pp.50-68.
- [200] Chassiakos, A.P., Samaras C.I., and Theodorakopoulos D.D., 1998, "An Algorithm for Determining the Optimal Duration of Large Projects," *Proceedings, Advances in Civil and Structural Engineering Computing for Practice*, pp.449-54.
- [201] Barber, T.J., and Boardman, J.T., 1986, "A pragmatic approach to the optimisation of project control using heuristic techniques," *Proceedings, IEE Colloquium on The Application of Optimisation Techniques to Real Engineering Processes*, pp.5/1-4.
- [202] Barber, T.J., 1989, "Heuristics for project expedition," *Proceedings, IEE Colloquium on Advances in Optimisation*, pp.7/1-6.
- [203] Bowman, R.A., 1994, "Stochastic Gradient-Based Time-Cost Tradeoffs in PERT Networks Using Simulation," *Annals of Operations Research*, 53, pp.533-51.
- [204] Sunde, L., and Lichtenberg, S., 1995, "Net-Present-Value Cost/Time Tradeoff," *International Journal of Project Management*, 13 (1), pp.45-49.
- [205] Taeho, A., and Erenguc, S.S., 1998, "The Resource Constrained Project Scheduling Problem with Multiple Crashable Modes: A Heuristic Procedure" *European Journal of Operational Research*, 107 (2), pp.250-259.
- [206] Ramani, S., 1986, "A Simulation Approach to Time-Cost Trade-Off in Project Network," *Proceedings, Modeling and Simulation on Microcomputers (SCS)*, pp.115-20.
- [207] Patrick, C., and Topuz, E., 1995, "Time-Cost, Trade-Off Analyses Of Longwall Face Transfers," *Mining Engineering*, 47 (4), pp. 281-286.
- [208] Chishaki, T., and Tatish, M., 1992, "New model for project planning by time-cost trade-off procedure using fuzzy durations for project activities," *Memoirs of the Faculty of Engineering Kyushu University*, 52 (4), pp. 339-360.
- [209] Feng, C-W, Liu, L., and Burns, S.A., 1997, "Using Genetic Algorithms to Solve Construction Time-Cost Trade-Off Problems," *Journal of Computing in Civil Engineering*, 11 (3), pp. 184-189.
- [210] Li, H., Cao, J.-N., Love, P. E. D., 1999, "Using Machine Learning and GA to Solve Time-Cost Trade-Off Problems," *Journal of Construction Engineering and Management*, 125 (5), pp. 347-353.

- [211] Feng, C-W., Liu, L., and Burns, S.A., 2000, "Stochastic Construction Time-Cost Trade-Off Analysis," *Journal of Computing in Civil Engineering*, 14 (2), pp.117-126.
- [212] Kusiak, A., Wang, J., He, D. W., and Feng, C-X, 1995, "A Structured Approach for Analysis of Design Processes," *IEEE Transactions on Components, Packing, and Manufacturing Technology – Part1*, 18 (3), pp. 664-673.
- [213] Abramson, D., Krishnamoorthy, M, and Dang, H., 1999, " Simulated Annealing Cooling Schedules for the School Timetabling Problem," *Asia-Pacific Journal of Operational Research*, Singapore.
- [214] Kalsi, M., Hacker, K., and Lewis, K., 2001, "A Comprehensive Robust Design Approach for Decision Trade-Off in Complex System Design," *Journal of Mechanical Design*, 123 (1), pp. 1-10.
- [215] Larson, N., and Kusiak, A., 1996, "Managing Design Processes: A Risk Assessment Approach," *IEEE Transactions on System, Man, Cybernetics*, 26, pp 749-749.
- [216] Hammond, J., Choo, H. J., Austin, S., Tommelein, I. D., and Ballard, G., 2000, "Integrating Design Planning, Scheduling, and Control with DePlan," *Proceedings of the 8<sup>th</sup> Annual Conference of the International Group for Lean Construction (IGLC-8)*, 17-19 July, Brighton, UK.
- [217] Tarjan, R. 1972, "Depth-First Search and Linear Graph Algorithms," *SIAM J. Comput.*, 1 (2), pp. 146-160.
- [218] Bond, A. H., and Ricci, R., 1992, "Cooperation in Aircraft Design," *Res. Eng. Design*, 4 (2), pp. 115-130.
- [210] Kim, J. S., Ritzman, L. P., Benton, W. C., and Synder, D. L., 1992, "Linking Product Planning and Process Design Decisions," *Design Sciences*, 23 (1), pp. 44-60.
- [220] Goldratt, E. M., 1997, *Critical Chain*, North River Pres, Great Barrington, MA.
- [221] Leach, L.P., 2000, *Critical Chain Project Management*, Artech House, London.
- [222] Herring, B. E., and Murphy, L. C., 1987, " Recursive Relation Using The Critical Path to Perform Time-Cost Trade-Offs," *Project Management*, 5 (3), pp. 177-179.
- [223] <http://msdn.microsoft.com/vba/default.asp>
- [224] Kopra, T., Mavris, D., Gomez, P., and Schrage, D., 1994, "Application of Concurrent Engineering Methodology to the Design of a Dual Use VTOL Aircraft," AIAA-94-4329, 5<sup>th</sup> AIAA/NASA/USAF/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Panama City, FL.

## APPENDIX A: optDSM

### 1. Overview

This appendix describes different functions of 'optDSM'; the application tool of the architecture presented throughout the dissertation. The tool was developed by the author using Visual Basic for Application (VBA) programming language. As hinted previously, among its several modules, optDSM has the ability to interface with commercial risk analysis software called Crystal Ball to carry out the optimization process in cases where activity loads assumes stochastic values. The main functions of optDSM are:

1. Modeling of the project under consideration in the form of DSM.
2. Finding the optimum sequence of DSM activities based on a user selected objective function.
3. Structuring the optimized DSM into an equivalent DSM that has no feedback couplings.
4. Converting the structured DSM into a project schedule.

While the first two functions are fully operational, the later two are still under development.

Functional-wise, optDSM is a menu driven Excel add-in. The menu appears automatically, after proper setup, when Excel starts. As shown in Fig. 89, the menu is located on the standard menu tool bar just before the help menu. For operation, the user follows different functions on the menu in order. In the following section, these functions are described.

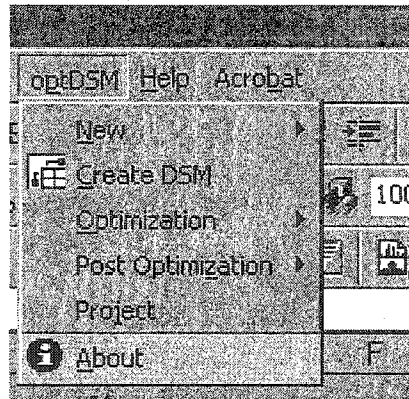


Figure 89. optDSM Main Menu.

## 2. Functions

optDSM has seven functions that have to be carried out sequentially, these are:

### 2.1 New: DSM

The first step of the process is entering the data of the DSM. This is done by choosing 'New' then 'DSM' from the menu as shown in Fig.90 (a). This event brings the 'New DSM options' window shown in Fig. 90 (b), in which the user:

- i. Enters the number of tasks.
- ii. Enters the number of couplings.
- iii. Choose the desired format (Steward's or Rogers's).
- iv. Define the data type (Deterministic or Stochastic).
- v. Enables coupling strength input (if desired).
- vi. Enable logical constraints input – and define their number.

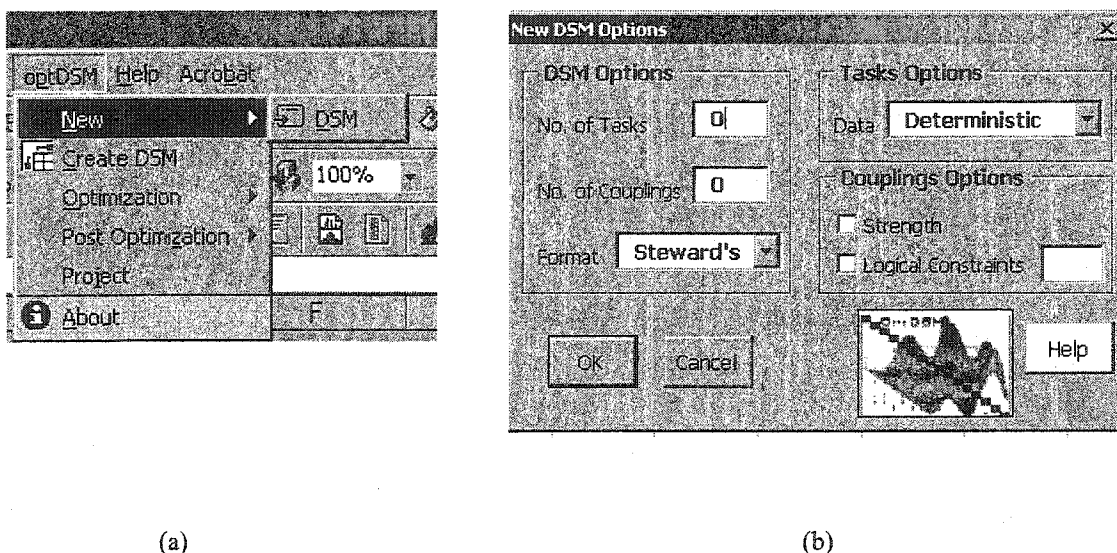


Figure 90. New DSM.

Once the user is finished entering the previous data and clicks on the 'OK' button, optDSM prepares tables for entering activity and couplings details as shown in Fig. 91 (a) and (b) respectively. In case of stochastic activity data, the user is asked to define the data probabilistic distributions using Crystal Ball menu. Figure 92 shows the entry tables after entering the data for a time and cost deterministic optimization case with no logical constraints.

9	22					
10	Task Order	Task Name		Time	Cost	
11	1					
12	2					
13	3					
14	4					
15	5					
16	6					
17	7					
18	8					
19	9					
20	10					
21	11					
22	12					
23	13					
24	14					
25	15					
26	16					
27	17					
28	18					
29	19					
30	20					
31	21					
32	22					
33						
34						
35						

(a) Activities Table

9							
10							
11							
12							
13							
14							
15							
16							
17							
18							
19							
20							
21							
22							
23							
24							
25							
26							
27							
28							
29							
30							
31							
32							
33							
34							

(b) Couplings Table

Figure 91. Data Entry Tables.

9	22					
10	Task Order	Task Name	Task ID	Lead	Time	Cost
11	1	Initial Data	INITDAT		40	20
12	2	Dynamic Model	DYNMODL		30	30
13	3	Stability & Model Charact	STDMOCH		40	20
14	4	Structure Model	STRMODL		10	50
15	5	Stability Qualities	HANDQUL		10	50
16	6	Structure Mode	STRMODE		10	50
17	7	Geometry Development	GEOMDEV		50	10
18	8	Aero Elastic Properties	AEROSRVO		40	20
19	9	Aero Elastic Analysis	STRDYNA		50	10
20	10	Control System Analysis	CSYSANAL		20	40
21	11	Flex. Aero Characteristic	FAREROCH		20	40
22	12	Revise Initial Data	RVSEDAT		30	30
23	13	Mission Performance	MISPERF		30	30
24	14	Vehicle Performance	VEHEPERF		20	40
25	15	Rigid Aero Characteristic	RAEROCH		30	30
26	16	Aero Analysis	AEROANL		20	40
27	17	Pressure & Deflection	PRESDEF		30	30
28	18	Structure Analysis	STRANAL		40	20
29	19	Structure Weight	STRCTWT		50	10
30	20	Weight & Inertia Analysis	WIANAL		40	20
31	21	Aero Model	AEROMDL		20	40
32	22	Final Data	FINLDAT		20	40
33						

(a) Activities Table

9							
10	No.	From	To	Strength	Iteration Factor	Initial Coupling Type	
11	1	INITDAT	GEOMDEV	es			
12	2	RVSEDAT	INITDAT	n			
13	3	MISPERF	RVSEDAT	vw			
14	4	GEOMDEV	AEROMDL	es			
15	5	GEOMDEV	STRMODL	ew			
16	6	AEROMDL	AEROANL	es			
17	7	PRESDEF	AEROANL	vs			
18	8	AEROANL	PRESDEF	s			
19	9	STRANAL	PRESDEF	es			
20	10	PRESDEF	STRANAL	vs			
21	11	STRCTWT	STRANAL	s			
22	12	STRMODL	STRANAL	w			
23	13	STRANAL	STRCTWT	es			
24	14	WIANAL	STRCTWT	ew			
25	15	GEOMDEV	WIANAL	w			
26	16	STRCTWT	WIANAL	es			
27	17	AEROANL	RAEROCH	s			
28	18	PRESDEF	FAREROCH	es			
29	19	STRANAL	STRMODE	s			
30	20	WIANAL	VEHEPERF	w			
31	21	RAEROCH	VEHEPERF	es			
32	22	FAREROCH	VEHEPERF	ew			
33	23	AOSRVO	VEHEPERF	vs			
34	24	HANDQUL	VEHEPERF	vw			
35	25	VEHEPERF	MISPERF	s			
36	26	STRDYNA	STDMOCH	es			
37	27	GEOMDEV	STRDYNA	s			
38	28	RAEROCH	STRDYNA	w			

(b) Couplings Table

Figure 92. Data Entry Tables Filled.

## 2.2 Create DSM

Once user enters data, the 'Create DMS' function is chosen so that optDSM:

- i. Determines the iteration factor of each coupling according to its coupling strength.
- ii. Determines the coupling initial nature (feedback or feed forward).
- iii. Develops the initial DSM according to the sequence entered.

## 2.3 Optimization: Settings

The next step is the development of the excel sheets associated with optimization model calculations. When the user instantiates the 'Optimization: Settings' function, as shown in Fig. 93, a main menu titled 'Optimization Options' appears.

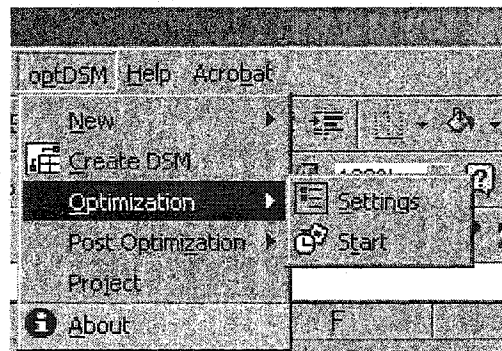


Figure 93. Optimization Sub-menu.

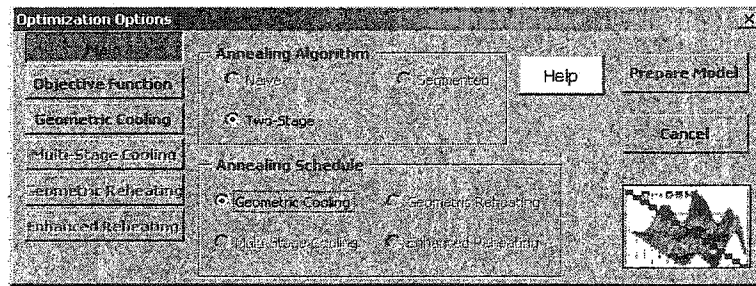


The 'optimization options' window is divided into six layers, three of which are currently active, as shown in Fig. 94, and the other three will be active in future work.

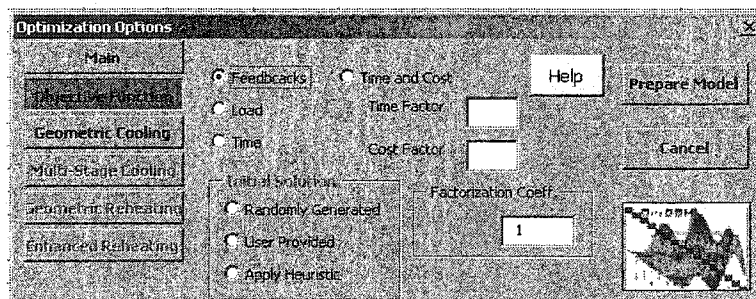
The three layers are:

1. Main, shown in Fig. 94 (a). Allows the user to choose:
  - a. The SA algorithm used (currently only one is available).
  - b. The annealing schedule used (also only one is available in the presented version).
2. Objective Function, shown in Fig. 94 (b). Enables the user to:
  - a. Choose the optimization objective function.
  - b. Enter the weights to be used in case of (time and Cost) minimization.
  - c. Define the objective function factorization coefficient value.
  - d. Choose a strategy for the initial solution generation (currently, this option is disabled and initial solution is generated randomly).
3. Geometric Cooling, shown in Fig. 94 (c). Since the geometric cooling is the default annealing schedule, this window allows the user to choose between using the default values of the annealing schedule parameters provided by optDSM or entering different values for these parameters.

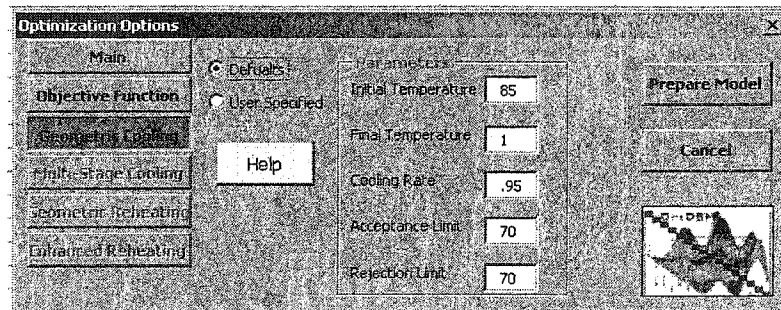
Once the user clicks 'Prepare Model' button, optDSM prepares calculations sheets. Figure 8.12 shows the feedback calculations sheet.



(a) Main window



(b) Objective Function Options



(c) Geometric Cooling Options

Figure 94. Optimization Options/Settings window.

Solution Configuration																											
ID	UNITDAT	BYUNIT	STRANCA	STRANAL	RANRANL	STRANCAE	CEANRCE	ANRANR	STRANR	CYANR	ANRANR	BYRANR	BYRANR	BYRANR	BYRANR	BYRANR	BYRANR	BYRANR	BYRANR	BYRANR	BYRANR						
Order	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22					
Feedbacks Table										Logic Constraints																	
Total										Total																	
From	To	From Order	To Order	Diff.	FM	MM	MM	MM	MM	No	LHS	RHS	Diff	MM													
UNITDAT	CEANRCE	1	7	6	0	7	1																				
BYUNIT	UNITDAT	12	1	-11	1	12	1																				
BYRANR	BYRANR	13	12	-1	1	13	12																				
CEANRCE	RANRANL	7	21	14	0	21	7																				
CEANRCE	STRANCAE	7	4	-3	1	7	4																				
RANRANL	RANRANL	21	16	-5	1	21	16																				
BYRANR	RANRANL	17	16	-1	1	17	16																				
RANRANL	BYRANR	16	17	1	0	17	16																				
STRANAL	BYRANR	18	17	-1	1	18	17																				
BYRANR	STRANAL	17	18	1	0	18	17																				
STRANCAE	STRANAL	19	18	-1	1	19	18																				

Figure 95. Feedback Calculations Sheet.

### 2.4 Optimization: Start

This function initiates the optimization process. optDSM maintains two sets of results:

1. The first for all evaluated solutions.
2. The second for the meta-stable optimal solutions.

Screen shots of optDSM while the optimization process is running for a deterministic case and for a stochastic case are shown in Fig. 96 and Fig. 97 respectively.

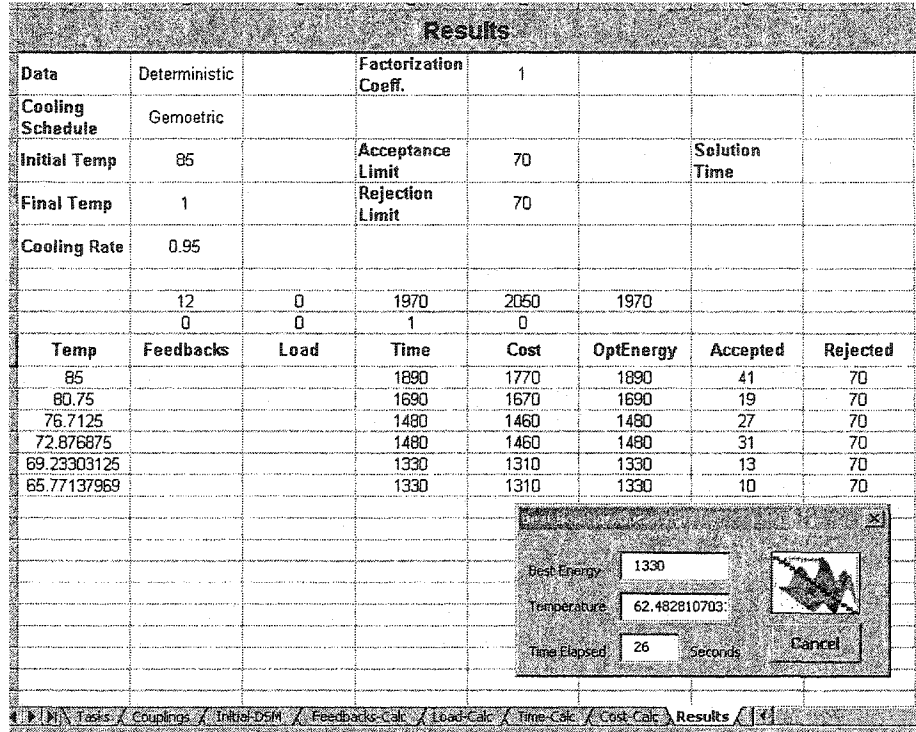


Figure 96. Screen Shot: Deterministic Optimization.

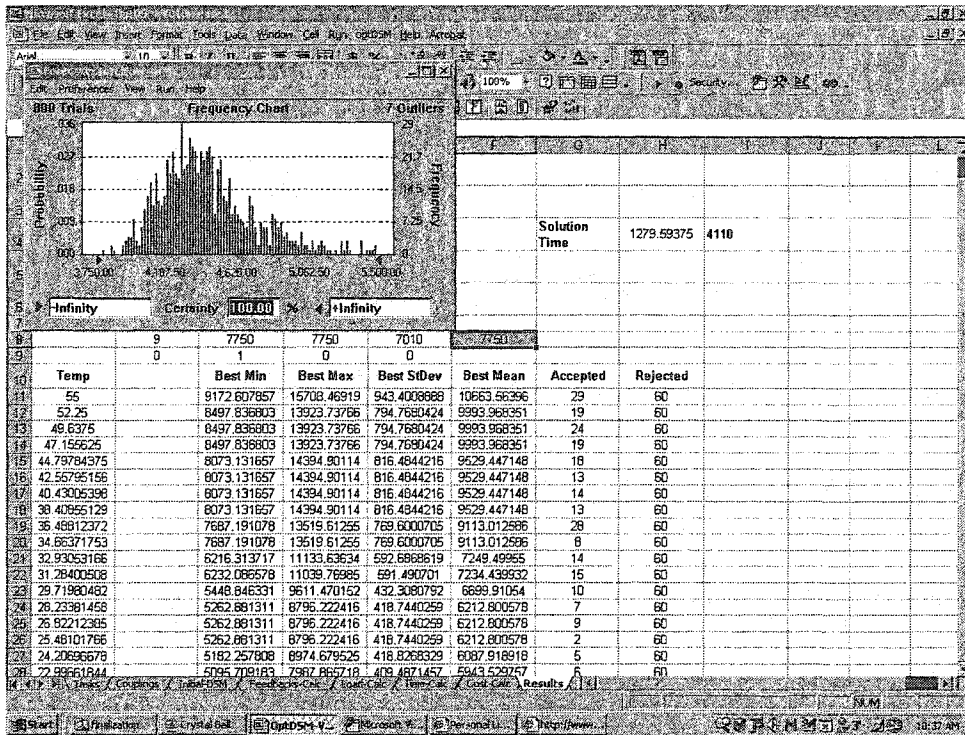


Figure 97. Screen Shot: Stochastic Optimization.

## 2.5 Optimization: Final DSM

After the optimization process ends, this functions, shown in Fig. 98, creates the final DSM; the one corresponding to optimal activity sequence.

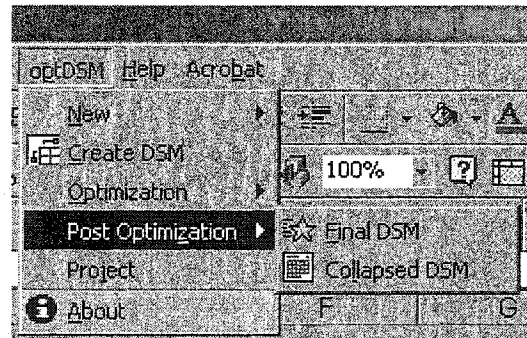


Figure 98. Post Optimization Sub-menu.

The next two functions are still under development, these are:

## 2.6 Post Optimization: Collapsed DSM

This function converts the optimally sequenced DSM into an equivalent DSM with no feedback couplings.

## 2.7 Project

Finally, an equivalent project schedule is developed and then transferred to MS Project for scheduling purposes.

### 3. Future Modifications

In addition to finishing the un-working functions, the followings are some improvements that could be added to optDSM in following versions:

1. A function to check the correctness of input data (activities, couplings, etc.). The objective here is to assure that everything was entered in order.
2. Facilitation the optimization process by introducing:
  - a. A function that determines activities with no input nor output (i.e. isolated) and moves these activities to the top of the DSM.
  - b. A function that determines activities with no inputs and moves these activities to the top of the DSM.
  - c. A function that determines activities with no outputs and moves these activities to the end of the DSM.
  - d. Reducing the optimization models after these functions.

## CURRICULUM VITA for

**Hisham Abdelsalam**

### DEGREES:

- Doctor of Philosophy (Mechanical Engineering), Old Dominion University, Norfolk, Virginia August 2003
- Master of Science (Mechanical Engineering), Old Dominion University, Norfolk, Virginia May 2000
- Bachelor of Science (Mechanical Engineering), Cairo University, Cairo, Egypt July 1996

### PROFESSIONAL CHRONOLOGY:

- **Graduate Teaching/Research Assistant**, August 1999 – Present  
Department of Mechanical Engineering, Old Dominion University, Norfolk, Virginia
- **Graduate Teaching/Research Assistant**, November 1997 - July 1999  
Operations Research & Decision Support Department, Cairo University, Cairo, Egypt
- **Environmental Project Engineer**, July 1997 – October 1997  
Development Research & Technological Planning Center, Cairo University, Cairo, Egypt.

### HONORS AND AWARDS:

- Mechanical Engineering 2001 Outstanding Graduate Student, Old Dominion University.
- Member, Phi Kappa Phi Honor Society, 2001.
- Graduate Assistantship, Old Dominion University, Fall 1999- August 2003.
- Honors Degree, B.Sc., Cairo University, 1996.
- Academic Achievement Scholarship, B.Sc., Cairo University, 1991 and 1993-96.

### SELECTED PUBLICATIONS:

- Abdelsalam, H. M., and Bao, H. P., "The Use of Design Structure Matrix and Simulated Annealing to Reduce Product Development Cycle Time," accepted for publication in the proceedings of the 13<sup>th</sup> International Flexible Automation and Intelligent Manufacturing (FAIM) conference, June 9-11, 2003.
- Abdelsalam, H. M., and Bao, H. P., 2000, "Towards a Collaborative Engineering-Computation Environment: An Application Of An Object-Oriented Database To Project Management," Proceedings of DETC'00, ASME 2000 Design Engineering Technical Conferences and Computers and Information in Engineering Conference, September 10-13, 2000, Baltimore, Maryland.
- Abdelsalam, H. M., and Bao, H. P., 2000, "Solving the Project Time-Cost Trade-Off Problem Through an Integrated Engineering-Computation Environment," Proceedings, The 11<sup>th</sup> Annual Conference of the Production and Operations Management Society, POM-2000, April 1-4, 2000, San Antonio, TX.