

COORDINATED RESCHEDULING OF PRECAST PRODUCTION

ZENG ZHEN

NATIONAL UNIVERSITY OF SINGAPORE

2006

**COORDINATED RESCHEDULING OF
PRECAST PRODUCTION**

ZENG ZHEN

(B.Eng., M. Eng.)

**A THESIS SUBMITTED
FOR THE DEGREE OF DOCTOR OF PHILOSOPHY**

**DEPARTMENT OF CIVIL ENGINEERING
NATIONAL UNIVERSITY OF SINGAPORE**

2006

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude, first and foremost, to my supervisor, Associate Professor Chan Weng Tat, for his patient, generous and constructive guidance, continuous inspirations and encouragements in the course of this study.

I am fortunate to be a research student in the Department of Civil Engineering, National University of Singapore (NUS). Thanks to the remarkable people and outstanding academic environment of NUS and Singapore, my experience as a research student at NUS has been pleasurable and fruitful.

I am also grateful to Mr. Ong Ting Guan, Project Manager; and Ms. Loh Li Hwa, Construction Engineer of the Construction Technology Pte Ltd; and Ms. Tan Meow Cheng Debbie, General Manager of the Eastern Pretech Pte Ltd for several interviews, plant visits and invaluable suggestions in the course of my study.

Finally, I would like to express my heartiest gratitude to my family in China for their sacrifice, understanding and support over these years; and my husband Pan Heng for these years we spent together in Singapore.

TABLE OF CONTENTS

Acknowledgements -----	I
Table of Contents -----	II
Summary -----	V
List of Tables -----	VII
List of Figures -----	IX
CHAPTER 1 INTRODUCTION -----	1
1.1 The Precast Industry in Singapore-----	2
1.2 Schedule Coordination for Precast Production-----	3
1.2.1 Precast Supply Chain-----	3
1.2.2 Schedule Coordination Practices-----	5
1.3 Rescheduling Practices in Precast Factories-----	7
1.3.1 Production Planning and Control Processes-----	7
1.3.2 Occurrence of Schedule Disturbances-----	9
1.3.3 Features of Rescheduling Practices-----	11
1.4 Current Research in Precast Planning and Scheduling-----	12
1.5 Research Objectives and Scope-----	14
1.6 Research Methodology-----	15
1.7 Thesis Organization-----	17
CHAPTER 2 LITERATURE REVIEW -----	19
2.1 Planning and Scheduling for Precast Production-----	19
2.2 Reactive Scheduling-----	25
2.2.1 Overview-----	25
2.2.2 Approaches in Reactive Scheduling-----	27
2.3 Multiobjective Optimization Problems-----	30
2.3.1 Basic Concepts and Terminologies-----	30
2.3.2 Multiobjective Optimization Methods-----	32
2.4 Genetic Algorithms and Applications to Scheduling-----	34
2.4.1 Overview of GAs-----	34
2.4.2 Multiobjective Genetic Algorithms-----	35
2.4.3 Applications to Scheduling Problems-----	36
2.5 Summary-----	38
CHAPTER 3 PRECAST PRODUCTION RESCHEDULING -----	41
3.1 Precast Production Rescheduling Problem-----	41
3.1.1 Overview of Precast Production Process-----	41
3.1.2 Production Constraints-----	44
3.1.3 Optimizing Objectives-----	45

3.1.4 Mathematical Formulation-----	46
3.2 Coordinated Production Reactive Scheduling Model-----	50
3.2.1 Model Overview-----	50
3.2.2 Disturbance Detection-----	51
3.2.3 Global Search with Multiobjective Optimization-----	54
3.2.4 Local Search with Specific Constraints-----	61
3.2.5 Ranking of Outcomes for Negotiation-----	63
CHAPTER 4 MULTIOBJECTIVE GENETIC ALGORITHMS FOR GLOBAL SEARCH-----	65
4.1 Basic Mechanisms of Genetic Algorithms-----	65
4.2 Genetic Algorithms for Multiobjective Optimization-----	69
4.2.1 Key Issues in Multiobjective Search-----	69
4.2.2 Weighted Sum Genetic Algorithm-----	71
4.2.3 Non-dominated Sorting Genetic Algorithm (NSGA)-----	72
4.2.4 NSGA with Proposed Elitist Strategies-----	77
4.3 Implementation of GAs in Global Search-----	82
4.3.1 Chromosome Representation-----	83
4.3.2 Decoding-----	85
4.3.3 Objective Functions-----	85
4.3.4 Relation among chromosomes, schedules and objective functions-----	86
4.3.5 Genetic Operators-----	87
4.3.6 Software Used for the Study-----	87
CHAPTER 5 GLOBAL SEARCH FOR REPAIRED SCHEDULES – CASE STUDY-----	88
5.1 Illustrative Test Cases-----	88
5.2 Performance Measurement-----	91
5.3 GA Parameters-----	93
5.4 Results and Discussion-----	94
5.4.1 The L-U example-----	95
5.4.2 The M-U example-----	99
5.4.3 The H-U example-----	102
CHAPTER 6 LOCAL SEARCH WITH SPECIFIC CONSTRAINTS-----	109
6.1 Overview of Local Search-----	109
6.2 Implementation of Local Search with Specific Constraints-----	112
6.2.1 Specific Constraints-----	112
6.2.2 Objective Function-----	114
6.2.3 Initial Solutions-----	115
6.2.4 Neighborhood Structure-----	116
6.2.5 Search Heuristics-----	117
6.3 Case Study-----	121
6.3.1 Illustrative Examples-----	121
6.3.2 Results and Discussion-----	123

CHAPTER 7 CONCLUSIONS AND RECOMMENDATIONS	133
7.1 Conclusions.....	133
7.1.1 Development of a Coordinated Production Reactive Scheduling Model --	134
7.1.2 Generation of Repaired Schedules along a <i>Pareto</i> Front	135
7.1.3 Exploration of Schedules with Specific Constraints.....	136
7.2 Limitations of the Research	137
7.3 Recommendations for Future Research.....	138
REFERENCES	141
APPENDIX I Data and Schedules for the L-U Example	153
APPENDIX II Data and Schedules for the M-U Example	155
APPENDIX III Data and Schedules for the H-U Example	157

SUMMARY

Schedule disturbances are common and inevitable in the process of precast production. Not only is it necessary for the precaster to repair existing production schedule to accommodate these unexpected changes, it is also critical that the precaster and the contractor reach agreement on a new delivery schedule. However, the current practice of rescheduling is rudimentary in terms of computer support and depends largely on human experience. Without a proper exploration of the possibilities to resolve the schedule disturbances, both parties are likely to adopt overly conservative assumptions to optimize their own interests. A more beneficial approach would be to incorporate specific requirements from both parties and support negotiation through computer-aided approaches to the generation of a range of alternatives meeting these requirements.

This research has proposed and developed a coordinated production reactive scheduling model for this purpose. The fundamental basis of the model involves the formulation of the precast production rescheduling problem as a multiobjective optimization problem, in a way that includes the objectives from both the precaster and the contractor. A multiobjective genetic algorithm is applied in the global search procedure for a rich set of alternative repaired schedules. This search exploits the use of a solution representation that gives the best sequence and the corresponding heuristics needed to resolve the disturbances. The results from several examples in a case study have demonstrated the utility of the procedure developed, principally in automating the generation of alternative schedules that involve different degrees of trade-off between the objectives. Unlike the commonly adopted approach to solve multiobjective optimization problems, this has been achieved without the need to

pre-determine weights for the objectives. Comparisons between several GA-centric optimization techniques show that a variation of non-dominated sorting genetic algorithm with the elitist strategy proposed in this research is more consistent in locating non-dominated solutions along the *Pareto* front regardless of different mold utilization levels in production schedules.

As a further enhancement to the proposed model, a local search process is implemented to conduct incremental exploration of the search space in specific areas identified by either the precaster or the contractor. The basic idea is to improve existing repaired schedules iteratively by searching for alternatives with specific characteristics in the neighborhoods of solutions on the *Pareto* front. This capability would be useful when minimal adjustments are needed for the alternatives generated by the global search in the first phase. The encouraging results obtained from the case study suggest that the proposed Min-Max Conflicts heuristic is capable of finding specific schedules by exploiting domain knowledge associated with specific constraints; furthermore, the local search can be completed within a reasonable amount of computational time. Together, the alternative schedules generated by the global search procedure as well as the specific schedules from the local search procedure provide the precaster and the contractor useful insight into the trade-offs between their objectives as they negotiate a new delivery schedule.

Keywords: *rescheduling, schedule coordination, multiobjective optimization, genetic algorithms, local search, precast production.*

LIST OF TABLES

Table 3.1 Multiple optimizing objectives for precast production rescheduling -----	45
Table 3.2 Production schedule representation -----	47
Table 3.3 Parameters considered for the rescheduling problem-----	47
Table 3.4 Characteristics representation of schedule disturbances -----	52
Table 5.1 Problem parameters -----	89
Table 5.2 Characteristics of schedule disturbances -----	90
Table 5.3 Heuristics representation -----	91
Table 5.4 Frequency of convergence with different GA parameters -----	94
Table 5.5 GA parameters used in case study -----	94
Table 5.6 Performances of NSGA, NSGA-ESI and NSGA-ESII for the L-U Example-----	98
Table 5.7 Performances of NSGA, NSGA-ESI and NSGA-ESII for the M-U Example -----	103
Table 5.8 Performances of NSGA, NSGA-ESI and NSGA-ESII for the H-U Example -----	108
Table 6.1 Different circumstances for repaired schedules -----	113
Table 6.2 Information for available repaired schedules -----	122
Table 6.3 Specific constraints considered in two cases -----	122
Table 6.4 Performance of search heuristics for Case 1 -----	124
Table 6.5 Performance of search heuristics for Case 2 -----	129
Table A.1 Site demands for the L-U example -----	154
Table A.2 Original production schedule of the L-U example -----	154
Table A.3 One repaired production schedule of the L-U example-----	154
Table B.1 Site demands for the M-U example -----	156
Table B.2 Production schedule of the M-U example -----	156

Table B.3 One repaired production schedule of the M-U example -----	156
Table C.1 Site demands for the H-U example -----	158
Table C.2 Production schedule of the H-U example -----	158
Table C.3 One repaired production schedule of the H-U example -----	158

LIST OF FIGURES

Figure 1.1 Precast supply chain -----	5
Figure 1.2 Planning and control processes for precast production -----	9
Figure 1.3 “External” and “In-house” causes for schedule disturbances -----	10
Figure 2.1 Illustration of domination in multiobjective optimization -----	31
Figure 3.1 Relationship between elements and molds -----	42
Figure 3.2 Layout of production line in a precast factory -----	43
Figure 3.3 Framework of the coordinated production reactive scheduling model-----	51
Figure 3.4 Illustration of schedule array and disturbances-----	52
Figure 3.5 Illustration of heuristic strategies for schedule repair -----	57
Figure 3.7 Iterative repair process with specific constraints-----	64
Figure 4.1 GA chromosome -----	66
Figure 4.2 Illustrations of crossover and mutation-----	68
Figure 4.3 Flowchart of the non-dominated sorting genetic algorithm (NSGA) -----	73
Figure 4.4 Flowchart of the NSGA with proposed elitist strategy -----	82
Figure 4.5 Illustration of random representation-----	84
Figure 4.6 Chromosome representations and decoding-----	85
Figure 4.7 Relationship for chromosomes, schedules and objective functions -----	86
Figure 5.1 $PF_{current}$ found with the weighted sum GA for the L-U example-----	96
Figure 5.2 $PF_{current}$ found with NSGA for the L-U example-----	97
Figure 5.3 $PF_{current}$ found with the weighted sum GA for the M-U example -----	99
Figure 5.4 $PF_{current}$ found with NSGA for the M-U example-----	100
Figure 5.5 $PF_{current}$ found with NSGA-ESI for the M-U example -----	101
Figure 5.6 $PF_{current}$ found with NSGA-ESII for the M-U example -----	101

Figure 5.7 Frequency of convergence and solutions found for the M-U example---	102
Figure 5.8 $PF_{current}$ found with the weighted sum GA for the H-U example -----	104
Figure 5.9 $PF_{current}$ found with NSGA for the H-U example -----	105
Figure 5.10 $PF_{current}$ found with NSGA-ESI for the H-U example -----	105
Figure 5.11 $PF_{current}$ found with NSGA-ESII for the H-U example -----	106
Figure 5.12 Frequency of convergence and solutions found for the L-U example --	107
Figure 6.1 Algorithmic skeleton of iterative improvement -----	112
Figure 6.2 The swap neighborhood of the production schedule -----	117
Figure 6.3 Search procedure of the Max-Min Conflicts heuristic -----	121
Figure 6.4 Results of the random search heuristic for Case 1 -----	126
Figure 6.5 Results of the Max-Min Conflicts heuristic for Case 1 -----	127
Figure 6.6 Results of the random search heuristic for Case 2 -----	131
Figure 6.7 Results of the Max-Min Conflicts heuristic for Case 2 -----	132

CHAPTER 1

INTRODUCTION

The public housing program in Singapore has featured the use of a significant proportion of precast building components in order to meet cost, schedule and quality goals. Although the production of precast components takes place in a controlled factory environment, a high frequency of schedule disturbances is still inevitable in the production process, especially those that originate from the construction site. This makes frequent rescheduling necessary in order to repair the current schedule to accommodate changes and keep the production process going smoothly. However, current rescheduling practices in precast factories are fairly basic in nature and very much based on experience. Coordination in handling schedule disturbances and rescheduling between the precast factory and the construction site is also a challenge for the parties involved in the precast supply chain since they are independent entities. Currently, there has been little research work in the area of precast planning and scheduling that addresses these problems. Therefore, the focus of current research is on (1) modeling the precast production rescheduling problem, and (2) adopting appropriate methods to solve the problem effectively and facilitate coordination of schedules between the parties involved.

The subsequent sections provide some background information including the development of the local precast industry and the stages in the construction supply chain for precast construction, and highlight the importance of schedule coordination in the use of precast building components in construction. Production planning and control processes employed in precast factories, the nature of schedule disturbances encountered and common features in the rescheduling practice are detailed.

1.1 The Precast Industry in Singapore

Precast building components range from smaller items like concrete planks, beams and slabs, to larger and more voluminous items like bathrooms, staircases and water storage tanks. Unlike cast in-situ building members, precast components are made with molds at a centralized facility. They are then transported to the construction site for erection and assembly. Over the last few decades, precast fabrication has gained wide recognition and application through developments in design, material, manufacturing and erection technologies, and helped the construction industry achieve great savings in construction cost. Precast fabrication is of special interest in Singapore, where there is both a significant national public housing building program and a constant drive to improve on the design, quality and buildability of public housing units (Chan *et al.* 1999).

The idea of using precast fabrication in Singapore was first mooted back in 1963, because of the need to complete a contract for ten blocks of standard ten-storey 3-room flats in a short time (Lee 1989). Precast fabrication for buildings took off only in the early 80s when the Housing and Development Board (HDB) introduced large-scale industrialization in its public housing program (CIDB Construction Productivity Taskforce, 1992). It was logical for the HDB to use prefabrication in its building programme as most of the requirements and structural dimensions for many of its housing unit designs had been standardized. Since then, precast fabrication has been used extensively in both construction and upgrading of public housing by the HDB.

The application of precast fabrication has also been identified as one form of buildable design. Buildable design is about ease of construction and characterized by

simplicity, standardization, and combination of related components together into a single element that may be prefabricated in the factory and installed on site. It was introduced by the Construction Industry Development Board (CIDB) in 1991, and is continued being promoted extensively by the Building and Construction Authority (BCA). It is developed in the face of increasing construction costs, as well as the necessity of solving the problem of low productivity and heavy dependence on unskilled foreign workers in the local construction industry. With the use of modular coordination and standardization of elements in the early years, buildable design is more recently achieved through the greater use of prefabrication especially in public housing and transport infrastructure projects (Chionh 1999). The promotion of buildable designs has also led to the local precast sector developing strong capabilities in design and fabrication, especially in the areas of detailed connections and aesthetics. This development in turn has made precast fabrication a viable option for private property projects in recent years (Tan 2003). The demand for precast fabrication in Singapore, both in public and private sector works, is good as the construction industry increasingly adopts the concept of buildable designs.

1.2 Schedule Coordination for Precast Production

The organization of the construction team changes somewhat because of a decision to adopt precast construction technology in a project. Decision coordination between the parties involved in the precast supply chain becomes an important practical issue.

1.2.1 Precast Supply Chain

The precast supply chain is short compared to that in other industrial manufacturing

sectors. The main parties involved in the precast supply chain include the contractor, the precaster, and the delivery supplier. Figure 1.1 shows key activities of these parties, as well as the information and material flow within the precast supply chain. When precast fabrication is adopted, the process of construction work is modified to include the following activities:

- (1) The production of precast components;
- (2) The transportation of precast components;
- (3) The erection and assembly of precast components; and
- (4) The connection of erected components.

All of the parties involved have to work closely on the design, fabrication, transport, and erection of precast building components.

Within the activities of the precast supply chain, the precaster and the contractor are the two primary parties involved. The precaster is responsible for producing precast components with off-site automation and will also arrange transportation of these components to the construction site. The contractor is responsible for subsequent operations within the site, such as erection, assembly, and connection of these components. Generally, the pace of precast production should keep up with the progress of construction on-site. Otherwise, delays in the production of some components may have repercussions on the production of other components and ultimately compromise the progress of construction. Therefore, cooperation and collaboration between the precaster and the contractor is important in the area of efficient logistics and production scheduling. Although the construction schedule dictates the precast production schedule, there remains a lot of scope for better coordination between these two schedules in order to promote a win-win situation for the contractor-precaster relationship.

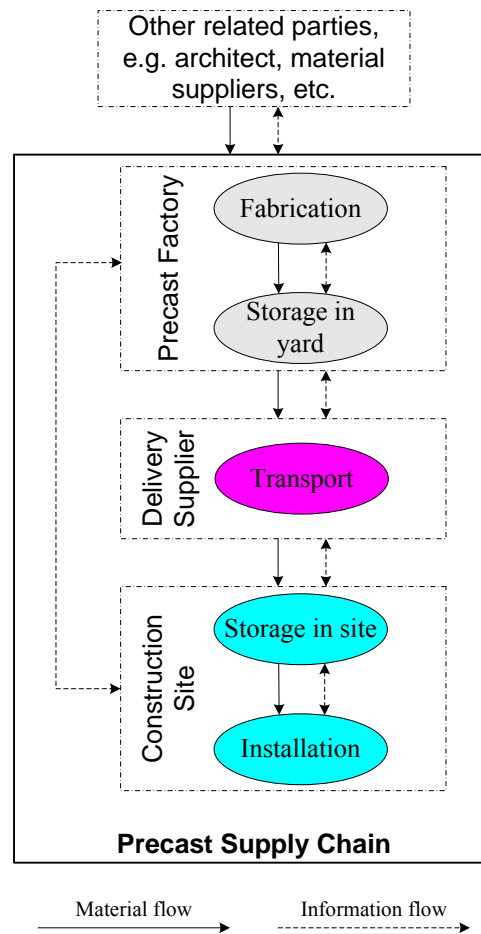


Figure 1.1 Precast supply chain

1.2.2 Schedule Coordination Practices

Schedule coordination involves consultation and negotiation between the precaster and the contractor throughout the precast production process. The most tangible outcome of this coordination is the delivery schedule, which consists of specific delivery quantities and due dates for the precast components. Events ranging from a change in design specifications, quantities, due dates for erection to resource availability are typically encountered in the course of the project. Both the precaster and the contractor customarily add lead times and inventory buffers as insurance against uncertain events. Should the impact of such events exceed the ability of these mechanisms to cope, a review of the production and construction schedules by the

respective parties, and possibly, even a requirement of rescheduling the production becomes necessary. New quantities and due dates for the delivery of precast components have to be negotiated between the precaster and the contractor.

However, production in precast factories is typically set up in a central location and organized in a manner that is similar to a manufacturing production assembly, which is different from the traditional manner in which production in the construction industry is carried out. Consequently, the way in which the production schedule is generated by the precaster is also different from that employed by the contractor for the construction schedule. The precast production scheduling involves resource allocation over time for the manufacture of precast elements, whilst the construction project scheduling allocates resources at specified times to tasks which together complete a project. These differences present difficulties for the precaster and the contractor in the coordination of their respective schedules.

Moreover, the precast factory typically operates as an independent business entity or a highly autonomous unit within a large parent company related to the contractor. Coordination between them may become harder when it has to be carried out across organizational lines. These two parties have their own concerns for the project, as well as goals that may conflict between one and another, leading to different emphasis during schedule coordination. For example, the contractors often express unhappiness with occurrences of late deliveries by the precasters, whilst the precasters blame the late deliveries on inaccurate demand schedules, slow revision of designs and communication of updates and changes, as well as last-minute requests by the contractors. Such diametrically opposed viewpoints for the cause of delivery delays make it difficult for both parties to negotiate and arrive at mutually beneficial

schedules. Furthermore, each side may adopt overly conservative assumptions in the negotiation process for a new delivery schedule in order to protect their own interests.

Rather than protecting individual self-interest to the fullest extent, it might be more beneficial for both parties to share scheduling information, mutually explore various possibilities, resolve conflicting demands and come up with the best possible compromise in the delivery schedule.

1.3 Rescheduling Practices in Precast Factories

Before considering the question of how to improve schedule coordination for precast production, it is necessary to look into the production planning, scheduling and rescheduling practices employed in precast factories.

1.3.1 Production Planning and Control Processes

In general, the precast factory tries to ensure both the timely delivery of required elements to the construction site and the most efficient utilization of factory resources. The planning and control processes in the precast factory include the following key activities:

(1) Planning and Scheduling: These two activities are closely linked to each other, and are carried out for each project. Planning deals with resource availability, plant and mold layout, as well as crew organization. It is often considered as a “higher” level problem because it acts the parameters that influence how scheduling is carried out, in particular the numbers of different molds available to the project. Thus, scheduling decides on when and what elements are to be produced on the available molds (Hu

2000).

(2) Production Rescheduling: This activity recurs throughout the duration of precast production for a project. Rescheduling is necessary during actual work progress because of variations and uncertainties that might affect the existing production schedule. The production scheduler keeps track of the actual production process, detects problems caused by unexpected events, initiates the rescheduling process, and negotiates with the contractor for new delivery quantities and due dates.

(3) Information Transfer: It is necessary to exchange production and schedule information between the parties involved in the precast supply chain. On the one hand, information transfer within the factory occurs between different departments, such as the casting department, the molds fitter and the reinforcement shop. On the other hand, the precaster will also be in communication with the contractor and the delivery supplier. Efficient information exchange facilitates decision coordination among different parties involved in precast production.

The literature on production scheduling research refers to two types of scheduling – predictive scheduling (off-line) and reactive scheduling (on-line), as shown in Figure 1.2. Predictive scheduling predicts/forecasts the planned start and completion times of job operations based on deterministic inputs for processing times, sequencing order, availability of resources, and demand. Reactive scheduling is viewed as continuous adaptation and improvement of the pre-computed predictive schedule in order to keep it in line with unfolding and often unanticipated events. (Sabuncuoglu and Bayiz 2000).

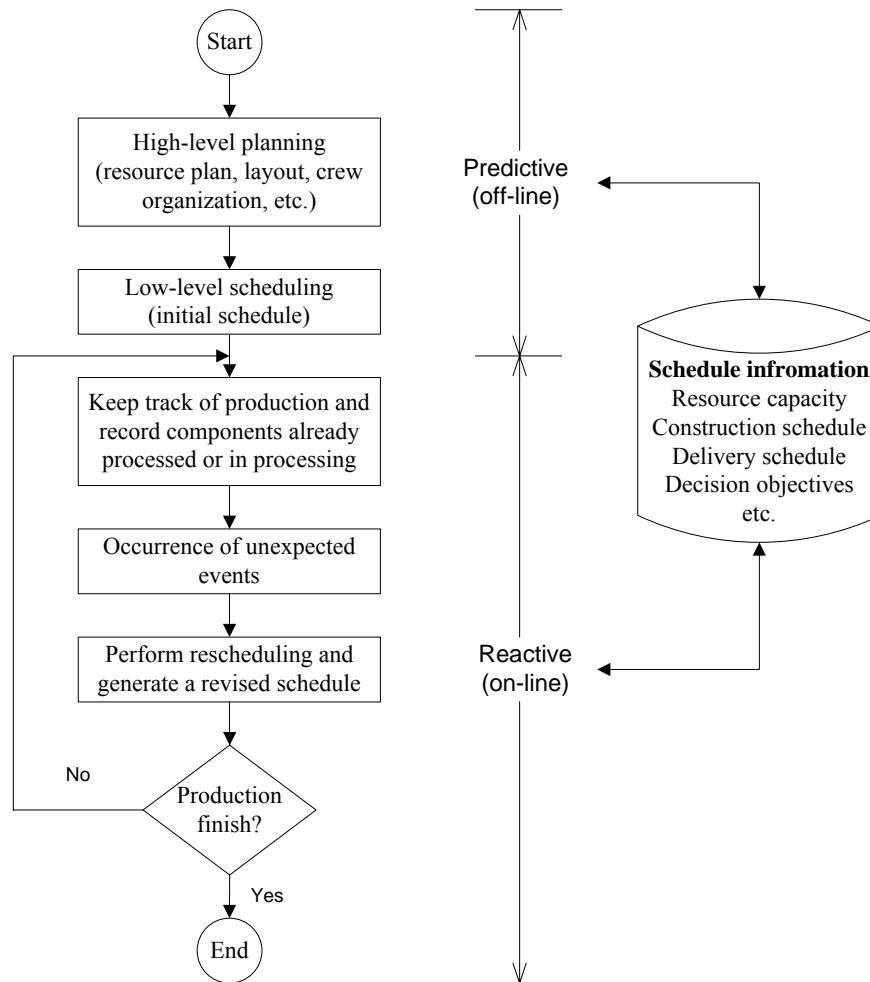


Figure 1.2 Planning and control processes for precast production

1.3.2 Occurrence of Schedule Disturbances

The generation of a precast production schedule prior to execution is based on several key inputs, such as the demand schedule for different precast components and the available capacity of various resources. Although these inputs are assumed to be known deterministically when the production schedules are decided, they are likely to change during actual project performance. Changes in the details of the delivery schedule or the availability of resources may force production schedules to change when available resources are overcommitted.

There are several common causes for schedule disturbances. As shown in Figure 1.3,

these causes are categorized as either “in-house factors” or “external factors” based on whether it is within the control of the precast factory or not. The in-house factors mainly include variations in different kinds of resources, such as production space, storage space, casting molds, skilled workers, raw materials, gantry crane, *etc.* Such factors generally result in a changed resource constraint for precast production. The external factors are associated with parties outside the precast factory, and include demand changes by the contractor, design changes by the architect, or increased demand from a new business opportunity. In this research, it is assumed that the external factors are more significant in generating schedule disturbances than the internal factors within the factory itself. Planning for the production of the precast elements in the precast factory starts before the beginning of actual site construction work. As such, the high-level resource plan and low-level production schedule are based on a baseline master schedule of the project. Amendments to the blueprints,

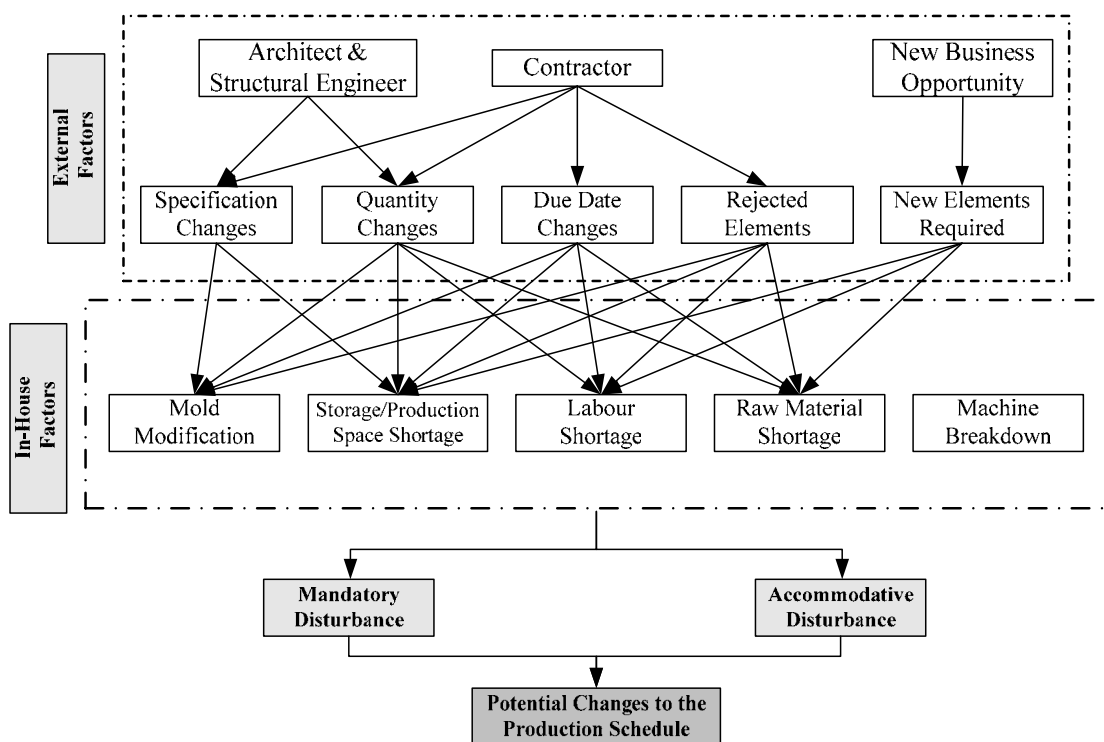


Figure 1.3 “External” and “In-house” causes for schedule disturbances

specifications, and the construction sequence of the project during the course of construction will disrupt the original production schedule. Schedule disturbances caused by these amendments are unpredictable, and the impact of such disturbances is exacerbated without timely exchange of relevant information between the precaster and the contractor.

Not all of the disturbances mentioned require adjusting the existing production schedule. For example, the precaster can ignore a new business opportunity if it is too costly to accommodate the required increase in existing production. On the other hand, local precasters take the rejection of their components because of poor quality seriously since such incidents not only cause a loss of profits but also affect the reputation of the precaster.

1.3.3 Features of Rescheduling Practices

The practice of local precasters is to create a rolling production schedule over a planning horizon. The rolling schedule is revised and updated weekly or biweekly to accommodate various types of schedule disturbances. How disturbances are resolved and the orders in which these are done are largely decided by the personnel in the factory based on their previous experience. However, the outcome could be merely “satisficing”, solving the immediate problem but causing subsequent further disturbances that cost more to resolve.

The precaster and the contractor have to reach agreement on a new delivery schedule after rescheduling. Different alternatives for revised production schedules form the basis of negotiation between these two parties. Ideally, both the precaster and the contractor ought to carry out this negotiation in a cooperative manner by taking into

account objectives such as timely delivery, reasonable inventory levels, efficient resource utilization levels, and so on. Unfortunately, these objectives may often conflict between one and another. Since rescheduling is performed concurrently with the production process, the time for obtaining a new schedule is short. Each party is therefore likely to explore only limited options for the revised schedule due to this time pressure, and these options may have been generated by adopting overly conservative assumptions that seek to optimize their own objectives. Such a circumstance will prevent both parties from having a common understanding of the problem and appreciating the differences and conflicts that stem from their respective decisions. Moreover, it will shift the focus of negotiation from the balance of trade-offs between different alternative delivery schedules to the relative importance of the objectives of the respective parties. The process of negotiation may prove tedious when the parties involved cannot agree on one of the limited options and additional time is required to generate other alternatives.

It would be more beneficial if the negotiations are supported by the inclusion of a range of alternative schedules that explore the various trade-offs between different objectives. It is probably too time-consuming and onerous to generate this range of alternatives manually, so some form of computer-aided procedure will be necessary.

1.4 Current Research in Precast Planning and Scheduling

The wider adoption of precast building components has motivated much research interest in methods for precast production, including the planning and scheduling aspects. Warszawski (1984) first proposed a mathematical model for the production

planning of precast components in the make-to-stock manufacturing system. Subsequent models for the same general problem went beyond traditional operation research techniques and applied other computer intensive methods to solve the problem. For example, Dawood and his colleagues developed a simulation-based model that integrated demand forecasting, production planning and stock forecasting for the precast industry (Dawood and Neale 1993; Dawood 1994, 1995 and 1996; Dawood and Smith 1996). Chan and Hu (2000, 2001, and 2002a) developed a flow shop sequencing model for the specialized precast production scheduling with genetic algorithms (GAs). Chan and Hu (2002b) also adopted a constraint programming (CP) approach to solve the comprehensive precast production scheduling by incorporating the constraints encountered in practice. Leu and Hwang (2001 and 2002) proposed a GA-based flow shop scheduling model to obtain optimal resource-constrained schedules for repetitive precast production. Comparison of the results obtained from these models with those obtained typically from the use of heuristic rules showed the benefits of these models for the precast planning and scheduling problem.

However, these models are predictive scheduling models that determine the schedule using deterministic inputs before actual production commences. Such schedules are vulnerable to schedule disturbances because of the nature of the precast production process noted earlier. To date, there has not been much research addressing the problems of schedule revision and coordination for precast production although these are important in practice. There is a need to extend the research on production scheduling in the precast industry to include rescheduling during the execution of previously planned schedules.

1.5 Research Objectives and Scope

This research will focus on the rescheduling problem for precast production. The research scope includes:

- (1) Identifying the factors contributing to the occurrence of disturbances in precast construction and the heuristics used to repair the production schedule;
- (2) Identifying the key constraints involved in the rescheduling process for precast production, and the criteria used by the precaster and the contractor to evaluate alternative schedules;
- (3) Formulating the precast production rescheduling problem and proposing a rescheduling model for precast production;
- (4) Developing optimization procedures for the model;
- (5) Validating the feasibility of the proposed methods and evaluating their effectiveness.

The primary objective of this research is to develop a coordinated production reactive scheduling model (CPRSM) for the precast production process. The research scope includes work to:

- (1) Develop a repair-based method for precast production rescheduling. It is impractical and potentially disruptive to generate new schedules from scratch each time a disturbance occurs. The main reason is because of the use of a rolling planning horizon when generating schedules. This kind of approach is therefore not much favored in practice as the new schedule can differ considerably from the current schedule in use. Furthermore, many other decisions like assignment of personal, delivery of raw material and subsequent processing of the jobs in other facilities may be severely disrupted. This phenomenon is commonly referred to as shop floor nervousness. Therefore, repairing existing schedules to handle the

disturbances is a more attractive alternative. The repair-based method for precast production rescheduling reassigns the production of precast elements involved in schedule disturbances with available resources, while trying to maintain most of the originally scheduled production of precast elements unchanged simultaneously. The new production schedule will deviate less from the old schedule, and results in less shop floor disruption and nervousness. In this research, the schedule repair action is achieved by deciding the best sequence of disturbance resolution and selecting the best heuristic to solve each disturbance.

- (2) Generate alternative schedules along a *Pareto* optimal front to facilitate schedule coordination. This is achieved by formulating the rescheduling problem as a multiobjective optimization problem. The model can simultaneously search for repaired schedules representing different degrees of trade-off between the objectives of the precaster and the contractor. A multiobjective optimization approach for the exploration of trade-offs between conflicting objectives avoids the debate as to the “correct” choice of the weights for these objectives. The search algorithm for the identification of *Pareto* optimal solutions is based on the GA.
- (3) Develop a local search process to conduct incremental exploration in specific areas identified by the parties involved. The local search process can improve existing schedules by incorporating specific requirements from either the precaster or the contractor, and is complementary to the search process described in (2).

1.6 Research Methodology

The research methodology consisted of interviews with industry professionals,

literature review, model development, and verification of the rescheduling methodology with cases.

In this research, a review of the literature on precast production scheduling research first revealed that, among researchers the problem of rescheduling/reactive scheduling had received less attention than predictive scheduling, although rescheduling occurred frequently in practice. It became clear that including the concept of rescheduling in existing precast production planning and scheduling models would be the next logical step in the development of computer-based production planning and scheduling models.

With this research objective, interviews with industry professionals were then conducted to: (i) understand the current practice in precast production rescheduling; (ii) determine the objectives and constraints used in repairing actual schedules; and (iii) map the process involved in schedule coordination and the factors that could cause a disruption in the precast supply chain.

Based on the information obtained from the interviews with industry professionals, it was realized that multiple conflicting objectives were involved in the actual precast rescheduling process. A true multiobjective optimization model was therefore necessary in the model development. This would improve the current practice of weighting criteria to form a single objective. As a global search and optimization technique, GAs were identified as an appropriate method to solve the precast production rescheduling problem with multiobjective optimization, and find good candidates for repaired schedules. Further feedback from the industry also identified the need for a concept to find schedules with particular characteristics and a corresponding local search procedure. Some local search techniques were developed

for use in the search procedure based on the mechanism of iterative improvement. The end result was a coordinated production reactive scheduling model for precast production that incorporated these two search procedures.

Finally, the model was tested on several case studies in order to verify the feasibility and practicability of these two search procedures. The first set of cases illustrated the efficiency and effectiveness of the GA-based algorithms in generating schedules along a *Pareto* front under different mold utilization levels. The second set of cases demonstrated the capability of the local search techniques in finding such particular schedules with specific constraints.

1.7 Thesis Organization

This thesis is organized into seven chapters, beginning with this chapter. *Chapter Two* reviews the previous research on precast planning and scheduling models. Reactive scheduling, multiobjective optimization, as well as techniques available to solve these two problems are also briefly introduced. The chapter also includes a review of the single and multi-objective genetic algorithms, particularly their application to scheduling problems.

Chapter Three describes the development of the proposed coordinated production reactive scheduling model (CPRSM), including the mathematical formulation of variables, constraints and objectives considered, and functions for each key process within the model.

Chapter Four describes key issues of GAs in multiobjective search, some well-established multiobjective GAs and elitist strategies proposed in this research,

and demonstrate the application of GAs in the global search procedure.

Chapter Five verifies the implementation of GAs in generating alternative repaired schedules along a *Pareto* front in three examples with different levels of mold utilization rate. The performances of several selected GA-based algorithms are compared.

Chapter Six describes the local search process for an incremental exploration of schedules through the incorporation of specific constraints. The proposed local search techniques are tested with two examples to illustrate the iterative repair mechanism.

Chapter Seven concludes the thesis with a summary of the main contributions of the research, its limitations and suggestions for future study.

CHAPTER 2

LITERATURE REVIEW

This chapter first reviews previous research on planning and scheduling models for precast production. Since the present research aims to develop a reactive scheduling model for precast production, two research fields: i) the reactive scheduling problem and existing approaches; and ii) the multiobjective optimization problem and related methods, are then reviewed. Finally, genetic algorithms (GAs) are introduced with an emphasis on their application in the field of scheduling problems with multiobjective optimization, as the GA is the basis of the search approach adopted in the research.

2.1 Planning and Scheduling for Precast Production

The wider use of precast building components has motivated the interests of many researchers in the precast industry. Their works range from generating resource plans and detailed schedules for precast production to developing integrated information systems to aid and facilitate decision-making and coordination in the precast production process, especially for planning and scheduling.

Several planning and scheduling models for the precast industry have been proposed from the mid of 1980's. With applications of inventory control methods and heuristic rules in the early years, precast planning and scheduling problems are more recently solved with many advanced techniques in computer science, information technology and decision science in these models. With the help of these advanced techniques, the problem modeling has reflected the precast production process more practically.

Moreover, these techniques become feasible and reliable to find better solutions to the precast planning and scheduling problem. Some planning and scheduling models that have been developed are reviewed as follows:

(1) Mathematical models

Warszawski and Ishai (1982) proposed a long-range planning model for the prefabrication industry, which includes determinations of the optimal location and capacity of prefabrication plants from the perspective of a national economy. After that, Warszawski (1984) developed another model for the short range planning in prefabrication plants. This model was proposed as a classical optimization problem to decide the minimum cost assignment of precast elements to molds in the plant by applying the inventory slope theory. It provided schedules for two production series, namely short (specific orders) and long (continuous demand for standard elements) by considering some constraints encountered in practice. These early studies have concentrated on the application of mathematical models on the planning and scheduling problem with a make-to-stock manufacturing system. Although the proposed mathematical models for the short range planning represented production operations in a prefabrication plant to some extent, not all necessary constraints were identified (Hu 2000). One of the reasons is that large number of parameters would increase computational difficulty in finding an optimal solution for the mathematical model.

(2) Capacity planning model

In order to help production managers make better planning decisions, Dawood and Neale (1993) developed a capacity planning model for the precast industry. The model is a “finite rough-cut” capacity planning system that develops a twelve-month

capacity plan using a backward scheduling technique. Being incorporated the demand and stock forecasting, this model was developed as an integrated production management system for the precast industry (Dawood 1994, 1995 and 1996; Dawood and Smith 1996). Simulation technology was applied in the model to automate the planning process and predict the effect of several schedule strategies that are basically heuristic rules for product selection and plant allocation. The capacity planning model went beyond the use of standard stock control models in the precast industry. Due to the rule-driven nature and its ability to mimic the decision making of a human planner, the simulation approach was concluded to be able to complement human knowledge through eliciting scheduling knowledge under different circumstances. However, the model was developed with the aim of evaluating alternative planning options before actual production commences as indicated in the literature, and based on sinusoidally varying demands. Therefore, the model cannot realistically reflect the real conditions in the precast plant and is incapable of real-time scheduling of production driven by contracts.

(3) Production simulation models

Production in precast factories that is organized with different contracts is full of random factors in practice, such as differences in individual component size and shape, variations in site progress and randomness in production times. Some researchers applied simulation technologies to analyze difficulties of precast production under such circumstance. Halpin and Riggs (1992) developed a CYCLONE model for the precasting process. The system studied changes in task durations, crane availability, rate of supplying concrete and crew size for precasting. Vern and Gunal (1998) proposed a simulation model that is capable of capturing various random elements and facilitating the analysis of complicated what-if scenarios for precast production.

Balbontín-Bravo (1998) also applied simulation to analyze the optimization of precast production in a workshop and obtain working alternatives to improve productivity. However, these simulation models only serve as an early warning system, which can help production managers find out bottlenecks in the precast production process and adjust resource allocation to increase productivity before the production process commences.

(4) Processing scheduling models

The production in precast factories is organized and carried out similarly to that in other manufacturing industries. This similarity opens an opportunity of applying production process scheduling models to the precast industry, which have been widely applied to the manufacturing systems (Baker 1974). Chan and Hu (1998) first identified that the flow shop scheduling model was most relevant to operational conditions for precast factories in Singapore. Thereupon, Chan and Hu (2000, 2001 and 2002a) proposed a flow shop sequencing model (FSSM) for the specialized production method applied in local precast factories. With this production method, the total process of precast element producing is broken into several activities that are performed by different crews with specialized tools and work methods (Warszawski 1990). The traditional makespan and the more practical tardiness penalty objectives were minimized separately, as well as simultaneously using a weighted approach. The GA and classical heuristic rules were applied in the model to generate production schedules. From the experiments conducted, a conclusion was highlighted that the GA can obtain good schedules by giving a family of solutions that are at least as good as those produced by heuristic rules. Almost at the same period, Leu and Hwang (2001 and 2002) developed another flow shop precast scheduling model. A GA-based searching technique was also adopted in the model to provide the optimal or

near-optimal combination of production sequences, resource utilization and minimum makespan in consideration of resource constraints. Results of these flow shop scheduling models indicated that the flow shop process is suitable for modeling precast production, and the GA-based search is efficient and flexible for solving the problem. GAs therefore become a feasible option for solving the precast rescheduling problem studied in this research.

(5) Constrained scheduling model

Except for the specialized production method, there is another work organization known as the comprehensive production method (Warszawski 1990). With this method, a same crew performs all activities of the precast production until the product is finished. Chan and Hu (2002b) proposed a constrained precast scheduling model (CPSM) for the comprehensive precast production. Their model was formulated by incorporating several key constraints and objectives considered in real practice. A capacity-based backward scheduling earliest due date (EDD) rule and a constraint programming (CP) approach were developed to solve the model. Even though it incorporates many problem-derived constraints, comparisons of results by the CP and commonly used heuristic rules showed that the CP is computationally efficient. In order to improve the CP approach, Hu (2000) proposed a hybrid GA-CP approach to solve the CPSM, in which the GA was used to determine high-level control parameters or a sequence of tasks that directs the CP in its search for detailed scheduling solutions. One of the significances of the CPSM is that it is a natural way of representing the scheduling problem for the comprehensive precast production. Compared to the above-mentioned models, this model highlighted the more practical issue of constructing “optimal” production schedules in the face of many real constraints in practice. In the precast production rescheduling process, these constraints also should

be considered since any constraint applied in the initial scheduling process must also be observed in rescheduling.

(6) Information system models

The precast industry suffers from poor information transfer and management. This phenomenon has caused extra burden on the industry resulting from inaccurate planning and inefficient utilization of resource. Therefore, information technologies has the potential to modernize the industry in the field of planning and scheduling by making the information transfer quicker and more efficient, as well as better cooperation and collaboration among the parties involved. Dawood (1999) introduced an integrated intelligent computer-based information system for the precast industry. The framework of this proposed information system consisted of two portions: i) use of information technology in different phases and tasks of manufacturing process; and ii) integration of these phases and tasks through the use of digitally stored data and data transfer. Chan *et al.* (1999) proposed an object-oriented design of a collaborative precast scheduling system. It addressed a specific type of distributed problem solving that enables the sharing and exchange of related information. The system comprised two subsystems corresponding to two key functions during the scheduling process, which are (i) handling the static database management, and (ii) communicating and processing messages. These research works in the development of information systems highlighted the characteristic of dynamic for the precast production process, the importance of information flow therein, and the necessity of coordination both inside the factory and with outside parties.

In summary, the research in the precast planning and scheduling problem has made progress in modeling the corresponding process, developing artificial intelligent based approaches such as the GA and CP to generate better solutions, and facilitating

decision making with the help of information systems. However, most of these models only deal with predictive scheduling for precast production, which focus on generating production schedules before actual production commences. To date, there has not such research addressing the precast production rescheduling problem although rescheduling is a practical issue, and is important in the light of unexpected changes due to construction site events. Therefore, the present research work will focus on the issue of schedule disturbances and look into the rescheduling problem for precast production to complement the earlier predictive scheduling models.

2.2 Reactive Scheduling

Scheduling exists in a wide variety of domains, such as production scheduling, project scheduling, employee shift scheduling, school timetable scheduling, *etc.* All of them are faced with a certain number of uncertainties in reality. Therefore, scheduling problems are complicated due to dynamic environments, where rescheduling in response to changes is critical. Due to the shortage of research in precast rescheduling and similarities between the production process in precast factories and assembly lines in the chemical and electronic industries, it is necessary to review the research of reactive scheduling/rescheduling in these manufacturing industries and extend it to the area of precast production.

2.2.1 Overview

Generally, the scheduling process is an attempt to utilized limited resources in a manner as efficient as possible, as well as simultaneously satisfying several domain-specific constraints (Noronha and Sarma 1991). However, real production

scheduling problems are dynamic in nature as Graves (1981) stated, “*A frequent comment heard in many scheduling shops is that there is no scheduling problem but rather a rescheduling problem*”. Therefore, there are two classes of scheduling problems, which are predictive and reactive scheduling (Suresh and Chaudhuri 1993). They may be seen as complementary activities (Smith 1994).

Predictive scheduling is also called static scheduling in the literatures. Being considered to operate “off-line” before the system starts operation, predictive scheduling generates a solution for all available jobs within the entire planning horizon. It is specified by addressing the problem with deterministic inputs, such as processing times and resource availability. However, this schedule is subject to changes due to external and internal events e.g. delayed or shortened execution of an operation, machine breakdown or an additional high priority order.

On the other hand, reactive scheduling is called dynamic scheduling, real-time scheduling or rescheduling (Sun and Xue 2001). Reactive scheduling, which be conducted “on-line”, can be broadly defined as the reactive part of the system monitoring the execution of the pre-determined schedule and coping with unexpected events. The main alternatives to the revision of a schedule in the presence of real time feedback are either to incorporate the new information by completely regenerating the original schedule from scratch, or by “repairing” the previous schedule in some way (Dorn *et al.* 1996). The first approach might be better capable of maintaining optimal solutions in principle, but such solutions may be rarely achievable in practice and computation times are likely to be prohibitive. Furthermore, frequent schedule regeneration can result in increased costs attributable to what has been termed “shop floor nervousness” (Raheja and Subramaniam 2002). Thus, most approaches to

reactive scheduling are based on infrequent regeneration of a basic predictive schedule. It is then maintained continuity by serving as a nominal reference for the identification and specification of schedule changes as it is progressively modified.

2.2.2 Approaches in Reactive Scheduling

The character and content of the scheduling research have always kept pace with theoretical developments in operations research and computer science areas. A variety of techniques and approaches applied in reactive scheduling has been reported in these years.

Mathematical programming such as linear programming, integer programming and later multiple objective programming, as well as search techniques such as branch and bound and heuristic rules, are implemented widely in the scheduling research. However, the scope for these conventional approaches is limited due to complexity of the scheduling environment. Prohibitive computing times with these approaches are not applicable in industrial practice. Much of the research using these approaches in turn involves highly simplified versions of the actual problem, which can only model a fragment of the scheduling knowledge existing in practice. Therefore, scheduling models with these conventional approaches lack the level of fidelity and adaptability to changes (Szelke and Márkus 1995). Furthermore, the static nature of these approaches is not suitable to the dynamic nature of most real-time scheduling environments.

Over the last two decades, developments in artificial intelligence (AI) technologies paved the way for a host of new approaches to tackle the reactive scheduling problem. AI uses symbol-processing computer programs to strive for human-like performances.

AI-based approaches offer the well-known advantages of symbolic systems, which are easy to understand and perform like human experts.

Knowledge-based approaches provide the major AI approaches to production scheduling in particular. The fundamental characteristic of this kind of approaches is employing domain specific problems solving information to derive good schedules with reasoning, which is easily understood and accepted by the human user. Several knowledge structures have been described in the literature, such as rules, frames/schema, procedural nets, scripts, *etc.* Rules, expressed as if-then statements, figure prominently as knowledge for use within production rescheduling knowledge-based systems (Subramanyam and Askin 1986, O'Grady and Lee 1989; Tsukiyama *et al.* 1992; Li and Shaw 1996 and 1998; *etc.*).

With the perspective of scheduling as a constraint-driven process has lately become more and more dominant, the constraint-based reasoning is frequently employed for rescheduling (Kjenstad 1998). These constraint-based systems use constraints to measure the quality of candidate schedules and to prune alternative assignments during the scheduling/rescheduling process. Among them, ISIS developed by Fox and Smith (1984) and OPIS proposed by Smith *et al.* (1990, 1994) which is the successor of the ISIS are the most well known systems.

Multi-agents in a distributed artificial intelligence (DAI) environment have been widely reported in the works of the reactive scheduling problem, for instance, Szelke and Markus (1995), Tsukada and Shin (1998), Pendharkar (1999), Tranvouez *et al.* (2001), Chun and Wong (2003), *etc.* In DAI-based approaches, the reactive scheduling is achieved using multiagents, which possesses the knowledge pertaining to schedule repair. These independent intelligent agents coordinate their knowledge

and solve sub-problems while working toward a common goal for rescheduling. The human scheduler can also act as one of the intelligent agents and become actively involved in the decision process.

Other advanced artificial intelligent techniques, such as neural networks and genetic algorithms, have also been widely applied in solving the reactive scheduling problem. These techniques generally use biological performance in problem solving that start with the neuron or other basic building blocks and exploit these blocks to achieve intelligent behavior (Brown *et al.* 1995). The importance of production rescheduling and the promise of improved performance using these approaches have encouraged a number of researchers to investigate methods from this area, such as Garetti and Taisch (1995), Fang and Xi (1997), Kim *et al.* (1998), Bierwirth and Mattfeld (1999), Rovithakis *et al.* (2001), Nishimura *et al.* (2001), and Madureira *et al.* (2002), *etc.* These studies indicate that these techniques can offer excellent potential for adapting scheduling algorithms to changing conditions, and can be successfully combined with each other to enhance the search for good solutions in reactive scheduling.

Reviews of the AI-based approaches and systems in reactive scheduling have been provided by Suresh and Chaudhuri (1993), Szelke and Kerr (1994), Brown *et al.* (1995), Kjenstad (1998), Sabuncuoglu and Bayiz (2000), Raheja and Subramaniam (2002), *etc.* Artificial intelligence techniques have been identified as efficient methods to tackle the combinatorial exploration in the reactive scheduling problem. They can provide a large body of tools and techniques such as informed search methods, sophisticated knowledge representation schemes and high-level programming environments.

2.3 Multiobjective Optimization Problems

As discussed earlier in Chapter 1, the rescheduling problem for precast production is indeed a multiobjective optimization problem. Therefore, basic concepts and approaches for the multiobjective optimization problem are reviewed in this section.

2.3.1 Basic Concepts and Terminologies

Multiobjective optimization problems (MOPs) also called multicriteria optimization, multiperformance or vector optimization problems. This class of problems can be defined as the problem of finding (Osyczka 1985): “[...] a vector of decision variables which satisfies constraints and optimizes a vector function whose elements represent the objective functions. These functions form a mathematical description of performance criteria which are usually in conflict with each other. Hence, the term ‘optimize’ means finding such a solution which would give the values of all the objective functions acceptable to the decision maker.” Without loss of generality, the mathematical formulation of a minimization MOP is presented as follows:

$$\begin{aligned}
 \text{Min } y &= f(x) = \{f_1(x), f_2(x), \dots, f_m(x)\} \\
 \text{subject to } g(x) &= \{g_1(x), g_2(x), \dots, g_j(x)\} \geq 0 \\
 h(x) &= \{h_1(x), h_2(x), \dots, h_k(x)\} = 0 \\
 \text{where } x &= \{x_1, x_2, \dots, x_n\} \in X \\
 y &= \{y_1, y_2, \dots, y_m\} \in Y
 \end{aligned} \tag{2.1}$$

where, x is the vector of decision variables, y is the objective vector, X is the decision space, and Y is the objective space.

The performance measure of multiple objectives is not as straightforward as that of a single objective. At times when there are multiple conflicting objectives involved in the problem, it is not possible to optimize several objectives simultaneously. The concept of dominance is used to compare the performance of different solutions with

multiple objectives. Suppose a bi-objective minimization problem shown in Figure 2.1, five points, namely from A to E, are plotted in the objective space. As shown in the Figure 2.1, it is difficult to judge which is superior among points A, B and C, because none of these points is better than the others with respect to both criteria. Since there is no point that is better than these three points along both dimensions, points A, B and C are called non-dominated or non-inferior points for the problem considered. On the other hand, point D is bettered by point B and point E is bettered by point D in both criteria. Therefore, points D and E are known as dominated points, and both dominated by points A, B and C.

Based on the concept of dominance, a decision vector x^* is *Pareto optimal* if there exists no feasible vector x which would decrease some criterion without causing a simultaneous increase in at least one other criterion (Coello *et al.* 2002). Therefore, rather than a single solution, the “optimum” for the MOP is usually a set of equally efficient, non-inferior or non-dominated solutions, known as *Pareto optimal set*. The plot of the objective functions whose nondominated vectors are in the *Pareto optimal set* is called *Pareto front*.

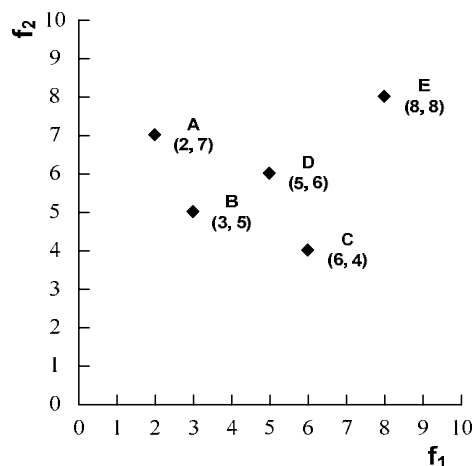


Figure 2.1 Illustration of domination in multiobjective optimization

2.3.2 Multiobjective Optimization Methods

In this section, a few commonly used classical methods for handling the MOP are reviewed first, which are different from the methods based on the evolutionary algorithm reviewed later.

(1) Classical methods

Classical multiobjective optimization methods have been existed for at least four decades. Most of these methods suggest a way to convert a multiobjective optimization problem into a single objective optimization problem. Detailed reviews of these classical methods for the MOP are available in (Steuer 1986). Some representatives of this class of methods are the weighted sum method (Zadeh 1963), the ε -constraint method (Marglin 1967), the weighted metric method (Zeleny 1973), the goal attainment method (Gembicki 1974), the multiattribute utility method (Keeney and Raiffa 1976), the goal programming method (Goicoechea *et al.* 1982), the lexicographic method (Rao 1984), *etc.* There are a number of difficulties accompanying these classical optimization methods (Deb 2001):

- (i) Only one *Pareto* optimal solution is expected to be found in one simulation run. In order to obtain an approximation of the *Pareto* optimal set, several optimization runs are required but it may cause high computation overhead.
- (ii) Some methods may be sensitive to the shape of the *Pareto* front. For example, not all *Pareto* optimal solutions can be found by some methods in nonconvex MOPs.
- (iii) All methods require some problem knowledge, such as suitable weight vectors and target values. However, such problem knowledge may not be available.

(2) Evolutionary algorithm based methods

The field of search and optimization has changed over the last few years by the

introduction of a number of non-classical, unorthodox and stochastic search and optimization algorithms. Of these, evolutionary algorithms (EAs), which mimic evolutionary principles of the nature to drive the search towards optimal solutions, seem to be especially suited to multiobjective optimization. It is because they are able to capture multiple *Pareto* optimal solutions in a single simulation run and may exploits similarities of the solutions by recombination. Some researchers suggest that the multiobjective search and optimization might be a problem area where EAs do better than other blind search strategies (Fonseca and Fleming 1995; Valenzuela-Rendón and Uresti-Charre 1997).

The need for finding multiple trade-off solutions in one single simulation run was suggested and worked out by Schaffer in 1984. However, no attention had been paid to multiobjective optimization for almost half a decade after Schaffer's study until Goldberg suggested a multiobjective evolutionary algorithm using the concept of domination in 1989. Since then a number of researchers have been motivated from his work and developed different implementations of multiobjective evolutionary algorithms (MOEAs). Of these, Hajela and Lin's weight-based approach (1992), Fonseca and Fleming's multi-objective GA (MOGA) (1993), Srinivas and Deb's non-dominated sorting GA (NSGA) (1994), and Horn, Nafpliotis and Goldberg's niched *Pareto*-GA (NPGA) (1994) were tested for different real-world problems to demonstrate that the *Pareto*-based MOEAs can be reliably used to find and maintain multiple trade-off solutions. Almost at the same time, a number of other researchers also suggested different ways to use an EA to solve multiobjective optimization problems, such as Kursawe's diploidy approach (1991), Osyczka and Kundu's distance-based GA (1995), and Ishibuchi and Murata's multiobjective genetic local search algorithm (1996). Comprehensive reviews and comparison studies on these

multiobjective evolutionary algorithms have been provided by Horn (1997), Zitzler and Thiele (1999), Zitzler *et al.* (2000), Coello (2000 and 2001), Van Veldhuizen and Lamont (2000); Carlyle *et al.* (2001), Deb (2001); Coello *et al.* (2002), Collette and Siarry (2003), *etc.*

2.4 Genetic Algorithms and Applications to Scheduling

Due to its successful applications in both the multiobjective optimization and the reactive scheduling problem, the basic mechanism of genetic algorithms (GAs) is reviewed in this section. It is followed by their application to scheduling problems, especially with multiobjective optimization.

2.4.1 Overview of GAs

As the name suggests, the processes observed in natural evolution inspired genetic algorithms. The GA is an example of a search procedure that uses random choice as a tool to guide an exploitative search through a coding of parameter space (Tam *et al.* 2001). In the parameter space, a new set of artificial creatures is created using bits and pieces of the fittest of the old. An occasional new part is tried for good measure. While randomized, GAs are no simple random walk but efficiently exploit historical information to speculate on new search points with expected improved performance.

Genetic algorithms are different from many normal optimization and search procedures in four ways (Goldberg 1989):

- (1) GAs work with a coding of parameter set, not the parameters themselves,
- (2) GAs search from a population of points, not a single point,

(3) GAs use payoff (objective function) information, not derivatives or other auxiliary knowledge, and

(4) GAs use probabilistic transition rules, not deterministic rules.

These four differences contribute to genetic algorithms' robustness and resulting advantages over other more commonly used techniques (Dandy and Engelhardt 2001).

Therefore, as powerful and broadly applicable stochastic search and optimization techniques, GAs are perhaps one of the most widely known types of evolutionary computation methods today which results in a fresh body of research and applications.

2.4.2 Multiobjective Genetic Algorithms

By working with a population, GAs have a built-in advantage of being able to work with multiobjective optimization problems. Indeed, a set of solutions is sought with multiobjective optimization, whilst the GA population can deliver several solutions of the efficient set (Yapo 1996). According to different fitness assignment strategies, there are three broad ways for GAs of working with multiobjective optimization (Fonseca and Fleming 1995, Zitzler and Thiele 1999), they are:

(1) Aggregation approaches: For this kind of approaches, the objectives are combined into a scalar function. There are numerous ways of aggregating the problem, such as objectives weighing, distance function and min-max formulation, *etc.* Then the single objective problem is solved by a traditional GA to produce one single solution.

(2) Population-based non-*Pareto* approaches: These approaches treat objectives separately as a move toward finding multiple non-dominated solutions with a single GA run. As a typical example, the vector evaluated genetic algorithm (VEGA) proposed by Schaffer (1984 and 1985) divides the population into as

many sub-groups as the number of objectives. Each sub-group is then assigned the task of optimizing one objective function. The individuals that perform well in each sub-group are selected for generating the next set of possible solutions.

- (3) *Pareto*-based approaches: These approaches rank the performance of individuals based on the concept of *Pareto* optimality, striving to guarantee equal probability of reproduction to all non-dominated solutions. Compared to non-*Pareto* approaches that are often sensitive to the nonconvex trade-off surfaces, the *Pareto*-based approaches do not raise such issue. MOGA, NSGA, and NPGA are all representatives in this class of approaches. For example, MOGA proposed by Fonseca and Fleming (1993) ranks the individual corresponding to the number of individuals in the current population by which it is dominated. NSGA proposed by Srinivas and Deb (1994) is developed from the ranking mechanism proposed by Goldberg (1989). The ranking process is executed front wise by assigning the individuals in each non-dominated front a same dummy fitness value.

2.4.3 Applications to Scheduling Problems

GAs are so efficient that they can find the optimal or near-optimal solution in a reasonable time even for the traditional NP-complete problems. Being referred to as NP-hard, the scheduling problems that are optimized with a single objective have already been solved by GAs in a vast number of studies. GAs have shown characteristics of domain independence, robustness and flexibility in solving such problems. In recent years, GAs have been widely applied to scheduling problems with multiobjective optimization in several specific fields.

- (1) Production scheduling problems: For example, Shaw and Fleming (1996, 1997) solved a production scheduling problem in a ready meals factory by MOGA with

optimization of makespan and tardiness penalty simultaneously. Li and Man (1998) applied MOGA to solve an extensive earliness/tardiness production scheduling and planning problem with lot-size consideration and capacity balance. Sankar *et al.* (2003) applied the GA to generate a nearer-to-optimum production schedule with two contradictory objectives of the flexible manufacturing system.

(2) Process scheduling problems: Being a major application area, various kinds of multiobjective GAs have been applied in solving process scheduling problems. For example, Murata *et al.* (1996) and Ishibuchi and Murata (1998) solved the flow shop scheduling problem by hybridizing a GA with a linear combination of weights with a local search procedure, in which the weights are randomly generated at the time of performing recombination. Todd and Sen (1997) applied a multiple criteria GA in solving the job shop scheduling problem to optimize the makespan and the average job time simultaneously. Bagchi (1999, 2001) adopted NSGA and a variation of NSGA using elitism to solve both flow shop scheduling and job shop scheduling problems. Brizuela *et al.* (2001) and Brizuela and Aceves (2003) applied NSGA to solve a three-objective flow shop scheduling with different types of selection, crossover and mutation. Other applications can be found in Tamaki *et al.* (1999), Cui *et al.* (2001), Talbi *et al.* (2001), Chang *et al.* (2002), *etc.*

(3) Machine scheduling problems: Cochran *et al.* (2000) proposed a two-stage multi-population genetic algorithm (MPGA) to solve a parallel machine scheduling problem with three objectives. Results of MPGA were compared with those of the multiobjective genetic local search algorithm proposed by Murata *et al.* (1996). Carlyle *et al.* (2001) also compared these two algorithms in solving the bi-objective parallel machine scheduling problems.

(4) Real-time scheduling problems: Montana *et al.* (1998) solved large-scale real-time scheduling problems with a weighted sum GA. Two examples were considered, which are a field service scheduling problem with seven cost objectives and a military land move scheduling problem with two objectives. Oh and Wu (2004) solved a real-time task scheduling problem in the microprocessor system with GAs, which considers two criteria independently by using a vector-valued cost function. Rangsaritratsamee *et al.* (2004) applied the genetic local search algorithm in the proposed dynamic rescheduling methodology that uses multiobjective performance measures.

(5) Other scheduling problems: Except for the above applications, GAs are also applied in the scheduling problems in other domains. Some of these works are the time-tabling problem (Paechter *et al.* 1998), the telephone operator scheduling problem (Yoshimura and Nakano 1998), the nurse scheduling problem (Jan *et al.* 2000), the pavement maintenance programming problem (Fwa *et al.* 2000), the earthmoving operations problem (Marzouk and Moselhi 2004), the project time-cost optimization problem (Zheng *et al.* 2004), *etc.*

The results obtained by these works indicate that GAs are effective in handling a variety of objectives available in scheduling problems, as well as fast and promising in providing a set of non-dominated solutions for these problems.

2.5 Summary

Previous literature review enables us to draw some conclusions and consequently identify several issues that this research attempts to address:

Firstly, most of the developed precast production planning and scheduling models did not consider the rescheduling problem. Production schedules generated by these models are vulnerable and lack the capability of accommodating different kinds of schedule disturbances identified in this research. It is necessary to extend the research in precast planning and scheduling to cover the topic of how to respond to these disturbances and resolve them efficiently.

Secondly, advanced developments in computer sciences, information technology and decision sciences have become available to precisely model and find better solutions to the precast production scheduling problem. Similarities between the production process in precast factories and assembly lines in other manufacturing industries open a possibility of transferring research findings and practical experiences of scheduling between these two processes. Therefore, the precast production rescheduling problem is an area of research where formal optimization models can be beneficially employed.

Thirdly, reactive scheduling is also called rescheduling, dynamic scheduling, and “on-line” scheduling in the literatures. Artificial intelligence techniques have achieved promising results in this field of research. Among them, GAs have attracted attentions as an effective method. The reason is that GAs are easy to implement and fast to run, and therefore can be taken into consideration as a “reasoning” methodology for reactive scheduling in live environment. Furthermore, GAs have also been successfully applied in several precast scheduling models, and the results of these models showed that GAs are computationally simple yet powerful in search for optimal solutions. All of these make GAs stand for one of the most promising techniques in solving the rescheduling problem for precast production.

Finally, compromises and trade-offs are realized to be continually necessary in the scheduling process due to multiple conflicting objectives involved in reality. It is just the situation for the precast production rescheduling problem studied in this research. Hence, it becomes increasingly unrealistic to concentrate on a tool that optimizes only one objective and provides only one solution. Flexible scheduling methods that can deal with these multiple, often-conflicting objectives simultaneously and provide a set of *Pareto* optimal solutions without pre-determined weights for each objective are required. Therefore, multiobjective genetic algorithms that can meet these requirements appear to be well suited in this research.

CHAPTER 3

PRECAST PRODUCTION RESCHEDULING

In this chapter, a coordinated production reactive scheduling model (CPRSM) is proposed. The model supports four key elements of the precast production rescheduling process, namely disturbance detection, global search with multiobjective optimization, local search with specific constraints, and ranking of outcomes for negotiation. A computer program that implements these rescheduling elements has been developed to demonstrate the operation of the CPRSM.

3.1 Precast Production Rescheduling Problem

The underlying production process for the precast production scheduling and rescheduling problems is the same. The rescheduling problem only occurs when the existing production schedule has to be modified due to the occurrence of schedule disturbance. Therefore, the precast production scheduling and rescheduling problems share the same production schedule representation, production constraints and the objectives to be optimized. Characteristics of the production process adopted in local precast factories, as well as constraints and objectives considered for production scheduling/rescheduling are described firstly in this section. This is followed by a mathematical formulation of the precast production rescheduling problem addressed in this research.

3.1.1 Overview of Precast Production Process

Production in local precast factories is typically set up as a make-to-order

manufacturing system based on a comprehensive work organization, wherein the same team of workers performs all operations necessary to produce precast elements on stationary molds. Due to the prevalence of heterogeneous elements in projects, the factory generally allocates resources and arranges production for each individual project. Each project needs several different types of precast elements, in which several types of similar elements with minor variations are organized as an element group. Production of the element group is organized around a corresponding group of molds that may consist of several similar mold types. A mold changeover among these types of molds is needed to shift from the production of one element type to another within the same element group. Such a changeover can be accomplished within the same day except for very complicated cases. Figure 3.1 illustrates the relationship between the precast elements and the molds.

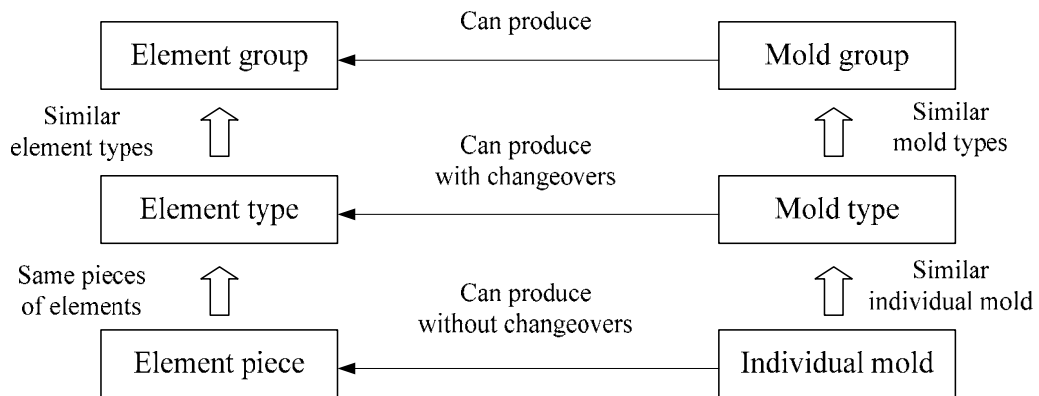
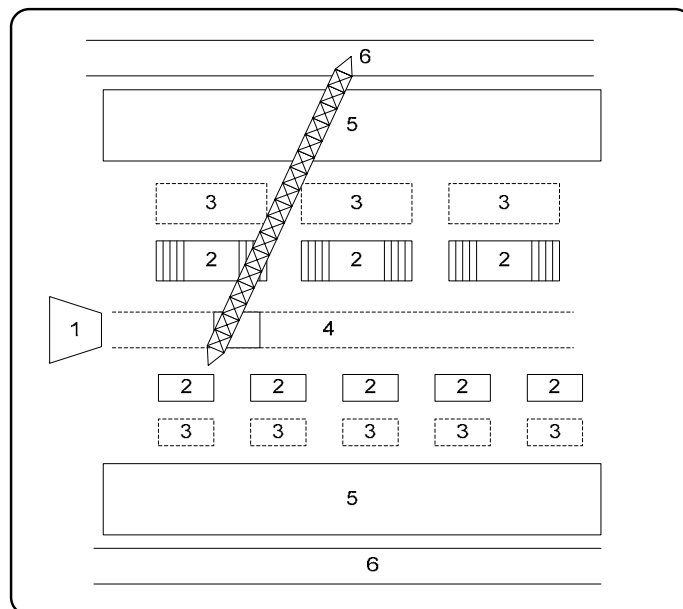


Figure 3.1 Relationship between elements and molds
(Adapted from Chan and Hu 2002b)

When a new project comes in, the precaster first classifies precast elements and groups them by type. The number of molds in each group is determined by considering the production capacity of the molds, the volume and timing of the delivery requirements, and the budget allocated by the factory. The production of

different element groups is carried out in one or more production lines with a layout such as depicted in Figure 3.2. Typically, the production process in a precast factory consists of the following sequence of activities:

- (1) Preparing and handling materials required – mixing concrete in a mixer, and assembling the reinforcement mesh;
- (2) Setting of molds – cleaning, oiling of mold surfaces, and fastening of side frames;
- (3) Placing of reinforcements, fixtures, electrical conduits and inserts;
- (4) Casting – pouring, compacting, and leveling of concrete;
- (5) Curing through an artificial or natural air curing process;
- (6) Demolding – stripping the side frame and taking out the components;
- (7) Finishing, patching and repairing components; and
- (8) Lifting the components with a mobile crane, and placing them in the stockyard to achieve the required strength; they may then be transported to the construction site.



- | | |
|--|------------------|
| 1 - Concrete mixing center | 4 - Tower crane |
| 2 - Molds | 5 - Stock yard |
| 3 - Space for Reinforcement mats and side frames | 6 - Loading area |

Figure 3.2 Layout of production line in a precast factory

3.1.2 Production Constraints

Production rescheduling has to satisfy a variety of constraints encountered in practice, including delivery constraints, capacity constraints for different resources, functional constraints, *etc.*; these are described further below:

(1) Delivery constraints: This kind of constraint specifies the due date, quantity and element type required to be delivered to the construction site. Meeting delivery schedules is a key concern of production scheduling and rescheduling.

(2) Capacity constraints: These constraints specify the limits of resources used within the production process. For example, molds are critical resources as casting is a key activity in precast production. Rescheduling the production of an element involves reassigning its production to an available mold within the production schedule. The availability of suitable molds that can produce the element, as well as the mold production capacity that means how many elements a mold can produce per day, will restrict the choices available during rescheduling. The high cost of land and the bulky nature of most precast elements also make the precast factory pay more attention to the storage space than other manufacturing industries. Although the precaster does keep a minimum number of elements as a buffer against unexpected events, the total number of precast elements for a project kept in the storage yard cannot be too large.

(3) Functional constraints: This kind of constraint specifies details of the basic working procedures required for the production of completed precast elements. For example, a minimum lead time between production and delivery must be observed for the curing of precast elements to attain approximately 70% of their 28-day strength under controlled curing conditions.

3.1.3 Optimizing Objectives

Besides the various constraints discussed above, rescheduling for precast production must also take into account several different objectives. Apart from the overall objective of profit, the precaster and the contractor may have other specific objectives associated with production, delivery, storage, resources, and quality control, as shown in Table 3.1.

Table 3.1 Multiple optimizing objectives for precast production rescheduling

Category	Precast Factory	Construction Site
<i>Production/Construction</i>	<ul style="list-style-type: none"> ● Minimize production time ● Avoid interruption in the production process 	<ul style="list-style-type: none"> ● Timely installation of precast components
<i>Delivery</i>	<ul style="list-style-type: none"> ● Timely and correct delivery of components to the site 	<ul style="list-style-type: none"> ● Timely and correct delivery of components from the factory ● Avoid site congestion
<i>Storage</i>	<ul style="list-style-type: none"> ● Minimize inventory in the yard 	<ul style="list-style-type: none"> ● Minimize inventory on-site
<i>Resource</i>	<ul style="list-style-type: none"> ● Maximize utilization of resources, such as molds, workers, cranes, <i>etc.</i> ● Minimize resource shift, such as mold changeovers, personnel shift, <i>etc.</i> 	<ul style="list-style-type: none"> ● Maximize utilization of resources, such as crane, workers, <i>etc.</i>
<i>Quality Control</i>	<ul style="list-style-type: none"> ● Maximize production quality ● Minimize rework 	<ul style="list-style-type: none"> ● Maximize precast products quality ● Maximize quality of installation

However, some of these objectives conflict between one and another. For example, from the viewpoint of the precaster, long runs of a particular element type keep the number of mold changeovers down and improve mold utilization rates. Unfortunately, it may increase the inventory of this element type and result in missing due dates for other elements. Emphasis on meeting due dates alone may result in many mold changeovers and drive up related costs, which can be considerable if the cost of lost production time is included. From the viewpoint of the contractor, storage of critical precast elements on-site can safeguard against the failure of the precaster in providing

timely deliveries, and ensure smooth and continuous hoisting operations. However, storing on the construction site increases the risk of damage to the precast components, and may not be practical on congested sites. Moreover, the preferences of the precaster and the contractor for these objectives can be different under different circumstances. For example, the contractor may care more about timely delivery in order to ensure timely installation of these components on-site and avoid site congestion. On the other hand, the precaster may want the best utilization of manufacturing facilities among different concurrent projects. This research recognizes that different objectives may exist in the rescheduling process, and the precast production rescheduling problem is indeed a multiobjective optimization problem.

3.1.4 Mathematical Formulation

The mathematical formulation of the precast production rescheduling problem follows that developed by Chan and Hu (2002b) for the precast production scheduling problem as the same underlying production process is involved. This mathematical formulation represents the precast scheduling/rescheduling problem for the comprehensive precast production and incorporates the different constraints encountered in practice. The essential difference with Chan and Hu (2002b) lies in the new search procedures and search representations adopted when rescheduling is attempted.

3.1.4.1 Decision variables and parameters

In general, the production schedule of precast building components can be thought of as a specification of which element is to be produced using a particular mold on a specific day. Let $m = 1, 2, \dots, M$ denote the serial number of molds, $t = 1, 2, \dots, T$ denote the workdays in the planning horizon, and $e = 1, 2, \dots, E$ denote the element

types to be produced. With this notation, the production schedule can be presented as shown in Table 3.2. In order to facilitate mathematical formulation and calculation, a set of classical binary valued variables $x_{t,m,e}$ with a domain of $\{0, 1\}$ is used to represent the schedule. Therefore, $x_{t,m,e} = 1$ means that mold m is assigned to produce element type e on day t , whilst $x_{t,m,e} = 0$ represents the opposite. The parameters that are used in defining the constraints and objectives of the rescheduling problem are defined in Table 3.3. The values of these parameters are set up for each precast production rescheduling problem.

Table 3.2 Production schedule representation

Molds(m)	Days (t)				
	1	2	...	$T-1$	T
1	1	1	...	1	1
2	2	2	...	1	1
...
M	E-1	E-1	...	E	E

Table 3.3 Parameters considered for the rescheduling problem

Parameters	Description
T	Length of the scheduling period in days, $t=0, 1, 2, \dots, T$;
M	Maximal number of molds included in the schedule, $m=1, 2, \dots, M$;
E	Number of element types to be produced in the scheduling period, $e=1, 2, \dots, E$;
$S_{0,e}$	Initial stock of element type e at the beginning of the scheduling period;
$S_{t,e}$	Number of element type e in stock on day t ;
S_e	Maximum allowable storage level of element type e ;
S_e'	Minimum buffer storage level of element type e ;
$R_{t,e}$	Number of element type e required to be delivered on day t ;
$D_{t,e}$	Number of element type e delivered on day t ;
T_e'	Delivery day for element type e ;
TN	Normal production days; obtained by subtracting non-working days from T ;
\bar{L}_e	Minimum required lead time between production and delivery for element type e ;
L_e	Lead time between production and delivery of element type e ;

3.1.4.2 Constraints

The constraints considered in this research are formulated as follows:

(1) Mold capacity constraint: The elements that can be produced by each mold are restricted to those that belong to the same element group with only minor variations among different types within the group. Each mold is assumed to produce at most one element per day in the comprehensive mode of precast production (Equation 3.1). The maximum production capacity per day can not exceed the total number of molds (Equation 3.2).

$$\sum_{e=1}^E x_{t,m,e} \leq 1 \quad \text{all } m, t \quad (3.1)$$

$$\sum_{m=1}^M \sum_{e=1}^E x_{t,m,e} \leq M \quad \text{all } t \quad (3.2)$$

(2) Minimum lead time constraint: A minimum lead time for each element type is assumed in order to attain required concrete strength before delivering to the construction site. The length of lead time can be varied with different element types under different curing conditions.

$$L_e \geq \bar{L}_e \quad \text{all } e \quad (3.3)$$

(3) Non-production constraint: It is assumed that there is no production on Sundays and public holidays, while lead times can span Sundays and public holidays.

$$x_{t,m,e} = 0 \quad t \in \text{sundays and public holidays} \quad (3.4)$$

(4) Delivery constraint: It specifies requirements of delivering precast elements to the construction site in a given day. Both the required quantity and due date should be met, otherwise a penalty would be incurred (Equation 3.5). Hence, the sum of the initial stock and the total production of any element before each delivery date should meet

the number of elements required to be delivered (Equation 3.6).

$$D_{t,e} \leq R_{t,e} \quad \text{all } t, e \quad (3.5)$$

$$S_{0,e} + \sum_{t=1}^{T'_e} \sum_{m=1}^M x_{t,m,e} \geq \sum_{t=1}^{T'_e} R_{t,e} \quad \text{all } e \quad (3.6)$$

(5) Inventory constraint: The stock levels for each element type are assumed to be kept within a band between the minimum buffer required and the maximum inventory limit:

$$S'_e \leq S_{0,e} + \sum_{t=1}^T \sum_{m=1}^M x_{t,m,e} - \sum_{t=1}^T D_{t,e} \leq S_e \quad \text{all } e \quad (3.7)$$

Depending on whether the requirement that the constraint be satisfied is absolute or not, these constraints are categorized as either hard or soft in this research. Hard constraints, which must always be satisfied, include the mold capacity constraint, the minimum lead time constraint and the non-production constraint. On the other hand, the delivery constraint and the inventory constraint are soft constraints, which can be relaxed when necessary but would incur a penalty if they are not met.

3.1.4.3 Objectives

Based on different circumstances in the rescheduling process for precast production, the precaster and the contractor will have different objectives and preferences towards these objectives in problem solving. Here, two objectives are highlighted in this research.

(1) Minimizing late delivery, i.e. minimizing the sum of element units that are delivered beyond their required due date:

$$F_D = \sum_{t=1}^T \sum_{e=1}^E (R_{t,e} - D_{t,e})^+ \quad (3.8)$$

$$\text{where } (R_{t,e} - D_{t,e})^+ = \max(0, R_{t,e} - D_{t,e})$$

(2) Minimizing the sum of ‘out-of-band’ element units to maintain inventory levels within a prescribed band. Tight space constraints both on-site and in the factory, severely limit the number of units that can be produced ahead of the schedule. On the other hand, the precaster needs to keep production somewhat ahead of the promised delivery schedule as a buffer against the changes of site conditions.

$$F_I = \sum_{t=1}^T \sum_{e=1}^E \{(S_{t,e} - S_e)^+ + (S_e' - S_{t,e})^+\} \quad (3.9)$$

where $(S_{t,e} - S_e)^+ = \max(0, S_{t,e} - S_e)$, $(S_e' - S_{t,e})^+ = \max(0, S_e' - S_{t,e})$

These two objectives are highlighted for the precast production rescheduling problem as they are very commonly considered in the rescheduling process and they obviously conflict. The first objective of minimizing late deliveries reflects the interest of the contractor, whilst the second objective of minimizing the sum of ‘out-of-band’ element units for inventory levels reflects the concern of the precaster.

3.2 Coordinated Production Reactive Scheduling Model

A coordinated production reactive scheduling model (CPRSM) is proposed in this research. The CPRSM is developed to support production rescheduling in precast factories and facilitate schedule coordination between the precaster and the contractor.

3.2.1 Model Overview

The CPRSM includes the following four modules for precast production rescheduling, namely disturbance detection, global search with multiobjective optimization, local search with specific constraints, and ranking of outcomes for negotiation, as shown in Figure 3.3. With these four sub-processes, rescheduling for precast production begins

with an occurrence of new schedule disturbances. Details for each module are discussed in the following sections.

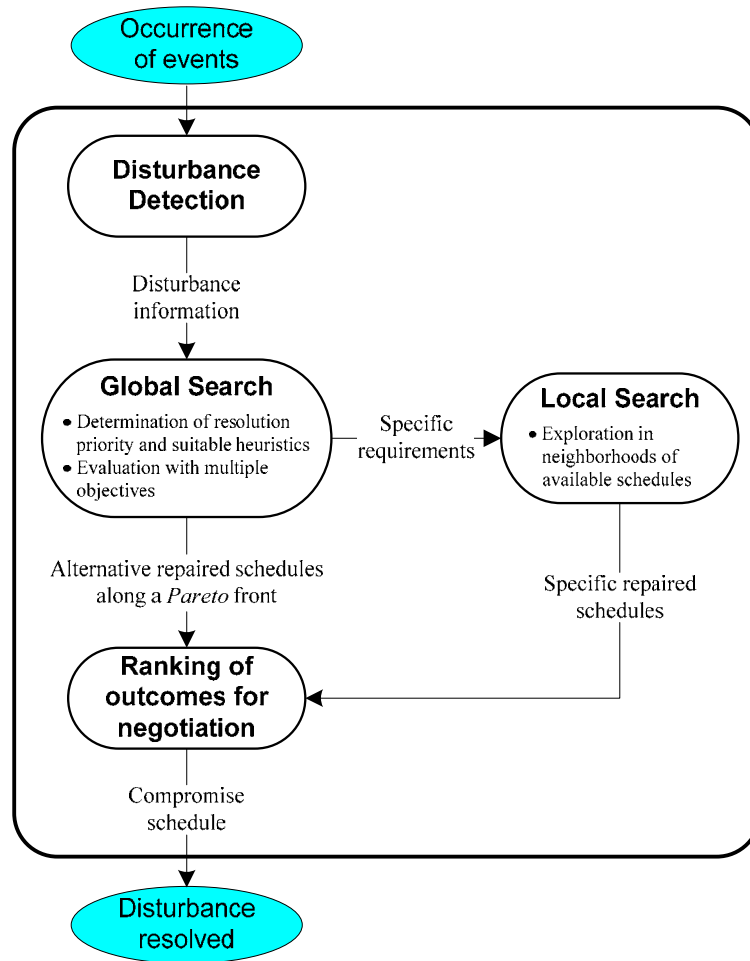


Figure 3.3 Framework of the coordinated production reactive scheduling model

3.2.2 Disturbance Detection

In the proposed CPRSM, the module for disturbance detection is first evoked by events that will cause schedule disturbances to the existing production schedule. Detailed disturbance information need to be collected, which includes the identity, quantity, due date and time of scheduled production of the precast components involved in disturbances. Schedule disturbances are caused by changes in delivery quantities, delivery due dates, element specifications, or number of workdays

available. Each type of schedule disturbance is represented differently. In order to better illustrate this, a 10-day production schedule for a single mold is presented. The schedule arrays before and after disturbance detection are shown in Figure 3.4. The disturbance information is represented in Table 3.4.

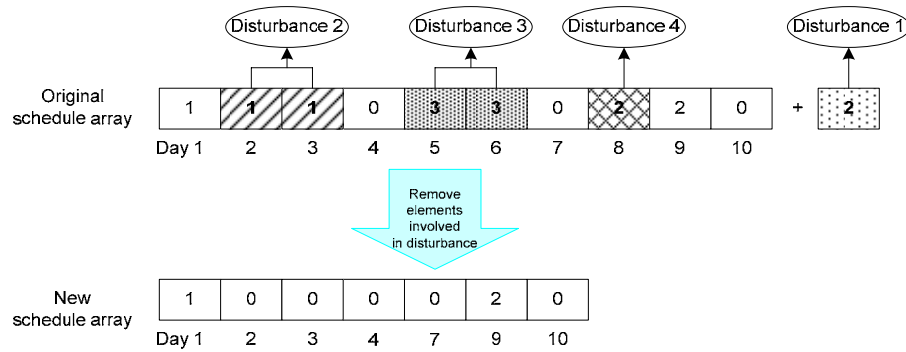


Figure 3.4 Illustration of schedule array and disturbances

Table 3.4 Characteristics representation of schedule disturbances

Disturbance No.	Element type	Quantity	Scheduled day	Original due date	New due date	Type
1	E2	1	--	--	Day 2	Change in quantity due to poor quality
2	E1	2	Day 2 to 3	Day 8	Day 5	Change in delivery due date due to progress on-site
3	E3	2	Day 5 to 6	Day 9	Day 9	Change in design specification
4	E2	1	Day 8	Day 10	Day 10	Change in workday due to crane breakdown

3.2.2.1 Change in quantities

A change in production quantity of any element could be either an increase or a decrease of the demand required. However, it is assumed that the need to increase production is of more concern to the precaster and triggers a necessity for rescheduling. The need to increase production for certain elements usually results from causes, such as replacing elements rejected due to quality defects, amendments

to the design by the architect or the structural engineer, or as a result of new orders. Representing this kind of disturbances is straightforward. An example of such a disturbance is “Disturbance 1” in Figure 3.4, which is not scheduled in the original schedule array. In this instance, one piece of element E2 has to be produced to compensate for a rejected piece due to a quality problem and its due date for delivery is day 2, as indicated in Table 3.4.

3.2.2.2 Change in due dates

The precast production schedule is closely related to the pace of construction on-site. Any change in site progress is likely to cause a schedule disturbance by changing the due dates of delivery to the construction site. On the one hand, the due date could be advanced because the construction progress is better than anticipated in the project schedule. On the other hand, the due date could also be delayed due to stoppage or delay of work on-site caused by unanticipated events. The disturbance is represented by noting the affected elements and the new due date. As shown in Figure 3.4 and Table 3.4, the second disturbance involves two pieces of element E1 that were originally planned to be produced on days 2 and 3. The due date for these two elements is changed from day 8 to day 5 since progress on-site is ahead of the original project schedule. The productions of these two elements are then removed from the original schedule array. It makes the relevant production space available in the new schedule array.

3.2.2.3 Change in element specification

This kind of disturbance is often caused by design changes as the construction project progresses. The precaster has to take some time to modify the mold to suit the new design. Consequently, the mold can not be used to cast new elements and its

production capacity is lost during this modification period. This, in turn, makes the number of available workdays within the planning horizon decrease. The schedule disturbance indicates the precast elements that are affected during the period of mold modification. Their delivery dates are kept unchanged. For example, disturbance No.3 in Table 3.4 shows that the mold requires modification for two days from day 5 to day 6. As shown in Figure 3.4, two pieces of element E3 that were originally scheduled to be produced during these two days have to be rescheduled, and the due date is kept as day 9.

3.2.2.4 Change in workdays

These disturbances occur as a result of a change in the availability of resources in the precast factory, and include the breakdown of the crane, problem with cleanliness of the molds, and shortage of skilled workers or raw materials. Troubleshooting and solving such problems need time, resulting in a loss of workdays. This kind of disturbance is represented in the same way as a disturbance due to a change in specification. The disturbance information identifies the elements produced in the affected workdays with their original due dates. The number of available workdays within the planning horizon is also decreased. The fourth disturbance shown in Figure 3.4 and Table 3.4 involves a crane breakdown in day 8, which causes the production of E2 to be rescheduled with the original due date on day 10. Consequently, the schedule array is reduced by 3 days corresponding to days 5-6 involved in Disturbance 3 and day 8 involved in Disturbance 4.

3.2.3 Global Search with Multiobjective Optimization

Following disturbance detection, the global search module in the CPRSM starts to perform rescheduling considering the different constraints and optimization objectives.

The functions of the global search module include (1) determination of the resolution priority for schedule disturbances along with the corresponding heuristics, and (2) generation of alternative repaired schedules that are non-dominated with respect to the quantitative evaluation criteria employed.

3.2.3.1 Rescheduling heuristics for precast operations

Precast elements involved in the schedule disturbances are first removed from the existing schedule and put into a list to be rescheduled. Rescheduling is performed with the use of heuristics in actual practice. However, repair actions are likely to cause further disturbances if these disturbances are solved with the wrong heuristics or in the wrong sequence. Therefore, it is necessary to consider not only *how* to resolve the disturbances but also the *order* in which the disturbances are to be resolved as both of them have a bearing on the desirability of the final repaired schedule.

There are several heuristics used by the precaster to repair the production schedule. This research considers some basic heuristics and hybrid strategies resulting from the combination of the basic heuristics. Figure 3.5 illustrates the operations of these heuristics.

(1) Right Shifting (RS): This heuristic strategy resolves disturbances by simply “pushing” the existing production forward in time until the disturbance is resolved. It is frequently employed in situations when a resource become temporarily unavailable, such as when the crane breaks down or the mold has to be cleaned.

(2) Left Shifting (LS): This is a strategy similar to RS, and it shifts an operation backwards in time. It is particularly useful when a hard constraint that previously prohibits the commencement of operation is softened or removed. However, there is the problem of high inventory levels if this heuristic is applied.

(3) Opportunistic Insertion (OI): This strategy makes use of idle days in the schedule to accommodate a disturbance, possibly including breaking it into smaller parts and fitting these smaller parts into the schedule in an opportunistic first fit manner. The efficiency of this heuristic rule largely depends on the initial utilization level of the production facilities.

(4) Deterministic Insertion (DI): This strategy is similar to opportunistic insertion but the disturbances have priority over scheduled production and displace them from the schedule. The affected scheduled elements are rescheduled using OI.

(5) As-soon-as-possible (ASAP)/Backward Scheduling (BS): The earliest time and the latest time that an operation could be scheduled to start are two important concepts in scheduling problems. Based on these two concepts, two methods of scheduling can be distinguished, namely ASAP and BS. The ASAP method reschedules the affected operations based on the earliest start time (EST). It would improve the utilization of resource and prevent late deliveries. On the other hand, using the BS method causes the operation to start at the latest start time (LST). This method could lower the inventory level since the elements are manufactured as close to the delivery date as possible.

(6) Sub-contracting: Outsourcing production to other operators is another option to resolve disturbances, especially when the precast factory is already producing at its peak capacity or it is more economically beneficial to do so

In precast production, several molds are assigned to produce the same group of elements with minimum changeover. Therefore, using the above heuristics, the disturbances are resolved by assigning elements to any one of the molds within the same group that are capable of producing the elements.

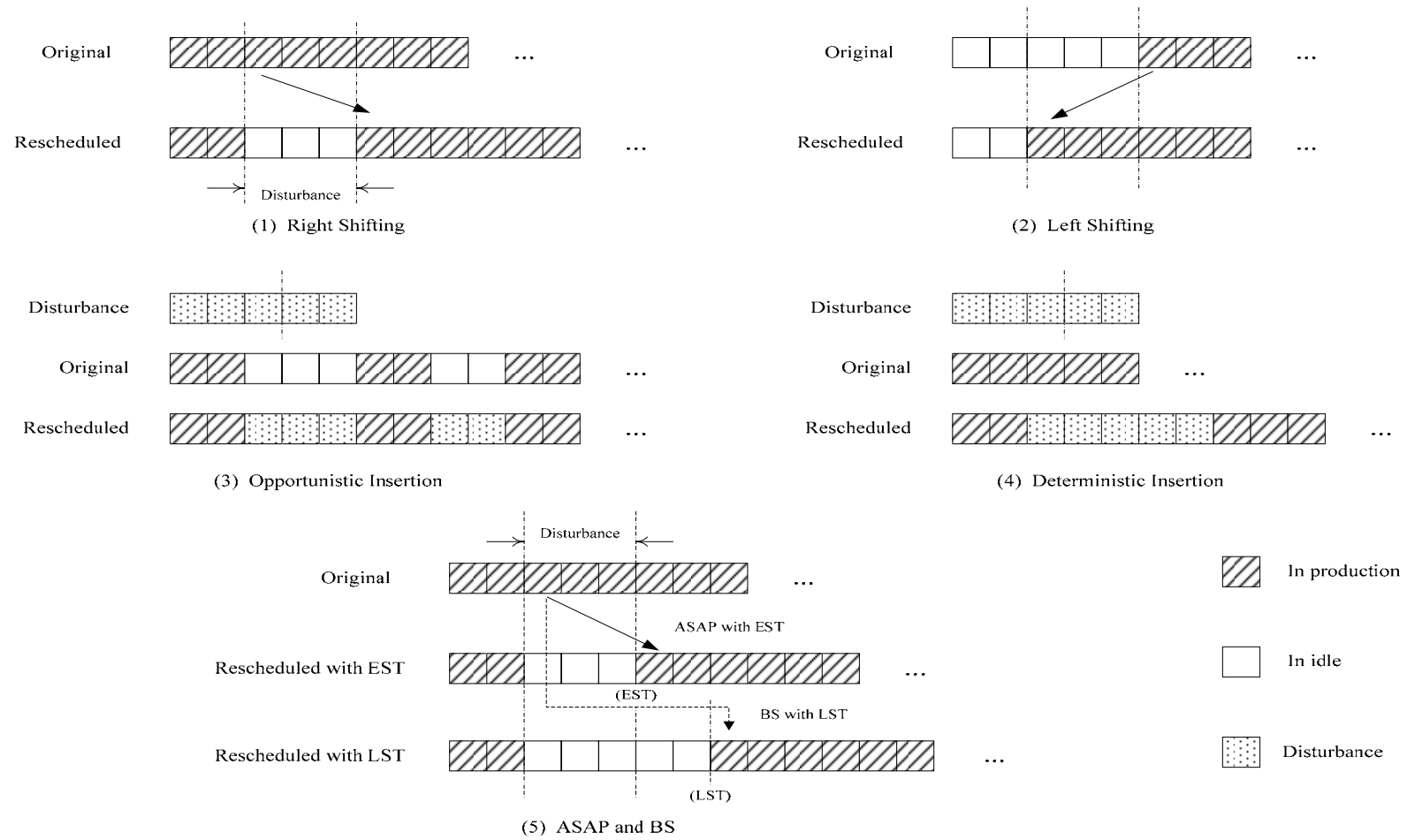


Figure 3.5 Illustration of heuristic strategies for schedule repair

3.2.3.2 Evaluation and generation of alternatives

Rather than relying solely on the subjective preferences of the decision makers, the desirability of schedule repair outcomes is evaluated using quantifiable objectives that are meaningful to the interests of the precaster and the contractor. The global search module focuses on generating alternative repaired schedules that simultaneously optimize the two objectives of minimizing late deliveries and maintaining inventory levels formulated in Equations (3.8) and (3.9).

The solving procedure for multiobjective optimization problems is not straightforward. Unlike the case of problems optimized with a single objective, this class of problems usually has no unique, optimal solution. Instead, there is a set of non-dominated solutions known as the *Pareto* optimal set for the multiobjective optimization problem. According to the multiple criteria considered, these solutions are not better than their peers as an increase in any one of these objectives may decrease the other objectives and vice versa. This set of non-dominated solutions describes trade-offs available for the objectives considered.

In solving a multiobjective optimization problem, two conceptually distinct types of problem difficulty can be identified in the literature: searching and decision making. The first aspect refers to the optimization process in which the feasible set is sampled for *Pareto* optimal solutions. The second aspect addresses the problem of selecting a suitable compromise solution from this set of *Pareto* optimal solutions. Based on how the searching and decision making processes are combined, the different ways of solving multiobjective optimization problems can be classified into three categories (Hwang and Masud 1979; Horn 1997; Fonseca and Fleming 1998; and Van Veldhuizen and Lamont 2000):

- (1) A *prior* articulation of preferences: The decision maker expresses preference information prior to optimization in terms of a unique aggregating function, which combines individual objectives into a single utility value and makes the problem into single objective one.
- (2) A *posterior* articulation of preferences: Searching is performed without any preference information given. Later the decision maker is presented with a set of candidate non-dominated solutions and chooses the compromise solution from this set depending on his preferences.
- (3) Progressive/interactive articulation of preferences: Decision making and searching occur in interleaved steps. At each step, the decision maker supplies partial preference information to guide the search, which in turn generates alternatives according to the information received.

If the preference factor among the objectives is known for a specific problem, the *prior* preference articulation to the multiobjective optimization procedure is adopted. It is only required to form a composite objective function such as a weighted sum of objectives, where a weight for an objective is proportional to the preference factor assigned to this particular objective. Then, a search is conducted for one particular trade-off solution for this specific preference combination. However, such preference information may not be easily available as preferences may involve other considerations that are often non-technical, qualitative and experience-driven. This requires an analysis of all the higher level information of the problem. It would be even more difficult in the absence of any knowledge of the problem. Unless a reliable and accurate preference vector is available, the optimal solution obtained by the *prior* preference articulation is highly subjective and dependent on a particular decision maker. By contrast, if a set of trade-off solutions is available, the decision maker can

evaluate the pros and cons of each of these solutions based on all such non-technical and qualitative considerations, and compare them to make a choice with the *posterior* preference articulation.

In view of the discussion above, this research adopts the procedure with *posterior* preference articulation. Since the precaster and the contractor are usually not sure of the exact trade-off relationship between the conflicting objectives for the rescheduling problem, it is better to make the effort to find the set of trade-off solutions by considering all objectives. With reference to the solving procedure depicted in Figure 3.6, the global search module is employed to search for multiple repaired schedules; these represent different trade-offs between the two objectives considered. After a well-distributed set of trade-off solutions is found, the precaster and the contractor can use higher-level information to choose one of these solutions.

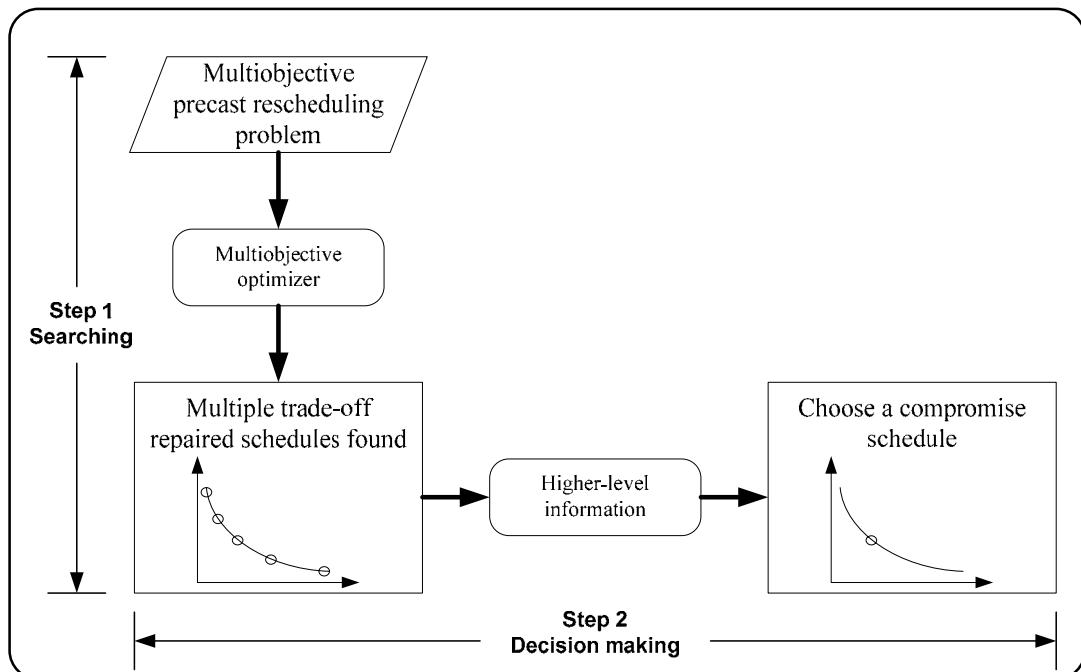


Figure 3.6 Schematic of solving procedure for multiobjective optimization

3.2.4 Local Search with Specific Constraints

Further alternative schedules are needed if the precaster and the contractor want to make minimal adjustments to the repaired schedules provided by the global search module. Furthermore, no algorithm is guaranteed to find all non-dominated solutions along the *Pareto* front. This implies a requirement to conduct incremental exploration of the search space in particular areas identified by the decision makers. This capability can enhance the usefulness of multiobjective optimization and is required in actual applications. A local search module is included in the CPRSM to explore the neighborhoods of available repaired schedules and obtain schedules that meet specific constraints imposed by either the precaster or the contractor.

3.2.4.1 Specific constraints and error functions

In the global search module discussed above, alternative repaired schedules are evaluated with two conflicting objectives, namely F_D and F_I , in terms of all element types within the production schedule. The precaster or the contractor may impose specific requirements on repaired schedules after reviewing these alternatives. Based on the *Pareto* front of the two objectives for all element types, the specific requirements considered in this research can be a setting of target values or value ranges on these two objectives for a certain element type. These requirements are treated as constraints with their degrees of satisfaction encoded as error functions. For example, if the target of total unit number of element E1 for late delivery is set as A and the number for the current schedule is B , the error function of this constraint is represented by $\max(0, |A - B|)$, whereby 0 signifies full satisfaction of the constraint, and other non-negative values reflect the degree to which the constraint is violated. Since these specific constraints are all measured in terms of the number of element units, all of their corresponding error functions can be aggregated directly into an

objective function for the local search module. The quality of a schedule is therefore explicitly measured by the objective function as its value represents the total degree of constraint violation for the schedule with respect to all the specific requirements.

3.2.4.2 Iterative repair process

Iterative repair techniques perform a kind of local search to improve a given schedule that is possibly flawed. This can easily be implemented as a single, general purpose technique applicable to both predictive and reactive scheduling (Kjenstad 1998).

Iterative repair techniques also have other advantages over constructive techniques in the incremental exploration considered in the local search module. Firstly, the rescheduling problem may become overconstrained with specific constraints, but this may not be known in advance by the decision maker. When the problem is over-constrained, a constructive method must exhaust all possibilities before it can infer that constraints must be relaxed. In contrast, the repair-based methods attempt to iteratively improve solutions regardless of whether the problem is over-constrained or not and terminate with a set of assignments that is as close to a solution as could be derived in the time allotted (Zweben *et al.* 1994). Secondly, the “global” constraints and optimization criteria can be easily evaluated since the repair methods search through a space of complete schedules, unlike the situation in constructive methods where the global criteria can only be approximately evaluated with a partial schedule.

For the iterative repair procedure adopted in this research, a schedule obtained from the global search module is selected as the initial solution, and then changed by simple inexpensive modifications. The efficacy of these changes is evaluated by the objective function of the sum of deviations from the specific constraints imposed. If the new schedule is an improvement, it is used in the next iteration, and if it is better

than any previous solution, it is stored as the best solution so far. The search may terminate when an acceptable schedule is found, when a predefined amount of search is completed or by user interruption. If the search starting with a schedule fails to find the specific schedule, the iterative repair process could run a number of times with other starting solutions. Figure 3.7 provides a general framework for the iterative repair process developed in the local search module.

3.2.5 Ranking of Outcomes for Negotiation

The alternative repaired schedules generated by both the global search and the local search modules provide valuable options for the precaster and the contractor to consider. They can express their preference for these repaired schedules and reach a compromise solution. This is not discussed in detail since the focus of this research is on the search for alternative repaired schedules.

With these four sub-processes, the CPRSM is proposed and developed as an integrated system to handle schedule disturbances in precast production and facilitate schedule coordination between the precaster and the contractor. Playing an important role in searching alternative repaired schedules, the global search module and the local search module are elaborated respectively in the following chapters.

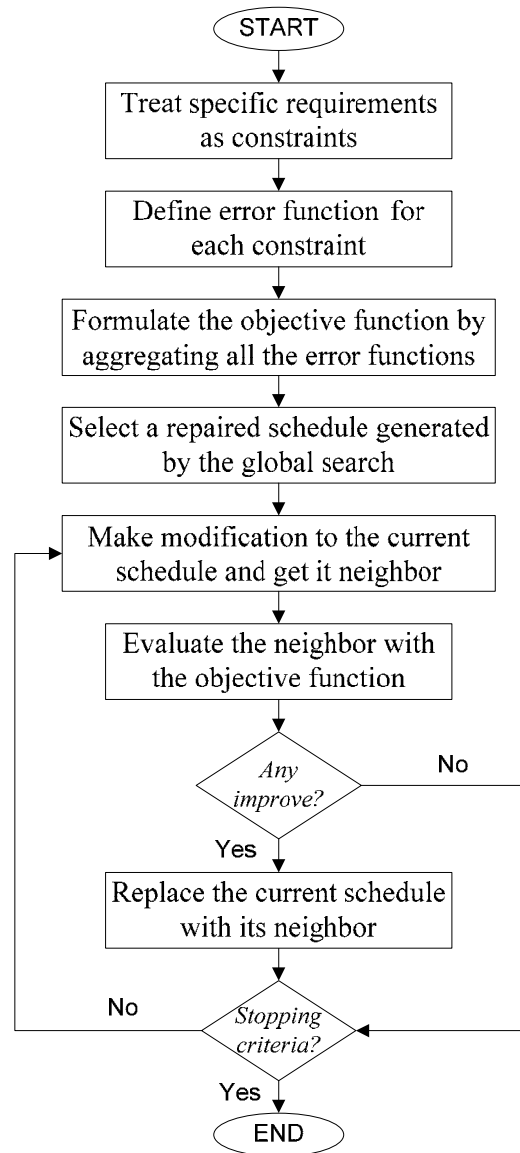


Figure 3.7 Iterative repair process with specific constraints

CHAPTER 4

MULTIOBJECTIVE GENETIC ALGORITHMS FOR GLOBAL SEARCH

The core of the proposed coordinated production reactive scheduling model is the global search module, which focuses on generating alternative repaired schedules considering multiple conflicting objectives. The implementation of multiobjective genetic algorithms in global search is described in this chapter, which includes chromosome representation, genetic operators, fitness assignment and sharing, and the elitist strategy employed.

4.1 Basic Mechanisms of Genetic Algorithms

As the name suggests, the genetic algorithm (GA) is a search algorithm based on the mechanics of natural selection and natural genetics. It combines survival of the fittest among string structures with a structured yet randomized information exchange to form search algorithms with some of the innovative flair of human search (Goldberg 1989).

The most important construct in the GA is the *chromosome* (Figure 4.1), which is essentially a candidate solution to the problem. The chromosome is made up of blocks of cells called *genes*. Each gene encodes a particular character of the candidate solution with a possible value termed as an *allele*. The site indicating the position of a gene held in the chromosome is called its *locus*.

In typical GAs, a group of random chromosomes forms a *population* of initial

Gene Position (Locus)	1	2	3	4	5	6	7	8	9	10
Gene Value (Allele)	1	1	0	1	0	1	0	0	1	0

Figure 4.1 GA chromosome

candidate solutions. These chromosomes reproduce through consecutive iterations, called *generations*. In each generation, the quality of each chromosome is evaluated based on how well it satisfies a predefined *objective function*. A *fitness* is then assigned to each chromosome with respect to its objective function value, which represents how “fit” it is in relation to other chromosomes in the population. Within this population, fitter chromosomes will have higher chances of being selected to participate in the generation of new solutions. After a selection procedure for *parent* chromosomes, new *offspring* are generated from these parents using genetic operators, such as crossover and mutation. As populations move from one generation to another, it is hoped that better and better solutions will evolve until the cycle stops on reaching a stopping criterion. Following the crucial steps of deciding upon a suitable chromosomal representation for the problem and determining the objective function for fitness evaluation, the GA optimization process is carried out as follows (Davis 1991):

- (1) Initialize a pool of solutions known as the parent pool;
- (2) Determine the fitness of each solution in the parent pool with respect to the problem specific objective function;
- (3) Select parent solutions for the creation of the next generation with a probability relative to their fitness;
- (4) Create offspring solutions by means of genetic operators on the selected parent

solutions;

- (5) Form a new parent pool for the new generation, namely through population replacement;
- (6) Check whether the stopping criteria are met or not. If not, go back to step (2); otherwise, stop the search and return the best solution.

There are three main genetic operators in genetic algorithms, namely selection, crossover and mutation, which are usually used to create the next generation. These operators provide means of weeding out poor solutions and generating better solutions through the subsequent iterations. Furthermore, these simple operators make GAs computationally simple to implement; in spite of their apparent simplicity, GAs have proven to be effective in tackling a variety of optimization problems.

Selection is usually the first operator that is applied to an existing population to create offspring. Being an artificial version of natural selection, the essential idea is to select “better-than-average” solutions from the existing population and insert multiple copies of these solutions in the mating pool in a probabilistic manner. The selection operator may be implemented in a number of ways. The most common way is to create a biased roulette wheel where each individual in the current population has a slot on the roulette wheel sized in proportion to its fitness value. Thus, those individuals with a greater fitness value are expected to receive more chances of contributing their genes to the population in the new generation than those with a lesser fitness value.

The power of GA derives from the crossover operation, where a randomized exchange of genetic material is executed with a possibility that “good” solutions can generate “better” ones (Goldberg 1989). Crossover takes building blocks from two individuals

and combines them into new offspring. It is hoped that by doing so, new individuals will be created whose fitness exceeds that of either parent. A simple single-point crossover may proceed in the following steps as depicted in Figure 4.2. Firstly, members of newly reproduced strings in the mating pool are picked at random. Secondly, each pair of strings exchanges part of its chromosome across a point chosen uniformly at random along the length of the chromosome string. The mechanics of crossover are simple; nonetheless, the combined effect of selection and the structured, though randomized, information exchange through crossover gives GAs much of their power (Man *et al.* 1999).

Mutation is another process essential for evolution. It operates on a single chromosome and produces a new genotype by making a random change to values of one or more genes in the chromosome string. When used sparingly with selection and crossover, mutation is an insurance policy against premature loss of important options. Furthermore, it helps push the search effort into different search spaces by introducing new allele values into the string structure, thus creating new possibilities that might not have been present in the initial pool of solutions. However, in order to avoid the disruption of good solutions, the probability of mutation is often kept very low. Figure 4.2 illustrates the mutation process.

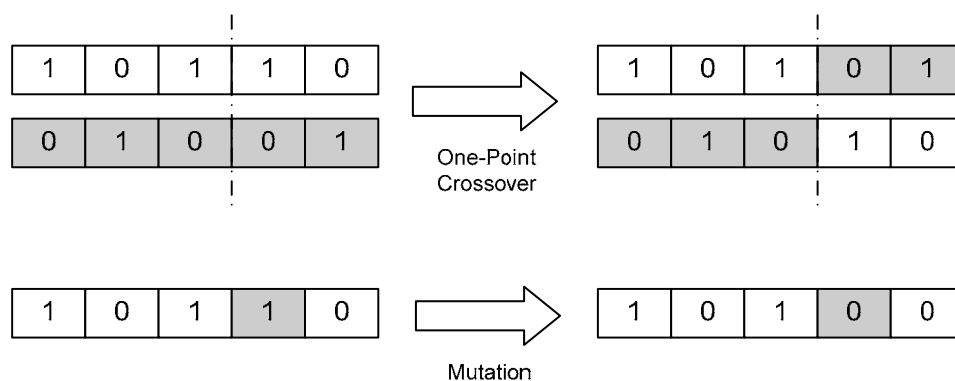


Figure 4.2 Illustrations of crossover and mutation

4.2 Genetic Algorithms for Multiobjective Optimization

A multiobjective genetic algorithm employs the same operations as the conventional single-objective genetic algorithm, but is required to evolve a set of solutions that is *Pareto* optimal. There are several ways for a multiobjective genetic algorithm to search for *Pareto* optimal solutions. A brief summary of the key issues in multiobjective search by GAs is given first, followed by a detailed introduction of the main features and computational procedures of several multiobjective genetic algorithms applied in this research.

4.2.1 Key Issues in Multiobjective Search

In a single objective optimization problem, the superiority of one solution to another can be easily determined by comparing their objective function values. There exists a single identifiable value that is superior to all other objective function values. This is not the case for a multiobjective optimization problem, since it is not possible to optimize several objectives simultaneously at times when there are conflicting objectives involved. Any two solutions $x^{(1)}$ and $x^{(2)}$ for such problem can have one of two possibilities, which are one dominating the other or neither dominating the other. A solution $x^{(1)}$ is said to dominate the other solution $x^{(2)}$ if both the following conditions are true (Deb 2001): (i) The solution $x^{(1)}$ is no worse than $x^{(2)}$ in all objectives; and (ii) The solution $x^{(1)}$ is strictly better than $x^{(2)}$ in at least one objective. If there exists no feasible solution $x^{(2)}$ dominates $x^{(1)}$, then $x^{(1)}$ is said to be a non-dominated or *Pareto* optimal solution. Therefore, based on the concept of *Pareto* dominance, a multiobjective optimization problem has a family of non-dominated solutions known as the *Pareto* optimal set, in which none of these solutions is better than others for all the objectives considered.

Genetic algorithms appear to be well suited for multiobjective optimization problems due to the ability of finding multiple solutions simultaneously in each run (Carlyle *et al.* 2001). Therefore, research on multiobjective genetic algorithms have focused on the ability to emphasize all non-dominated solutions equally and preserving a diverse set of these solutions in the population at each generation. It may lead the population to converge and form a *Pareto* front with a good spread after some generations. Modifications, such as *Pareto*-based fitness assignment and sharing, have been developed to accomplish this based on the evaluation of fitness of each solution.

In contrast to single objective optimization, where the objective function and the fitness function are often identical, multiobjective optimization problems require that both fitness assignment and selection must allow for the existence of several objectives. Several approaches have been developed, these include:

- (1) Selection by switching objectives (Schaffer 1984 and 1985; Kursawe 1991), i.e., an individual is selected based on a different objective, such as one of the multiple objectives and a specific order of these objectives;
- (2) Selection with parameter variation (Hajela and Lin 1992; Ishibuchi and Murata 1996), i.e., an individual is selected based on a single objective function by aggregating the multiple objectives with systematically varied parameters in the same run;
- (3) *Pareto*-based selection (Goldberg 1989; Fonseca and Fleming 1993; Horn *et al.* 1994; Srinivas and Deb 1994), i.e., an individual is selected based on the concept of *Pareto* dominance.

The first two classes of selection techniques may have a bias towards “extreme” solutions and be sensitive to non-convex *Pareto* optimal front (Horn 1997, Van Veldhuizen 1999). In contrast, the class of *Pareto*-based techniques is capable of

finding any *Pareto* optimal solution as these techniques use the concept of *Pareto* dominance to calculate an individual's fitness relative to the whole population. Therefore, *Pareto*-based techniques seem to be the most popular in the field of evolutionary multiobjective optimization (Van Veldhuizen and Lamont 1998).

Although all non-dominated individuals in the population are emphasized simultaneously with *Pareto*-based techniques, it does not guarantee that the *Pareto* optimal set can be uniformly sampled because of the phenomenon known as genetic drift (De Jong 1975); i.e., when presented with multiple equivalent optima, finite populations tend to converge to only one of these. This is attributed to stochastic errors in the selection process. Fitness sharing, proposed by Goldberg and Richardson (1987), is the most frequently used technique to maintain a diverse population. It has been applied in many research works, e.g. Hajela and Lin (1992); Fonseca and Fleming (1993); Horn *et al.* (1994); Srinivas and Deb (1994); Todd and Sen (1997); and Zydallis *et al.* (2001). Fitness sharing is based on the idea that individuals in a particular *niche* have to share available resources, and aims to promote the formation and maintenance of stable niches (Zitzler 1999). In this method, the population is divided into different niches according to the similarity of individuals either in *phenotype* (the decoded parameter space or objective space) or in *genotype* (the gene space). Through degrading fitness values of similar solutions, the use of fitness sharing helps mitigate unbridled head-to-head competition between widely disparate points in the search space.

4.2.2 Weighted Sum Genetic Algorithm

The weighted sum genetic algorithm is representative of the aggregation approach to solve multiobjective optimization problems. It combines all the objective functions

into a single one using different weighting coefficients (weights) for each component objective function. It means that a multiobjective optimization problem is transformed into a scalar optimization problem. The weight is usually taken to represent the relative importance of the individual objectives. Each combination of weights yields a traditional simple genetic algorithm (SGA), which solves a single objective problem with the basic procedure described earlier. Since the results of solving an optimization problem using a weighted sum objective function can vary significantly as weights change, and very little is usually known about how to choose the weights for a particular problem in a *priori* manner, it is necessary to solve the same problem with many different weight combinations. Therefore, it can be used to find multiple non-dominated solutions by repeating the search procedure with varying weights. This combination technique has been widely applied in different fields of research probably due to its simplicity although it has its shortcomings.

4.2.3 Non-dominated Sorting Genetic Algorithm (NSGA)

Non-dominated sorting genetic algorithm (NSGA) (Srinivas and Deb 1994) is representative of the class of *Pareto*-based approaches. The idea behind NSGA is that a fitness assignment scheme is used to emphasize non-dominated solutions and a sharing strategy is used to maintain diversity in the population. Figure 4.3 shows a flowchart depicting the NSGA. The efficiency of NSGA lies in the way that multiple objectives are reduced to a dummy fitness function using a non-dominated sorting procedure. With this approach, any number of objectives can be solved, and both maximization and minimization problems can be handled (Coello 2000). Moreover, sharing in the decision parameter value space ensures a better distribution of individuals and allows multiple equivalent solutions to exist.

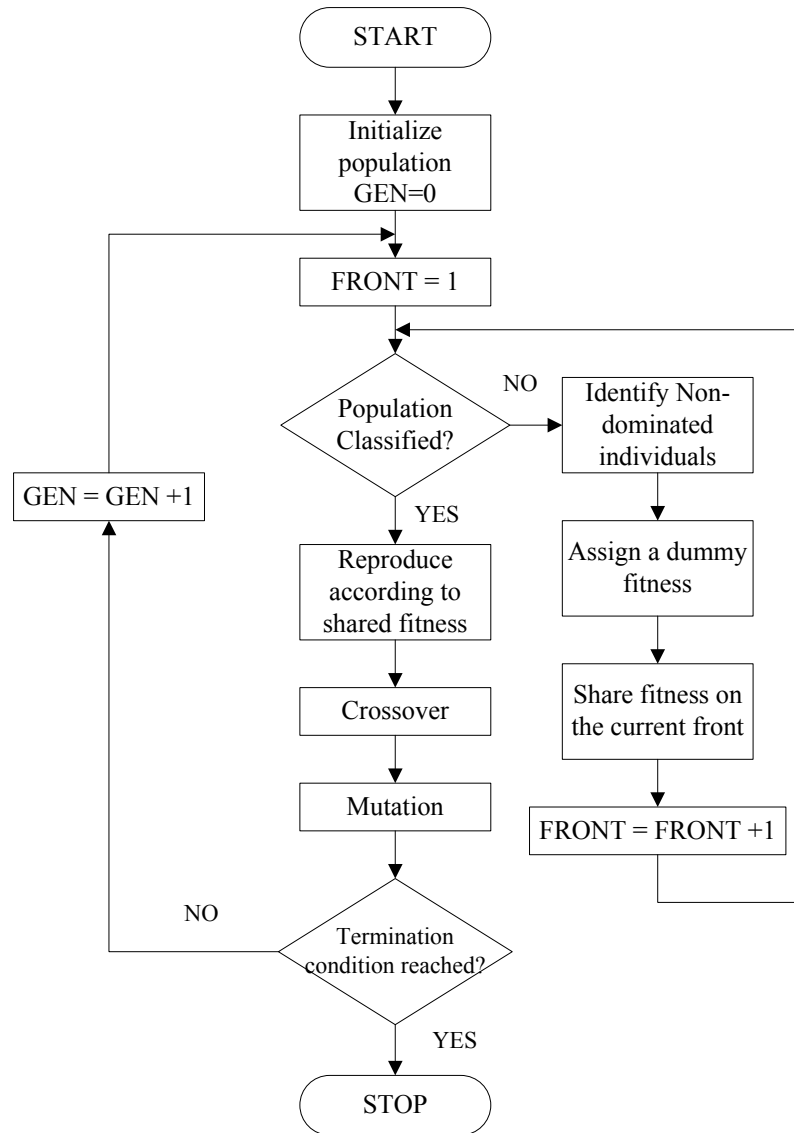


Figure 4.3 Flowchart of the non-dominated sorting genetic algorithm (NSGA)

4.2.3.1 Fitness assignment

The first step in NSGA is to sort the population into a number of mutually exclusive non-dominated sets (fronts). There are many approaches that have been suggested for finding the non-dominated set of solutions from a given population of solutions, all of which have different computational complexities. For example, the step-by-step procedure for a naïve and basic approach of finding the non-dominated set P' in a given population P of size N is as follows (Deb 2001):

Step 1: Set the solution counter $i = 1$ and create an empty non-dominated set P' .

Step 2: For solution $j \in P$ (but $j \neq i$), check if solution j dominates solution i using the two aforementioned conditions in Section 4.2.1; if yes, go to Step 4.

Step 3: If more solutions are left in P , increment j by one and go to Step 2; otherwise, set $P' = P' \cup \{i\}$.

Step 4: Increment i by one. If $i \leq N$, go to Step 2; otherwise stop and declare P' as the non-dominated set.

Once the classification task is done, it is clear that all solutions in the first non-dominated set are equally important, and represent the best in terms of their closeness to the true *Pareto* optimal front in the population. The solutions in the second non-dominated set are the second best in the population, and so on.

Following the classification of the population, fitness assignment in NSGA begins with the first non-dominated front. Every solution of the first non-dominated front is first assigned with the same dummy fitness value (i.e. equal to the population size) to provide equal reproductive potential to all of these individuals. In order to preserve diversity among solutions in the first non-dominated front, the initially assigned fitness of each solution is then shared based on the number of its neighboring solutions in the front. Next, the individuals in the first non-dominated front are removed from the current population temporarily, and the same procedure is carried out on the second front of non-dominated individuals. These second front solutions are assigned a new dummy fitness value, which is kept smaller than the minimum shared fitness value in the first front. This makes sure that no solution in the first front has a shared fitness worse than the assigned fitness of any solution in the second front. Thereafter, the sharing procedure is again performed on the solutions of the second non-dominated front. This process is continued until all solutions in the population are

assigned a fitness value, whereupon the population can undergo reproduction.

4.2.3.2 Fitness sharing

Fitness sharing is an important issue in NSGA. A *sharing function* is used on each front to determine the degradation of an individual's fitness due to a neighbor that is at some distance from it, as measured in some "similarity space" (Deb and Goldberg 1989).

Firstly, the sharing function $Sh(d_{ij})$ is defined in terms of d_{ij} -- a metric indicative of the distance between individuals i and j , and σ_{share} -- the sharing radius that controls the extent of sharing allowed:

$$Sh(d_{ij}) = \begin{cases} 1 - \left(\frac{d_{ij}}{\sigma_{share}} \right)^2 & \text{if } d_{ij} < \sigma_{share} \\ 0 & \text{otherwise} \end{cases} \quad (4.1)$$

The sharing function takes a value in $[0, 1]$, depending on the value of d_{ij} and σ_{share} . Any j which has a distance greater than σ_{share} from i contributes nothing to the sharing function value. Then, a niche count nc_i for individual i is formulated as the sum of all sharing function values between i and all members (including itself) belonging to the same front as follows:

$$nc_i = \sum_{j=1}^{|n|} Sh(d_{ij}) \quad (4.2)$$

where n is the number of solutions in a particular front. The niche count provides an estimate of the extent of crowding near solution i . Finally, the shared fitness of an individual i is equal to its old fitness divided by its niche count.

In the calculation of the sharing function $Sh(d_{ij})$, a normalized Euclidian distance that measures the phenotype distance between individuals i and j in the same

non-dominated front is adopted. It can be calculated as follows:

$$d_{ij} = \sqrt{\sum_{k=1}^K \left(\frac{x_k^{(i)} - x_k^{(j)}}{x_k^u - x_k^l} \right)^2} \quad (4.3)$$

where K is the number of decision variables for the problem. The parameter x_k^u and x_k^l are the upper and lower bounds of variable x_k corresponding to the value ranges for each gene in the chromosome. Choosing an appropriate value for σ_{share} is another important issue with the sharing function approach. Given normalized distance values are used, Deb and Goldberg (1989) proposed that this parameter could be determined using:

$$\sigma_{share} \approx \frac{0.5}{\sqrt[K]{q}} \quad (4.4)$$

where K is same as that used in Equation (4.3), and q is the number of equispaced niches in the search space. Therefore, the calculation of σ_{share} depends on the choice of q with this equation. If the q chosen to calculate σ_{share} is larger than the actual number of optima in the search space, the sharing function approach tends to form more niches than the function can allow. This may lead to finding a number of suboptimal solutions in addition to the optimal solutions. On the other hand, if the chosen q underestimates the actual number of optima in the search space, not all optima may be found by the sharing function approach. Thus, Srinivas and Deb (1994) suggested that a $q = 5$ to 10 may be tried in most applications where an exact number of optima is not known *a priori*.

4.2.3.3 Algorithm of fitness assignment and sharing

Considering a set of N population members, the algorithm for the fitness assignment procedure adopted in this research follows that described for NSGA in Deb (2001)

and Prasad and Park (2004):

Step 1: Choose sharing radius σ_{share} and a small positive number ε , and initialize $F_{min} = N + \varepsilon$.

Step 2: Classify population P according to non-domination:

Step 2a: Set all non-dominated sets P_j ($j = 1, 2, \dots$) as empty sets and the non-dominated front counter $j = 1$.

Step 2b: Find the non-dominated set P' of population P .

Step 2c: Update $P_j = P'$ and $P = P \setminus P'$; set $P' = \emptyset$.

Step 2d: If $P \neq \emptyset$, increase j by one and go to Step 2b. Otherwise, stop and declare all non-dominated fronts P_j , for $j = 1, 2, \dots, \rho$;

Step 3: Set the non-dominated front counter $j = 1$, for each $q \in P_j$

Step 3a: Assign dummy fitness $F_j^{(q)} = F_{min} - \varepsilon$.

Step 3b: Calculate the sharing function value $Sh(d_{qq'})$ with $q' \in P_j$ using Equation (4.1).

Step 3c: Calculate the niche count nc_q using Equation (4.2) among solutions of P_j only.

Step 3d: Calculate the shared fitness $F_j^{(q)} = F_j^{(q)} / nc_q$.

Step 4: $F_{min} = \min(F_j^{(q)} : q \in P_j)$ and set $j = j + 1$.

Step 5: If $j \leq \rho$, go to Step3; otherwise, the process is complete.

4.2.4 NSGA with Proposed Elitist Strategies

NSGA is somewhat lacking in both on-line performance (rapid convergence to good solutions) and off-line performance (superior quality of the final solution) (Bagchi

2001). One key reason is that NSGA does not preserve good solutions found from one generation to the next generation. Since chance is involved, good solutions lost in one generation may or may not reappear in the future. Therefore, the use of elitist strategies is proposed in this study to improve the performance of the original NSGA in solving the multiobjective precast production rescheduling problem.

4.2.4.1 Overview of elitist strategies

De Jong (1975) suggested the use of an elitist policy in the single objective genetic algorithm in order to prevent losing the best individuals due to sampling effects or operator disruption. The elitist policy always includes the best individuals of the current population into the next population. In the context of multiobjective optimization, the meaning of elite solutions is different from that in single objective optimization. A set of solutions that belongs to the best non-dominated front in each generation are considered elite individuals; all these solutions are equally important. Thus, the size of the elite set grows with each generation and can become significant compared to the size of the population, especially when the *Pareto* optimal set can admit an infinite number of solutions. This substantially complicates the incorporation of elitism in multiobjective optimization, especially in the appropriate selection of elite individuals. The use of elitism in evolutionary multiobjective optimization is still a subject of research (Laumanns *et al.* 2001).

Currently, there are several elitist strategies described in the literature. The first strategy is to copy the best individuals from the current generation to the next generation directly. The objective vectors of these individuals are either non-dominated with respect to all the objectives considered (Tamaki *et al.* 1994) or optimizing one of these objectives (Anderson and Lawrence 1996; Todd and Sen

1997). The second strategy is to choose better individuals with a comparison between the offspring individuals and the parent individuals to form the population of next generation. Such a comparison can be performed between the offspring and their corresponding parent only (Rudolph 1998) or between the offspring population and the parent population (Bagshi 1999; Deb *et al.* 2000). The third strategy, different from the first two strategies, maintains an external elite set of individuals that are non-dominated among all the solutions generated so far. In each iteration, the external set is updated and some non-dominated solutions in the set are selected to fill up a certain percentage of the new population or the parent population (Ishibuchi and Murata 1996; Murata *et al.* 1996; Parks and Miller 1998; Zitzler and Thiele 1998; and Knowles and Corne 2000). The members picked from the external set for insertion are either selected at random or according to some criteria, such as the period that an individual has stayed in the set.

4.2.4.2 Proposed elitist strategies for global search

The elitist strategy adopted in this study maintains an external archive of elite individuals. After the population of the new generation has been evaluated, this archive is updated with new generated non-dominated solutions. In order to prevent the archive size from growing too large with the passing of generations, only individuals that are not identical with the existing archived solutions in the *phenotypic* space (decoded decision values) are archived. Given K is the number of decision variables, individual i is dissimilar to individual j if

$$\sum_{k=1}^K s_k < K$$

$$\text{where } s_k = \begin{cases} 1 & \text{if } x_k^{(i)} = x_k^{(j)} \\ 0 & \text{otherwise} \end{cases} \quad (4.5)$$

Then, a fixed proportion ($p_e\%$) of the new population will be selected from the archive. In the early stages of a run, the number of solutions in the archive may be less than the number to make up the required proportion of the new population, so all the archived solutions are inserted into the new population. Once the archive size exceeds the required proportion of population, then $p_e\%$ of the new population is chosen from the archive. The remainder of the new population is generated by applying crossover and mutation to the selected parent solutions from the current population and new non-dominated solutions will be generated through this process. Figure 4.4 shows a flowchart depicting the NSGA with the proposed elitist strategy.

There are options in the way that elite solutions in the archive are chosen for inclusion into the new population; two selection methods are chosen for investigation in this research.

(1) Random selection: Non-dominated solutions in the elite archive are selected randomly to make up the fixed proportion of the new generation. This selection method is widely found in the literature describing elitist strategies.

(2) Clustering selection: Instead of having an identical objective value for single-objective optimization, non-dominated solutions in the elite archive may have different objective vectors in the precast rescheduling problem with multiobjective optimization. It is possible to cluster the different solutions based on their objective vectors. There may be an uneven distribution of elite solutions such that those in a particular cluster outnumber those in other clusters. The issue with using random selection (without consideration of clusters) is that non-dominated solutions in small clusters would have much less chance to be selected than those in larger clusters which may form an overwhelming majority in the elite population. Clustering seemed

like a good way of preserving diversity in the search population, and to inhibit bias in the evolutionary search. With clustering selection, elite solutions with different objective function vectors have equal opportunities to survive from one generation to the next generation. Given an external elite archive \bar{P} of size \bar{N} , the following describes the steps in selecting \bar{N}' individuals from this archive with clustering selection algorithm:

Step 1: Make individuals with the same objective function vector in the \bar{P} to a distinct cluster; then declare all clusters C_i with a size of \bar{N}_i ($i = 1, 2, \dots, N_c$).

Step 2: Let N_u equal to the quotient of $\frac{\bar{N}'}{N_c}$; set the cluster counter $i = 1$ and the counter for the number of selected individual $j = 0$.

Step 3: If $\bar{N}_i \geq N_u$, randomly select N_u different individuals in C_i and update $j = j + N_u$; otherwise, select all \bar{N}_i individuals in C_i and update $j = j + \bar{N}_i$.

Step 4: Set $i = i + 1$ and remove the selected individuals from C_i .

Step 5: If $i \leq N_c$, go to Step 3; otherwise, go to Step 6.

Step 6: If $j < \bar{N}'$, randomly select $\bar{N}' - j$ different individuals left in all C_i ($i = 1, 2, \dots, N_c$); otherwise, the selection process is complete.

In this study, NSGA-ESI stands for the variation of NSGA with the elitist strategy using random selection, whilst NSGA-ESII represents the variation of NSGA with the elitist strategy using clustering selection.

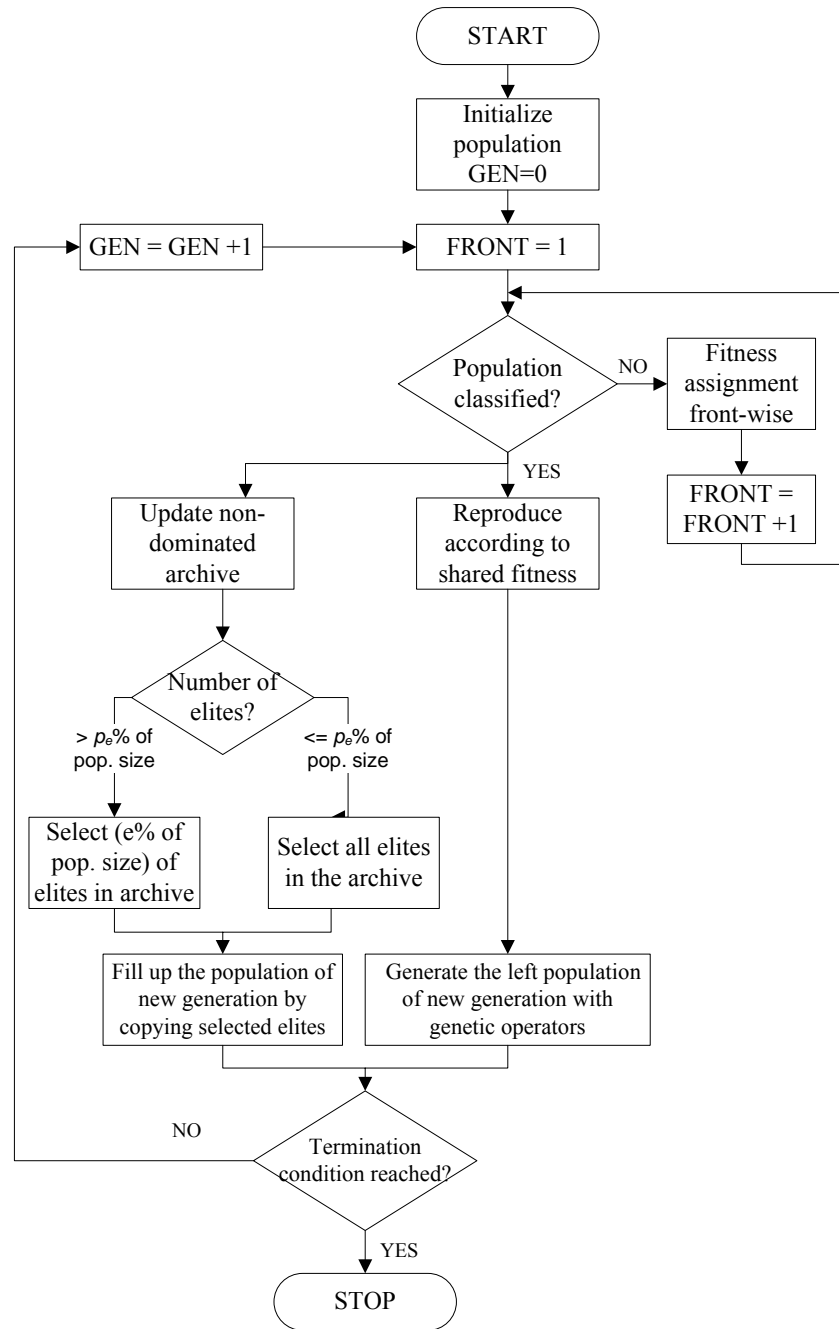


Figure 4.4 Flowchart of the NSGA with proposed elitist strategy

4.3 Implementation of GAs in Global Search

The basic mechanisms of GAs and the four algorithms, namely the weighted sum GA, NSGA, NSGA-ESI and NSGA-ESII, have already been described. The following

sections discuss program implementation details in connection with the application of these algorithms in multiobjective optimization.

4.3.1 Chromosome Representation

The choice of representation conditions all the subsequent steps in the implementation of GAs (Gen and Cheng 1997). A good representation should correctly describe the candidate solutions, facilitate constraint handling, and simplify the encoding/decoding of the chromosome. In this research, GAs are applied to decide not only *how* to resolve the disturbances with heuristic strategies, but also the *order* in which the disturbances are to be resolved. A custom chromosome structure is designed to encode the schedule repair decisions. The chromosome consists of equal numbers of D-genes (disturbance genes) and H-genes (heuristics genes); each set occupying half of the chromosome. Each disturbance to be resolved is represented by a pair of D and H-genes. The length of the chromosome is therefore equal to the number of decision variables, which is twice the number of disturbances to be resolved.

The D-genes are encoded with random numbers that serve as sort keys that determine the resolution priority of the disturbances. The index of each D-gene in the chromosome string also refers to a data structure that stores information on the disturbance, such as the element type, quantity and due date for delivery. The random keys representation (Bean 1994) encodes a solution with random numbers from a specified range and is suitable for problems in which precedence relationships are emphasized. Such a representation overcomes the issue of two feasible solutions not always resulting in a feasible offspring solution. For instance, if permutation encoding was adopted instead, the D-genes might look like [1 3 5 4 2], as shown in Figure 4.5. In this alternative representation, a disturbance is represented by a gene with a

particular value, whilst the position of this gene related to other genes indicated the resolution order for the disturbance. Under a crossover between the third and fourth gene with another chromosome [4 1 5 2 3], the offspring are [1 3 5 2 3] and [4 1 5 4 2]. It is obvious that the offspring are illegal since they contain more than one occurrence of a disturbance and leave out another disturbance entirely. On the other hand, the random key representation handles this by using random values as gene values, and these values are used as sort keys to determine the sequencing priority. As shown in Figure 4.5, suppose the two parent chromosomes mentioned above are represented with random numbers as [24 566 83 285 192] and [230 579 897 156 428]. With the same crossover, offspring are [24 566 83 156 428] and [230 579 897 285 192]. Translated to permutation representation, this gives the two disturbance resolution orders [1 3 4 5 2] and [5 1 4 2 3]. Thus, the problem of illegal offspring is eliminated.

On the other hand, the ordinal value of the heuristic used to resolve a disturbance is encoded in direct representation in the H-genes. For example, an allele value of 1 means that heuristic H1 will be used to resolve the corresponding disturbance.

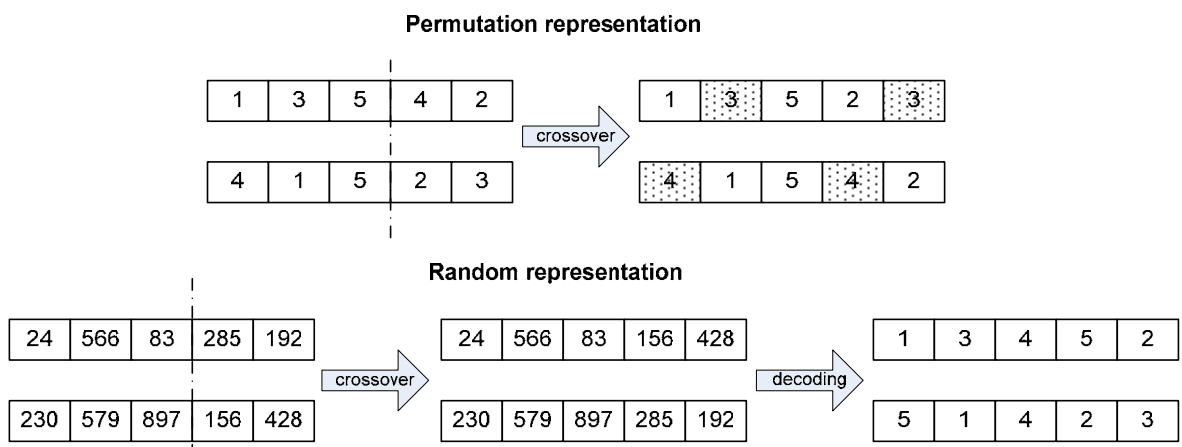
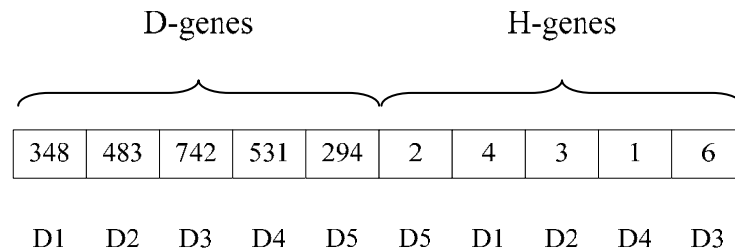


Figure 4.5 Illustration of random representation

4.3.2 Decoding

Upon decoding the chromosome, the sequence of resolving disturbances is determined by sorting the disturbances in increasing order of their D-gene values. The heuristics used are obtained from the corresponding H-gene. Therefore, for the chromosome shown in Figure 4.6, the sequence of resolving disturbances is $D5 \rightarrow D1 \rightarrow D2 \rightarrow D4 \rightarrow D3$, which is determined by sorting the disturbances in increasing order of the gene values. D5 is resolved using heuristic H2, D1 with heuristic H4, and so on.



Note: The sequence of resolution priority and corresponding heuristics used are $D5(H2) \rightarrow D1(H4) \rightarrow D2(H3) \rightarrow D4(H1) \rightarrow D3(H6)$

Figure 4.6 Chromosome representations and decoding

4.3.3 Objective Functions

In this study, two objectives are selected for evaluating the quality of repaired schedules, namely F_D and F_I . F_D is the sum of element units that are delivered late and F_I is the sum of element units that are out of a prescribed band for the inventory levels. F_D and F_I are calculated using Equations (3.8) and (3.9) respectively.

For the weighted sum genetic algorithm, these two objectives are combined in a straightforward way into a single objective F , by taking product of F_D and F_I with weight $r \in [0,1]$. The new objective function is represented as follows:

$$\min F = rF_D + (1-r)F_I \quad (4.6)$$

4.3.4 Relation among chromosomes, schedules and objective functions

In the schedule repair problem, there is a two-layer mapping relationship between chromosomes and their objective function values. As illustrated in Figure 4.7, different repaired schedules are generated by repairing the existing schedule using the information decoded from the D- and H-genes in the chromosomes to obtain the priority of disturbance resolution and the corresponding heuristics. Then, these repaired schedules are evaluated on each of the two measures, F_D and F_I . The quality of each schedule is represented as a vector containing the two objective values. This two-level mapping scheme from chromosome to objective function vector is more complicated than the usual one-to-one mapping relationship. For example, different chromosomes that encode different combinations of priority for disturbance resolution and corresponding heuristics can result in the same repaired schedule, and different schedules may have the same values for the two objective functions.

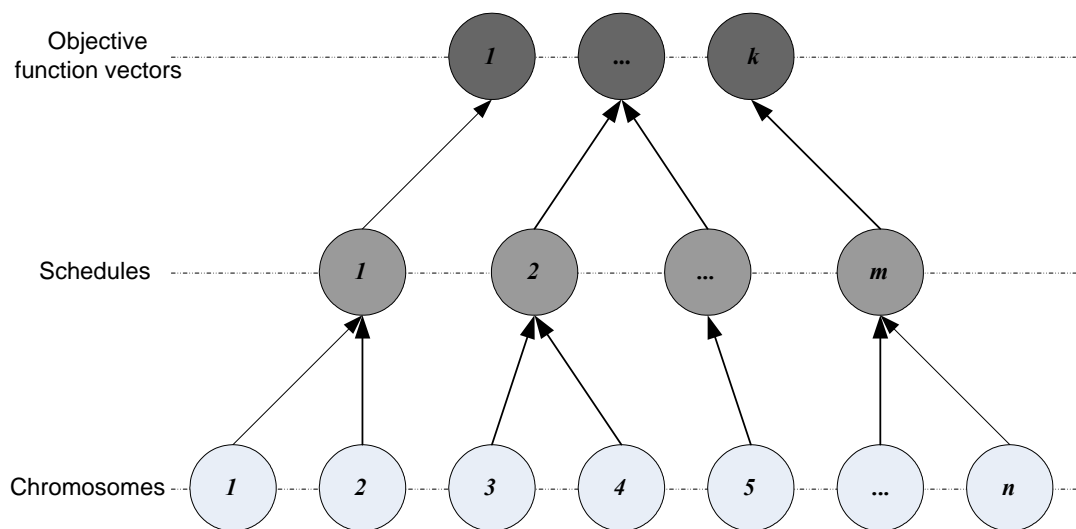


Figure 4.7 Relationship for chromosomes, schedules and objective functions

4.3.5 Genetic Operators

With the adoption of the random keys representation, traditional crossover and mutation operators can be used without modification on chromosomes. In this study, a two-point crossover is used, in which the parent individuals are split at two places in the chromosome string; whilst the mutation operator replaces the current value of the chosen gene with a value selected randomly from the allowable range based on a uniform probability distribution.

4.3.6 Software Used for the Study

In this study, the weighted sum GA was developed with PGAPack (Levine 1996). PGAPack is a parallel genetic algorithm library that provides most of the capabilities needed for coding GA applications in an integrated, seamless and portable manner. The NSGA, NSGA-ESI and NSGA-ESII were developed based on source code distributed by the Kanpur Genetic Algorithms Laboratory (<http://www.iitk.ac.in/kangal/codes.shtml>). All the code development was done in C language under the Windows operating system.

CHAPTER 5

GLOBAL SEARCH FOR REPAIRED SCHEDULES – CASE STUDY

Three test cases with different mold utilization rates were constructed from data obtained from the survey of precast factories. They were used to test the feasibility and applicability of several multiobjective genetic algorithms proposed in this research – the weighted sum genetic algorithm, non-dominated sorting genetic algorithm (NSGA) and two variations of NSGA with different elitist strategies, namely NSGA-ESI and NSGA-ESII.

5.1 Illustrative Test Cases

The utilization level of molds for precast production was an important consideration in the construction of the test cases. The utilization level is defined as the ratio of the total number of units required to be produced to the total number of available production capacity within the planning horizon. With a higher utilization level in the production schedule, it would become less flexible in accommodating disturbances with the available molds. This, in turn, may increase the level of difficulty in finding alternative repaired schedules. The utilization rate used reflects that observed in the local precast industry and ranged from a lower limit of 0.60 to the maximum of 0.90. It was decided that a utilization level above 0.9 would make it impractical to accommodate disturbances. Utilization rates of 0.60, 0.75 and 0.90 respectively were used in the test cases to represent situations of low (L-U), medium (M-U) and high (H-U) level of utilization rate.

Three examples have been constructed based on field data obtained from the survey of local precast factories. Each example involves a schedule for the precast items of a project over a time span of four weeks. The schedules involve three types of elements (E1, E2 and E3) produced by four molds (M1, M2, M3, and M4). These four molds belong to one group, and all three element types are produced by any of the molds with changeover accomplished within one day. The detailed site demand for each element type, as well as corresponding production schedules determined by manual calculation is shown in Appendixes. The minimum lead time for all elements is assumed to be 2 days. Other parameter values used in these examples like the total demand, initial stock levels, lower and upper bounds for inventory levels of each element type are shown in Table 5.1.

Table 5.1 Problem parameters

Element type	Total demand (Low)	Total demand (Medium)	Total demand (High)	Initial Stock*	Upper stock limit*	Lower stock limit*
E1	31	37	45	6	10	4
E2	8	9	7	2	3	1
E3	31	37	45	6	10	4

Note: *--These parameters are the same for the three examples.

Now consider the following events occurring:

- (1) The site rejected one piece of E2 because of a quality problem; hence, an additional piece of E2 has to be produced for delivery on Day 4;
- (2) The contractor informs the precast factory to advance the due date for delivering some elements since actual progress on-site is better than that anticipated in the earlier schedule. The new delivery schedule proposed by the site is as follows: (i) The requirement for six pieces of E1 and E3 to be delivered on Day 30 is now changed to three pieces of each element type on Day 21 and Day 25; and (ii) The

two pieces of E2 earlier planned to be delivered on Day 28 is now scheduled for Day 25.

Therefore, there are six schedule disturbances caused by these events that are identified to be solved within the planning horizon, as shown in Table 5.2.

Table 5.2 Characteristics of schedule disturbances

Disturbance	Element Type	Quantity	Original Due Date	New Due Date	Types of Disturbance
D1	E2	1	--	Day 4	Change in quantity; quality problem
D2	E1	3	Day 30	Day 21	Change in due date due to site progress
D3	E3	3	Day 30	Day 21	Change in due date due to site progress
D4	E2	2	Day 28	Day 25	Change in due date due to site progress
D5	E1	3	Day 30	Day 25	Change in due date due to site progress
D6	E3	3	Day 30	Day 25	Change in due date due to site progress

Insertion has been selected as a basic repair action in the case study. Since the three element types can be produced by any of the molds, insertion is based on the multiple mold approach in which all four molds (M1, M2, M3, and M4) can be used to resolve a disturbance. The search for the point of insertion into the original schedule can be carried out in a number of ways depending on choices made on many factors:

- (i) The search sequence used (either in a parallel manner across all mold schedules simultaneously or in a serial manner for each mold schedule);
- (ii) The direction of search (starting either from the beginning or the end of the mold schedules); and
- (iii) The manner of insertion (opportunistic or deterministic insertion).

With different combinations of these search characteristics, six insertion-based

heuristic strategies have been developed to resolve schedule disturbances in the study. In addition, sub-contracting was also considered as a seventh heuristic in the experiments. It means that the precaster will subcontract elements involved in the disturbance with the result that there is no repair action needed to accommodate the disturbance in the current schedule. The representation and description of repairing actions for these heuristic strategies are summarized in Table 5.3.

Table 5.3 Heuristics representation

Heuristic	Symbolic code	Description of repair actions
H1	S/ASAP/OI	Opportunistic insertion applied in a serial manner across molds and from the beginning of schedules
H2	S/BS/OI	Opportunistic insertion applied in a serial manner across molds and from the end of schedules
H3	S/ASAP/DI	Deterministic insertion on specific days, whilst affected initial productions are resolved by H1
H4	P/ASAP/OI	Opportunistic insertion applied in a parallel manner across molds and from the beginning of schedules
H5	P/BS/OI	Opportunistic insertion applied in a parallel manner across molds and from the end of schedules
H6	P/ASAP/DI	Deterministic insertion on specific days, whilst affected initial productions are resolved by H4
H7	Sub-contracting	No repair action

Note: S—Serial; P—Parallel; ASAP—As-soon-as-possible; BS—Backward scheduling; OI—Opportunistic insertion; DI—Deterministic insertion.

5.2 Performance Measurement

Appropriate performance metrics must be selected to enable meaningful comparison of these algorithms. Many metrics in the literature of the multiobjective optimization evolutionary algorithms measure performance in the phenotype domain by comparing the solution set generated by the algorithm with the true *Pareto* front for the problem. However, no single metric can entirely capture total performance for these algorithms, because some of them measure algorithm effectiveness while others measure

efficiency (Coello *et al.* 2002). Before introducing the performance metrics selected, the following notation is used to define the various solution sets generated by the algorithms in this study:

- (1) The solution set obtained at the final iteration of the algorithms is denoted by $PF_{current}$.
- (2) The optimal trade-off surface is denoted by PF_{true} . Due to the fact that the true *Pareto* front is generally not known for real-world problems, PF_{true} refers to the “best” *Pareto* front found so far by any of the selected algorithms in this study.

The following two performance metrics described by Van Veldhuizen (1999) in the phenotype domain have been selected to compare the performance of these GA-based algorithms. They will be calculated after all the experiments have been performed.

- (1) Error Ratio (ER): it reflects the proportion of vectors in $PF_{current}$ that are not members of PF_{true} . With a solution set having n vectors, it is mathematically defined as:

$$ER = \frac{\sum_{i=1}^n e_i}{n} \tag{5.1}$$

where $e_i = \begin{cases} 0 & \text{if vector } i, (i = 1, \dots, n) \in PF_{true} \\ 1 & \text{otherwise} \end{cases}$

The lower the ER, the more vectors in $PF_{current}$ are also members of PF_{true} .

- (2) Generational Distance (GD): it measures how “far” $PF_{current}$ found by the algorithm is from PF_{true} . The definition of GD is:

$$GD = \frac{\sqrt{\sum_{i=1}^n d_i^2}}{n} \tag{5.2}$$

where d_i is the distance between solution i in $PF_{current}$ and the closest solution that belongs to PF_{true} , and n is the number of vectors in $PF_{current}$. The farther the solution set is from PF_{true} , the greater is the generational distance.

5.3 GA Parameters

Several parameters, such as population size (N_{pop}), number of generations (N_{gen}), crossover probability (p_c), mutation probability (p_m), sharing radius (σ_{share}) and elite proportion (p_e) are considered to affect the performance of GAs. The optimal values for these parameters can not be ascertained by applying fixed rules; in fact, optimal GA parameters are known to be notoriously difficult to determine (Myers 2001). In this study, these values were determined by fine tuning default values over several runs of GAs on a trial example.

Ten separate runs were conducted for each set of these parameters with a different seed for the random number generator that is equally distributed between 0.1 and 1.0. Convergence is reached when the GA run attains a benchmark. This benchmark is obtained by selecting the “best” *Pareto* front obtained so far. Therefore, the number of times out of the ten runs where the solution sets converge to this front is noted.

Table 5.4 summarizes the performance of a set of experiments at the end of 70 generations. From the experiments, it was found that σ_{share} has an impact on the NSGA performance, as the experiments converged better with σ_{share} equaling to 0.4172 (for $q=10$) than others in principle. With this sharing radius determined, the results show that NSGA performed well under the population size of 400. With a mutation probability of 1%, the experiments returned the highest frequency of convergence irrespective of whether the crossover probability is 30% or 40%. Since two variations of NSGA, namely NSGA-ESI and NSGA-ESII, use elitist strategies, the elitist proportion is another important parameter considered in this study. Based on the parameters determined for the basic version of NSGA, trials were conducted with different values of the elitist proportion. There was no apparent improvement for

values of the elitist proportion higher than 10%. From the results of these experiments, it was decided to use the values shown in Table 5.5 for the GA parameters in the subsequent experiments.

Table 5.4 Frequency of convergence with different GA parameters

N_{pop}		300						400						500					
p_c (%)		30		40		50		30		40		50		30		40		50	
p_m (%)		0.5	1	0.5	1	0.5	1	0.5	1	0.5	1	0.5	1	0.5	1	0.5	1	0.5	1
σ_{share}	0.4372 ($q=5$)	6	9	9	7	10	7	8	9	9	7	9	4	9	8	8	8	9	8
	0.4127 ($q=10$)	8	8	8	5	8	7	9	10	9	10	9	8	8	10	9	9	9	10
	0.3990 ($q=15$)	8	6	8	10	6	6	10	8	9	9	6	8	9	9	9	10	7	9

Table 5.5 GA parameters used in case study

Parameters	Values
Population size, N_{pop}	400
Number of generation, N_{gen}	70
Crossover probability, p_c	0.40 [#]
Mutation probability, p_m	0.01
Sharing radius, σ_{share} ($q=10$)	0.4127
Elite proportion, p_e	10%

Note: #--In the weighted sum GA, $p_c=0.7$.

5.4 Results and Discussion

As discussed in the previous chapter, four GA-based algorithms, which are the weighted sum GA, NSGA, NSGA-ESI and NSGA-ESII, were selected to generate the alternative repaired schedules with the two conflicting objectives. Among these four algorithms, the weighted sum GA differs with the latter three *Pareto*-based algorithms in terms of generating a single non-dominated solution each time with a varied weight combination. Therefore, the experiments using the weighted sum algorithm were

conducted for the three test cases to compare against the other three algorithms in terms of the non-dominated solution sets found.

An exhaustive search involving increasing values of the weight (r) for the weighted sum approach was done. The value of r was varied in steps of 0.05 between 0 and 1. In order to avoid stochastic error, 5 runs were made for each r value to search the corresponding optimal solution. With 21 weight combinations considered in the study, 105 runs were made for each of the three examples using the weighted sum algorithm. All the optimal solutions found with the different weight combinations generated a solution set for this algorithm. On the other hand, the other three *Pareto*-based evolutionary algorithms, namely NSGA, NSGA-ESI and NSGA-ESII, were also applied to the three examples. For each combination of algorithm and example, 10 separate runs of the GA were made, each starting with a different random number seed equally distributed between 0.1 and 1.0. The results of these experiments are discussed in the following sections.

5.4.1 The L-U example

Based on the complete set of experiments involving the L-U example, four non-dominated solutions were found in the set of PF_{true} , namely the points (0, 7), (1, 4), (3, 3) and (4, 2) in the objective function space.

The results of the weighted sum GA are shown in Figure 5.1. Except for two extreme values with respect to two objectives F_D and F_I when r equals to 0 and 1 respectively, the solutions found with varied r from 0.05 to 0.95 are shown in the figure. The non-dominated solution set found by the weighted sum GA consists of three solutions: solution (4, 2) was obtained for value of r from 0.05 to 0.4; solution (1, 4) was

obtained for value of r from 0.4 to 0.75; and solution (0, 7) was obtained for value of r from 0.75 to 0.95. PF_{true} is also depicted in Figure 5.1; it is seen that the non-dominated solution (3, 3) has not been found by the algorithm. It could be the reason that this solution is in the “concave” portion of the trade-off surface; no solution point (3, 3) was ever found by any experiment with the weighted sum algorithm. It is known that a weighted sum GA will miss points on the concave portions of the trade-off curve.

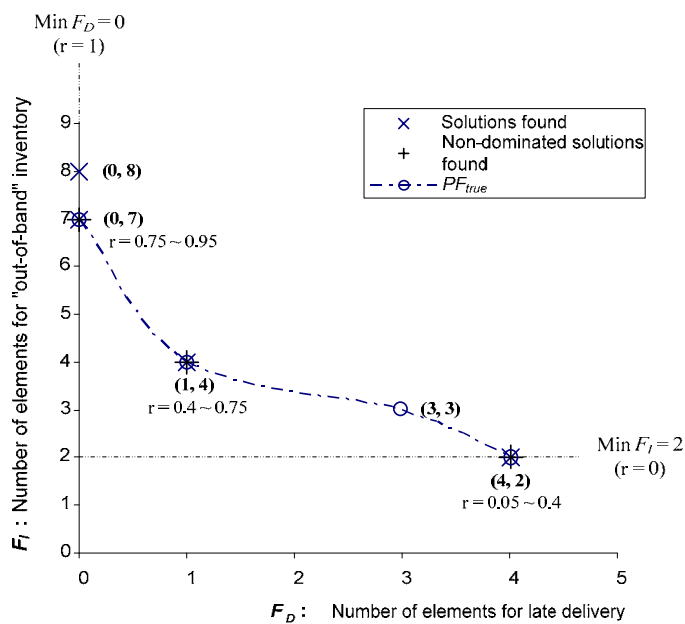
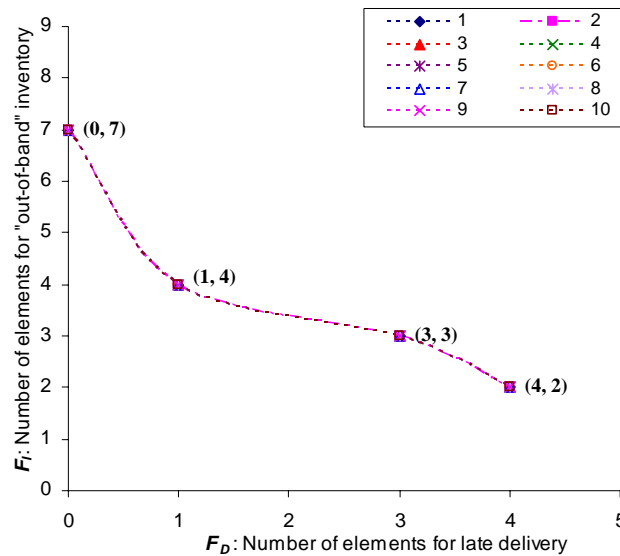


Figure 5.1 $PF_{current}$ found with the weighted sum GA for the L-U example

The solution sets ($PF_{current}$) generated by NSGA in the last iteration for the L-U example are displayed in Figure 5.2. The solution sets generated by NSGA-ESI and NSGA-ESII are the same as those generated by NSGA, which are not shown here. All four non-dominated solutions in PF_{true} were found by these three algorithms. Table 5.6 shows the performance metrics for the different runs involving the NSGA, NSGA-ESI and NSGA-ESII runs. It shows that every run of NSGA and NSGA-ESII successfully found PF_{true} for the L-U example. On the other hand, 3 out of 10 NSGA-ESI runs could only find three of the four non-dominated solutions [(0, 7), (1,

4) and (3, 3)] and consistently missed the solution (4, 2).

In summary, the convergence frequency was 10 for both NSGA and NSGA-ESII, and 7 for NSGA-ESI, as shown in Figure 5.3. Among the three *Pareto*-based algorithms, NSGA and NSGA-ESII performed better than NSGA-ESI for the L-U example. In contrast to the weighted sum algorithm, these findings also suggest that these *Pareto*-based algorithms (1) performed better in generating multiple non-dominated solutions from a single run, and (2) was able to find non-dominated solutions even in the nonconvex portions of the trade-off surface.



*: All of the 10 runs found PF_{true} .
Same for $PF_{current}$ found with NSGA -ESI and NSGA-ESII for the L-U example.

Figure 5.2 $PF_{current}$ found with NSGA for the L-U example

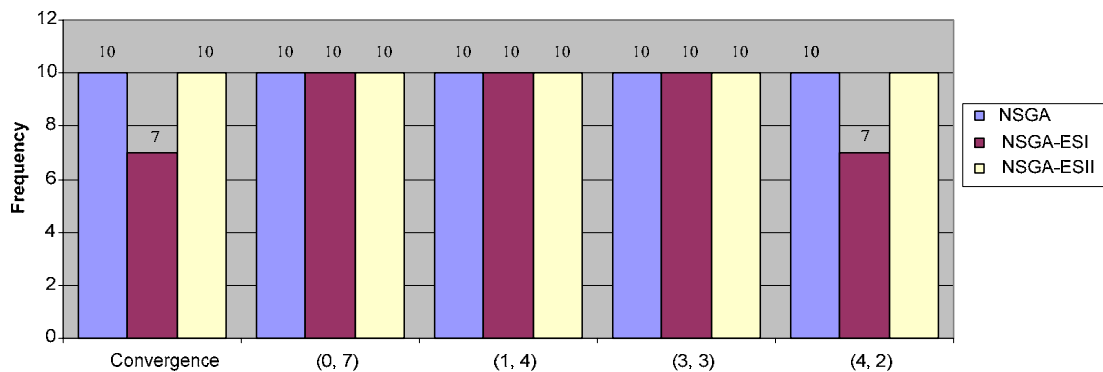


Figure 5.3 Frequency of convergence and solutions found for the L-U example

Table 5.6 Performances of NSGA, NSGA-ESI and NSGA-ESII for the L-U Example

Run	NSGA				NSGA-ESI			
	$PF_{current}$	ER	GD	Computing time (sec.)	$PF_{current}$	ER	GD	Computing time (sec.)
1	(0, 7) (1, 4) (3, 3) (4, 2)	0	0.00	4.6	(0, 7) (1, 4) (3, 3) (4, 2)	0	0.00	8.0
2	(0, 7) (1, 4) (3, 3) (4, 2)	0	0.00	4.4	(0, 7) (1, 4) (3, 3)	0	0.00	8.0
3	(0, 7) (1, 4) (3, 3) (4, 2)	0	0.00	4.5	(0, 7) (1, 4) (3, 3) (4, 2)	0	0.00	7.9
4	(0, 7) (1, 4) (3, 3) (4, 2)	0	0.00	4.8	(0, 7) (1, 4) (3, 3) (4, 2)	0	0.00	7.9
5	(0, 7) (1, 4) (3, 3) (4, 2)	0	0.00	4.6	(0, 7) (1, 4) (3, 3)	0	0.00	7.8
6	(0, 7) (1, 4) (3, 3) (4, 2)	0	0.00	4.3	(0, 7) (1, 4) (3, 3)	0	0.00	8.1
7	(0, 7) (1, 4) (3, 3) (4, 2)	0	0.00	5.0	(0, 7) (1, 4) (3, 3) (4, 2)	0	0.00	7.8
8	(0, 7) (1, 4) (3, 3) (4, 2)	0	0.00	4.8	(0, 7) (1, 4) (3, 3) (4, 2)	0	0.00	7.9
9	(0, 7) (1, 4) (3, 3) (4, 2)	0	0.00	4.8	(0, 7) (1, 4) (3, 3) (4, 2)	0	0.00	7.9
10	(0, 7) (1, 4) (3, 3) (4, 2)	0	0.00	4.5	(0, 7) (1, 4) (3, 3) (4, 2)	0	0.00	7.8
Ave.		0	0.00	4.6		0	0.00	7.9

Table 5.6 Performances of NSGA, NSGA-ESI and NSGA-ESII for the L-U Example (Cont.)

Run	NSGA-ESII			
	$PF_{current}$	ER	GD	Computing time (sec.)
1	(0, 7) (1, 4) (3, 3) (4, 2)	0	0.00	10.2
2	(0, 7) (1, 4) (3, 3) (4, 2)	0	0.00	10.3
3	(0, 7) (1, 4) (3, 3) (4, 2)	0	0.00	10.1
4	(0, 7) (1, 4) (3, 3) (4, 2)	0	0.00	10.2
5	(0, 7) (1, 4) (3, 3) (4, 2)	0	0.00	10.1
6	(0, 7) (1, 4) (3, 3) (4, 2)	0	0.00	10.1
7	(0, 7) (1, 4) (3, 3) (4, 2)	0	0.00	10.2
8	(0, 7) (1, 4) (3, 3) (4, 2)	0	0.00	10.2
9	(0, 7) (1, 4) (3, 3) (4, 2)	0	0.00	10.0
10	(0, 7) (1, 4) (3, 3) (4, 2)	0	0.00	10.1
Ave.		0	0.00	10.2

5.4.2 The M-U example

The experiments with the different algorithms on the M-U example fielded three non-dominated solutions in the objective function space, namely (0, 16), (1, 12) and (3, 7); these points constitute PF_{true} for the M-U example.

The results for the weighted sum GA are shown in Figure 5.3. The non-dominated solution set found by this algorithm is identical to PF_{true} . Solution (3, 7) was obtained with settings of r from 0.05 to 0.7; solution (1, 12) was obtained with settings of r from 0.75 and 0.8; and solution (0, 16) was obtained with settings of r from 0.8 to 0.95. Solutions (1, 12) and (0, 16) were both obtained when r equals to 0.8.

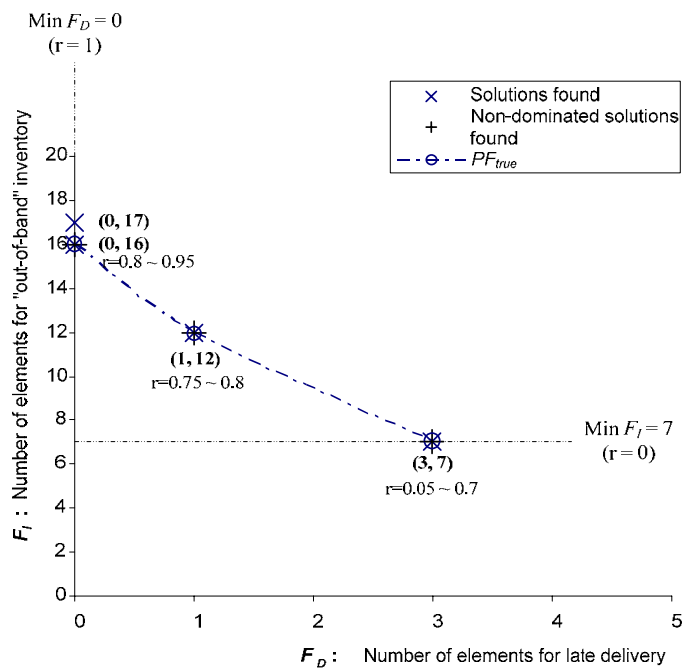


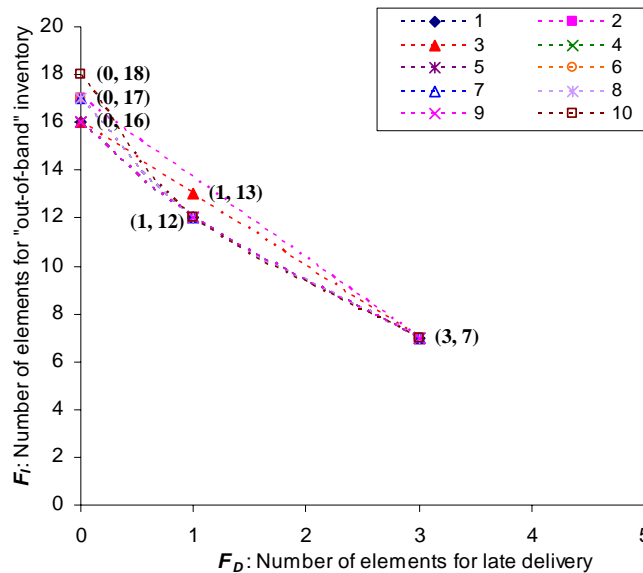
Figure 5.3 $PF_{current}$ found with the weighted sum GA for the M-U example

The solution sets found by the 10 NSGA runs on the M-U example are shown in Figure 5.4. The performance metrics for these different runs are shown in Table 5.7. Only 4 out of 10 runs (No. 1, 4, 5 and 9) converged to PF_{true} . The other 6 runs fielded the *Pareto* fronts that are not as good as the one defined by PF_{true} ; there involved

points like (0, 18), (0, 17) and (1, 13), which are dominated by points in PF_{true} .

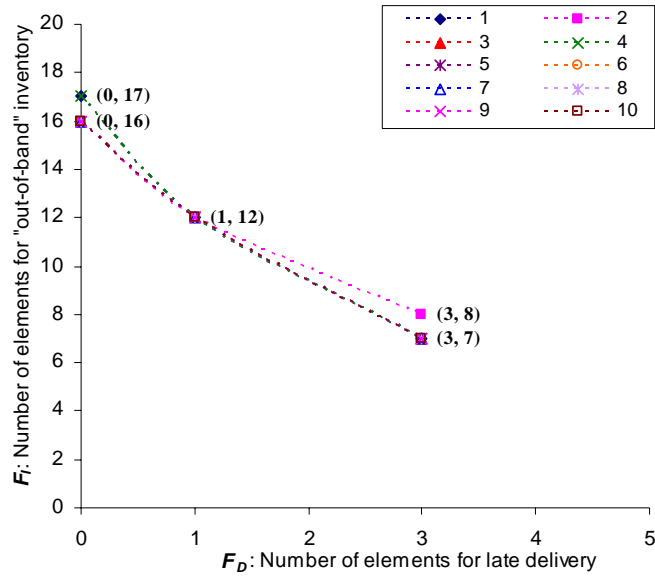
The solution sets generated by the 10 NSGA-ESI runs for the M-U example are shown in Figure 5.5. There were seven runs that converged to PF_{true} . Although the other three runs (No. 1, 2 and 4) found 3 non-dominated solutions in their $PF_{current}$, non-zero values for their GD and ER (Table 5.7) show that these fronts contained some solutions that are not members in PF_{true} . They are solution (0, 17) generated in the first and fourth run and solution (3, 8) generated in the second run.

The solution sets generated by the 10 NSGA-ESII runs for the M-U example are shown in Figure 5.6. The performance statistics for these solution sets in terms of ER and GD are provided in Table 5.7. All of the 10 NSGA-ESII runs found PF_{true} , and no other non-member solutions were found in the *Pareto* front returned by any of the runs.



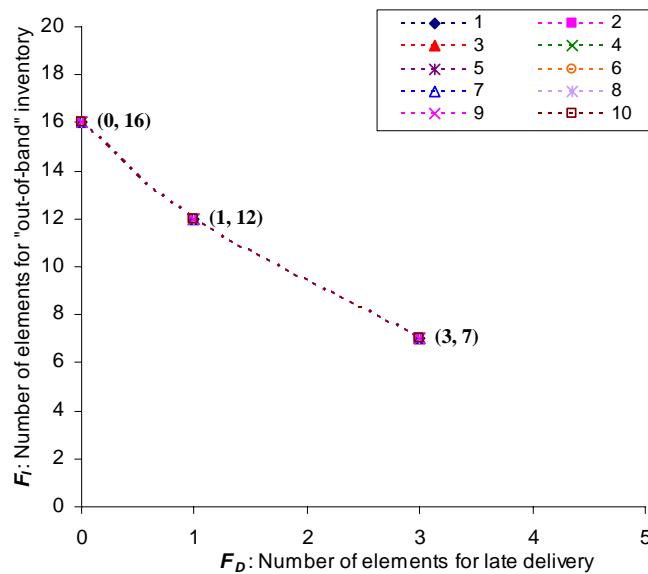
*: Runs 1, 4, 5 and 9 converged to PF_{true} , while the other six runs did not.

Figure 5.4 $PF_{current}$ found with NSGA for the M-U example



*: Runs 1, 2 and 4 did not converge to PF_{true} , while the other seven runs converged to PF_{true} .

Figure 5.5 $PF_{current}$ found with NSGA-ESI for the M-U example



*: All of the 10 runs converged to PF_{true} .

Figure 5.6 $PF_{current}$ found with NSGA-ESII for the M-U example

Figure 5.7 summarizes the frequency of convergence to PF_{true} with these *Pareto*-based algorithms for the M-U example. The frequency values improved in

successive, from NSGA to NSGA-ESI, and then to NSGA-ESII. Although the average computing time for NSGA-ESII (10.6 seconds) is more than that for NSGA and NSGA-ESI (4.6 and 7.9 seconds respectively), the former is more consistent across the 10 runs in generating PF_{true} . These results suggest that the elitist strategies proposed improved the performance of NSGA by keeping good solutions throughout the entire search process for the M-U example. Moreover, the elitist strategy with clustering selection appears to perform better than that with random selection. Although the weighted sum GA also obtained the exact PF_{true} , it took more effort (105 runs) and more computing times (27 seconds) than the *Pareto*-based algorithms.

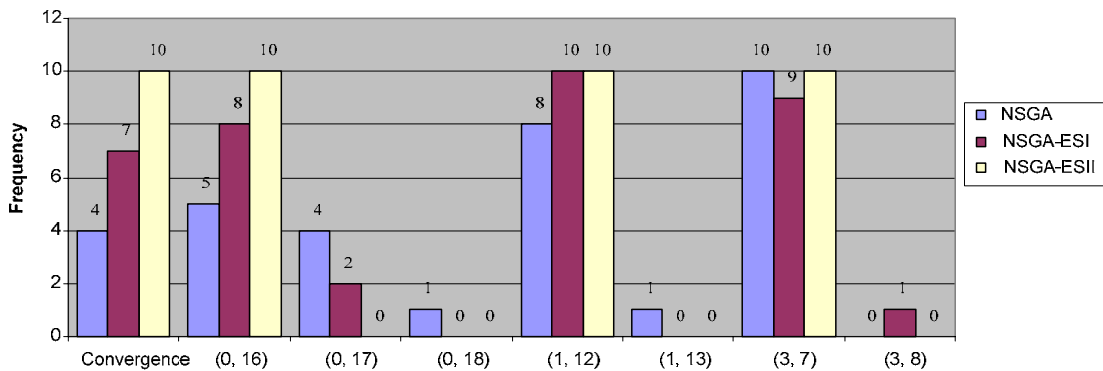


Figure 5.7 Frequency of convergence and solutions found for the M-U example

5.4.3 The H-U example

The experiments with the H-U example fielded four non-dominated solutions (8, 24), (10, 19), (12, 15) and (14, 13) as members of PF_{true} .

The results for the weighted sum GA are shown in Figure 5.8. The non-dominated solutions found by this algorithm are identical to those in PF_{true} . Solution (14, 13) was obtained with settings of r from 0.05 to 0.5; solution (12, 15) was obtained with settings of r from 0.55 to 0.65; solution (10, 19) was obtained with settings of r equaling to 0.7; and solution (8, 24) was obtained with settings of r from 0.75 to 0.95.

Table 5.7 Performances of NSGA, NSGA-ESI and NSGA-ESII for the M-U Example

Run	NSGA				NSGA-ESI			
	$PF_{current}$	ER	GD	Computing time (sec.)	$PF_{current}$	ER	GD	Computing time (sec.)
1	(0, 16) (1, 12) (3, 7)	0	0.00	4.4	(0, 17) (1, 12) (3, 7)	1/3	0.33	7.9
2	(0, 17) (3, 7)	1/2	0.50	5.2	(0, 16) (1, 12) (3, 8)	1/3	0.33	7.9
3	(0, 16) (1, 13) (3, 7)	1/3	0.33	5.1	(0, 16) (1, 12) (3, 7)	0	0.00	7.9
4	(0, 16) (1, 12) (3, 7)	0	0.00	4.4	(0, 17) (1, 12) (3, 7)	1/3	0.33	7.9
5	(0, 16) (1, 12) (3, 7)	0	0.00	4.3	(0, 16) (1, 12) (3, 7)	0	0.00	7.9
6	(0, 17) (1, 12) (3, 7)	1/3	0.33	4.6	(0, 16) (1, 12) (3, 7)	0	0.00	7.9
7	(0, 17) (1, 12) (3, 7)	1/3	0.33	4.8	(0, 16) (1, 12) (3, 7)	0	0.00	8.0
8	(0, 17) (1, 12) (3, 7)	1/3	0.33	4.6	(0, 16) (1, 12) (3, 7)	0	0.00	8.0
9	(0, 16) (1, 12) (3, 7)	0	0.00	4.5	(0, 16) (1, 12) (3, 7)	0	0.00	7.9
10	(0, 18) (1, 12) (3, 7)	1/3	0.67	4.3	(0, 16) (1, 12) (3, 7)	0	0.00	7.9
Ave.		0.22	0.25	4.6		0.10	0.10	7.9

Table 5.7 Performances of NSGA, NSGA-ESI and NSGA-ESII for the M-U Example (Cont.)

Run	NSGA-ESII			
	$PF_{current}$	ER	GD	Computing time (sec.)
1	(0, 16) (1, 12) (3, 7)	0	0.00	10.7
2	(0, 16) (1, 12) (3, 7)	0	0.00	10.5
3	(0, 16) (1, 12) (3, 7)	0	0.00	10.6
4	(0, 16) (1, 12) (3, 7)	0	0.00	10.6
5	(0, 16) (1, 12) (3, 7)	0	0.00	10.5
6	(0, 16) (1, 12) (3, 7)	0	0.00	10.6
7	(0, 16) (1, 12) (3, 7)	0	0.00	10.6
8	(0, 16) (1, 12) (3, 7)	0	0.00	10.7
9	(0, 16) (1, 12) (3, 7)	0	0.00	10.7
10	(0, 16) (1, 12) (3, 7)	0	0.00	10.6
Ave.		0	0.00	10.6

The solutions (12, 15) and (14, 13) have the same objective value when r is set to 0.5, but the former was not found within the 5 runs with this value of r . Although it has subsequently found with other r values, this indicates a potential risk of missing certain points with this kind of weighted sum approach.

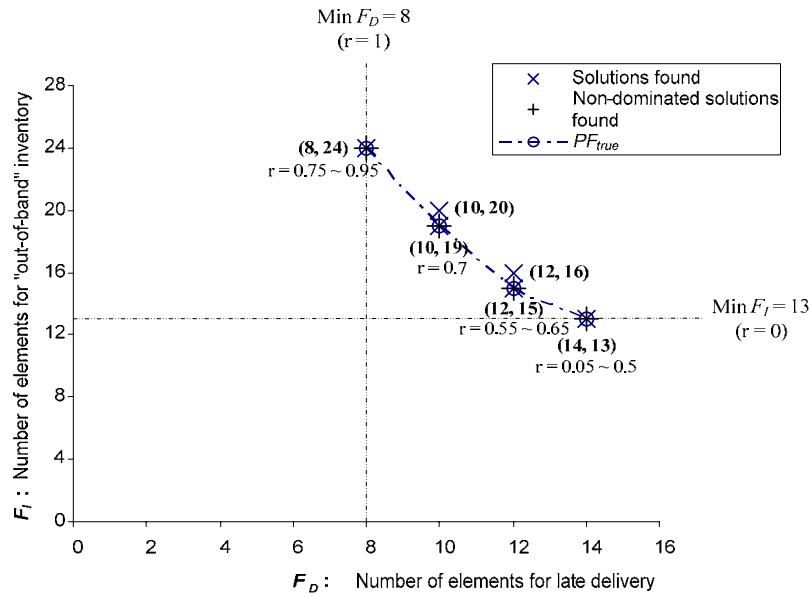
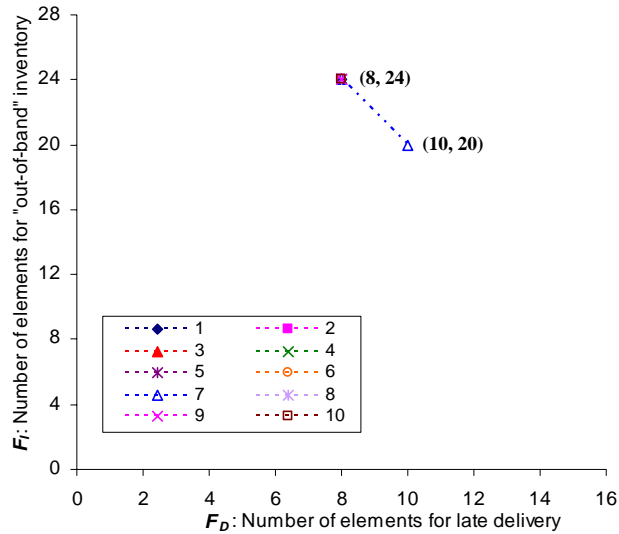


Figure 5.8 $PF_{current}$ found with the weighted sum GA for the H-U example

As shown in Figure 5.9, only two solutions (8, 24) and (10, 20) were found with the 10 NSGA runs for the H-U example. None of these NSGA runs successfully converged to the full set of PF_{true} . Based on the performance metrics for these runs shown in Table 5.8, although the values of ER and GD for 9 out of the 10 runs were zero, the solution set from each of these runs had only one non-dominated solution (8, 24). However, the second run did produce a solution set with two solutions, one of which, (10, 20) is adjacent to the non-dominated solution (10, 19).

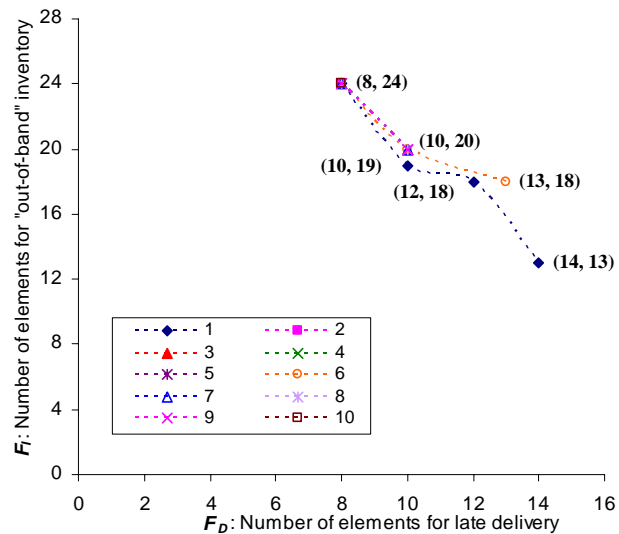
Besides the non-dominated solution (8, 24), the 10 NSGA-ESI runs also returned another two nondominated solutions in PF_{true} – (10, 19) and (14, 13), as shown in Figure 5.10. However, they were found only once in the first run. Moreover, solution

(12, 15) was not found in any of the runs. The performance metrics presented in Table 5.8 show that 7 out of the 10 NSGA-ESI runs generated the solution sets with more than one solution. However, non-zero values of the GD and ER for these fronts indicate that some solutions found are not members of PF_{true} , such as solutions (10, 20), (12, 18) and (13, 18).



*: Run 2 found two points (8, 24) and (10, 20) in $PF_{current}$.
The other nine runs only found one point (8, 24) in their $PF_{current}$.

Figure 5.9 $PF_{current}$ found with NSGA for the H-U example



*: None of the ten runs converged to PF_{true} .
Three nondominated solutions (8, 24), (10, 19) and (14, 13) were all found only in Run 1.

Figure 5.10 $PF_{current}$ found with NSGA-ESI for the H-U example

The solution sets generated by the 10 NSGA-ESII runs for the H-U example are shown in Figure 5.11. The performance metrics for the different NSGA-ESII runs are shown in Table 5.8. There are half of the 10 runs that converged to PF_{true} . Among the other 5 runs that did not converge to PF_{true} , the run (No. 6) only generated one non-dominated solution (8, 24) in its $PF_{current}$, whilst the other four runs (No. 2, 4, 7 and 10) found 3 or 4 solutions in their solution sets. Although some points in the latter four solution sets are not members of PF_{true} , such as (10, 20) and (12, 16), these solutions sets were closer to PF_{true} than those generated by NSGA-ESI in terms of the GD values.

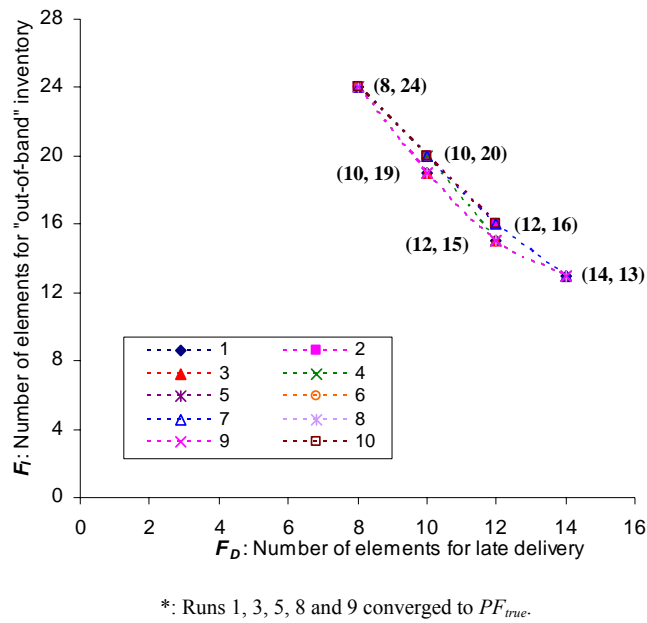


Figure 5.11 $PF_{current}$ found with NSGA-ESII for the H-U example

Figure 5.12 shows that no matter which algorithm is used, (8, 24) was the most frequently found solution among the four non-dominated solutions in PF_{true} . The frequency of convergence for NSGA, NSGA-ESI and NSGA-ESII was 0, 0 and 5 respectively. Although the performance of these three algorithms in the H-U example

is not as good as in the M-U and L-U examples, NSGA-ESII did do much better than NSGA and NSGA-ESI in the H-U example. Moreover, a single run of NSGA-ESII requires only 12.5 seconds averagely.

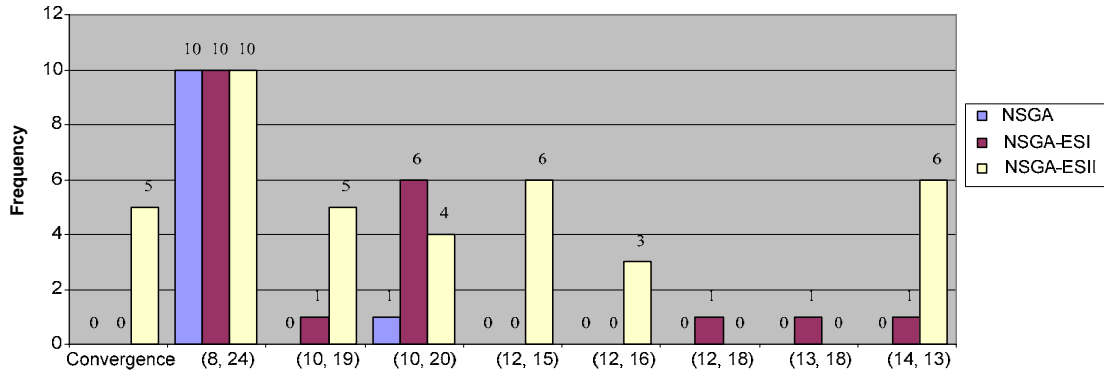


Figure 5.12 Frequency of convergence and solutions found for the L-U example

Based on the case study discussed above, GAs have proven to be capable of finding alternative repair strategies using different resolution priority for disturbances and appropriate heuristics to obtain alternative repaired schedules along a *Pareto* front. These alternatives not only provide insight of trade-offs in the rescheduling problem, but also act as valuable options for negotiation between the precaster and the contractor. However, there may still be circumstances that few adjustments/improvements are needed for these repaired schedules provided by the global search. It is desirable to continue the search for solutions satisfying particular constraints imposed by the decision makers. Therefore, a local search module, which is evoked under such circumstances, will be presented in the next chapter.

Table 5.8 Performances of NSGA, NSGA-ESI and NSGA-ESII for the H-U Example

Run	NSGA				NSGA-ESI			
	$PF_{current}$	ER	GD	Computing time (sec.)	$PF_{current}$	ER	GD	Computing time (sec.)
1	(8, 24)	0	0.00	4.8	(8, 24) (10, 19) (12, 18) (14, 13)	1/4	0.75	9.9
2	(8, 24)	0	0.00	4.5	(8, 24)	0	0.00	10.1
3	(8, 24)	0	0.00	4.7	(8, 24) (10, 20)	1/2	0.50	10.0
4	(8, 24)	0	0.00	4.5	(8, 24)	0	0.00	10.1
5	(8, 24)	0	0.00	4.6	(8, 24) (10, 20)	1/2	0.50	10.3
6	(8, 24)	0	0.00	4.4	(8, 24) (10, 20) (13, 18)	2/3	1.11	10.1
7	(8, 24) (10, 20)	1/2	0.50	4.8	(8, 24) (10, 20)	1/2	0.50	10.1
8	(8, 24)	0	0.00	4.9	(8, 24) (10, 20)	1/2	0.50	10.0
9	(8, 24)	0	0.00	4.6	(8, 24) (10, 20)	1/2	0.50	10.1
10	(8, 24)	0	0.00	4.1	(8, 24)	0	0.00	10.2
Ave.		0.05	0.05	4.6		0.39	0.44	10.1

Table 5.8 Performances of NSGA, NSGA-ESI and NSGA-ESII for the H-U Example (Cont.)

Run	NSGA-ESII			
	$PF_{current}$	ER	GD	Computing time (sec.)
1	(8, 24) (10, 19) (12, 15) (14, 13)	0	0.00	12.6
2	(8, 24) (10, 20) (12, 16)	2/3	0.47	12.3
3	(8, 24) (10, 19) (12, 15) (14, 13)	0	0.00	12.3
4	(8, 24) (10, 20) (12, 15)	1/3	0.33	12.8
5	(8, 24) (10, 19) (12, 15) (14, 13)	0	0.00	12.1
6	(8, 24)	0	0.00	13.2
7	(8, 24) (10, 20) (12, 16) (14, 13)	2/4	0.35	12.8
8	(8, 24) (10, 19) (12, 15) (14, 13)	0	0.00	12.8
9	(8, 24) (10, 19) (12, 15) (14, 13)	0	0.00	12.2
10	(8, 24) (10, 20) (12, 16)	2/3	0.47	12.3
Ave.		0.22	0.16	12.5

CHAPTER 6

LOCAL SEARCH WITH SPECIFIC CONSTRAINTS

In this chapter, the local search module of the proposed CPRSM is introduced. The local search module attempts to find a schedule that satisfies specific requirements imposed by either the precaster or the contractor. This is done by iteratively exploring the neighborhoods of available repaired schedules generated by the global search. Two local search algorithms are tested to implement the proposed local search mechanism, and the results of these tests are discussed in this chapter.

6.1 Overview of Local Search

Local search is a generally applicable approach for tackling combinatorial search and optimization problems. It has been successful in finding high quality solutions to a large number of hard combinatorial problems in a reasonable amount of computational time.

Generally speaking, a local search algorithm is based on the iterative exploration of neighborhoods of the current solution in order to improve it by some modification. Such modifications are predefined by a neighborhood structure (N), which results in a solution that differs only slightly from its originator. One can say that a neighboring solution is within the vicinity of its originator. An objective function (F) is needed for the local search in order to assess the quality of the solutions; this in turn drives the search towards good solutions in the search space (S). It is expected that a neighboring solution produces an objective function value that is of similar quality to

the original solution because they share a majority of solution characteristics. Since the chance to find an improved solution within a neighborhood is much higher than that in less correlated areas of the search space, local search algorithms concentrate the search within neighborhoods. Therefore, the local search only explore parts of a given search space as opposed to methods based on exhaustive search (Phan 2000).

Various problem specific local search algorithms have been developed based on the idea of neighborhoods. Finding efficient neighborhood structures that lead to high-quality performance can be viewed as one of the challenges of implementing local search. Following Aarts and Lenstra (1997) and Stützle (1999), a neighborhood structure can be defined as follows:

Let S be the set of feasible solutions for the problem, a neighborhood structure is a function $N : S \rightarrow 2^S$ that assigns to every $s \in S$ a set of neighbors $N(s) \subseteq S$. $N(s)$ is also called the neighborhood of s .

Typically, a neighborhood structure is not defined explicitly by enumerating the set of possible neighbors, but rather is defined implicitly by possible local changes that may be applied to a solution. The choice of an appropriate neighborhood structure often has to be done in a problem specific way. No general rules are available and each situation has to be considered separately. There are some considerations on desirable features of neighborhoods that should be addressed (Mattfeld 1996):

(1) Correlation: A neighbor solution should be highly correlated to its originator.

Ideally, a neighborhood $N(s)$ of s locates a neighboring solution s' that has similar characteristics as s .

(2) Improvement: Moving from a current solution s to its neighboring solutions, there should be a good chance of improving the originator in terms of the objective

function value.

- (3) Connectivity: It is desirable that there is a finite sequence of moves (worsening ones included) leading from an arbitrary solution to a global optimal one; otherwise, promising areas of the search space may be excluded from the search.
- (4) Size: The average size of the neighborhood $N(s)$ should be within reasonable bounds. On the one hand, a large neighborhood size may make the search for an improved neighboring solution become computationally prohibitive. On the other hand, a small number of neighbors may make it harder to find interesting solutions and risks a premature step with a poor quality solution.

Some of these considerations may conflict with each other and cannot be resolved theoretically. In order to develop appropriate neighborhood definitions, practical experience with applications and problem specific knowledge are needed.

A basic version of local search is *iterative improvement*. The general procedure of the iterative improvement algorithm that solves a minimization problem is shown in Figure 6.1. It starts with some initial solutions generated randomly or by some constructive heuristics, and searches its neighborhood for an improved solution according to the prescribed objective function. If such an improved solution is found, it replaces the current solution and the search continues; otherwise, the algorithm returns the current solution, which is then locally optimal. Iterative improvement can use either the *first-improvement* or *best-improvement* rule in determining which neighboring solution replaces the current one. In first-improvement, the current solution is replaced by the first improved solution found by the neighborhood search; whilst the current solution is replaced by the best improved solution in its neighborhood with best-improvement. Therefore, as the example of the simple iterative improvement algorithm shows, any local search algorithm has three basic

steps: (1) generation of a start solution; (2) generation of a neighboring solution; and (3) calculation and comparison of objective function values.

```

procedure Iterative Improvement ( $s \in S$ )
   $s := \text{InitialSolution}()$ ;
  begin
    repeat
       $s' = \text{modify}(s)$ ;
      if ( $F(s') < F(s)$ )
        then  $s := s'$ 
    until  $\forall s' \in N(s) : F(s') \geq F(s)$ 
    return  $s$ 
  end

```

Figure 6.1 Algorithmic skeleton of iterative improvement

6.2 Implementation of Local Search with Specific Constraints

The following sections discuss the implementation of local search in the proposed CPRSM, which includes the specification of constraints, the definition of an objective function and a neighborhood structure, the source of initial solutions, and the development of an efficient method for exploring the neighborhood.

6.2.1 Specific Constraints

As a result of the global search process with multiobjective optimization, multiple repaired schedules along a *Pareto* front have been identified. These schedules represent different degrees of trade-off between two conflicting objectives, namely F_D and F_I , for the production schedule. Associated with these schedules is the information on circumstances that lead to the schedule decision. That is, with identical values of the two objective functions, the schedules could be different in detail. For example, the L-U example tested in Chapter 5 is supposed to have three options for

the repaired schedules with the objective function vector of (1, 4). As shown in Table 6.1, each of these schedules has 1 unit of element that can not be delivered on time, but this element can be either E1, E2 or E3. In practice, it is possible that the global search only generates the first two kinds of the repaired schedules in the table. However, the decision maker may wish to find a schedule that satisfies the specific requirements in the third case. This requires an incremental exploration with the aim of finding the particular schedule with these specific requirements imposed.

Table 6.1 Different circumstances for repaired schedules

	F_D				F_I			
	E1	E2	E3	Total	E1	E2	E3	Total
1	1	0	0	1	2	2	0	4
2	0	1	0	1	1	2	1	4
3	0	0	1	1	1	2	1	4

Therefore, delivery and inventory levels, which are subject to optimization in the global search module, are treated as constraints in the local search module. Such objectives of deliveries and inventory levels as formulated in Equations (3.8) and (3.9) are modified so that specific numerical targets as specified by the precaster or the contractor are included. For example, the specific constraints on delivery and inventory level for the element type e can be formulated as follows:

(1) Specific constraint on delivery:

$$D_e^L \leq \sum_{t=1}^T (R_{t,e} - D_{t,e})^+ \leq D_e^U \quad (6.1)$$

where $(R_{t,e} - D_{t,e})^+ = \max(0, R_{t,e} - D_{t,e})$

The parameters D_e^L and D_e^U are the lower and upper bounds for number of element units that are delivered beyond their due dates.

(2) Specific constraint on inventory level:

$$S_e^L \leq \sum_{t=1}^T \{(S_{t,e} - S_e)^+ + (S_e' - S_{t,e})^+\} \leq S_e^U \quad (6.2)$$

where $(S_{t,e} - S_e)^+ = \max(0, S_{t,e} - S_e)$, $(S_e' - S_{t,e})^+ = \max(0, S_e' - S_{t,e})$

The parameters S_e^L and S_e^U are the lower and upper bounds for the number of element units that are out of a prescribed band of the inventory level.

When the values for the lower and upper bounds are equal, these constraints are expressed as equalities.

6.2.2 Objective Function

The addition of specific constraints in local search suggests trying to meet these constraints by modifying solutions that do not necessarily meet them. In order to guide the iterative repair process towards the particular one that can satisfy all these constraints, an appropriate objective function needs to be formulated.

In this research, an “error” function is first defined for each constraint to give an indication on how much the constraint is violated. That is, the error function returns the absolute distance of the current configuration to the specific target imposed. The objective function is then formulated as the sum of all the error functions for the specific constraints. Since the constraint violations considered in this research are all measured in terms of the number of element units involved, their corresponding error functions can be summed up directly without normalization. Therefore, the lower the objective function value of a solution, the greater the degree of satisfaction of the specific constraints imposed.

Given a set of C specific constraints, the objective function for the local search F_L is formulated as follows:

$$F_L = \sum_{i=1}^C f_i^{error} \quad (6.3)$$

where f_i^{error} represents the error function of the i -th constraint. Rather than minimizing the number of constraints that are not satisfied, this formulation of the objective function in terms of the number of element units involved in constraint violations makes the local search process more responsive to the degree of violation. Information on the degree of satisfaction for each of the constraints can be used to decide on the repair actions.

6.2.3 Initial Solutions

Repaired schedules generated by the global search module provide the initial solutions for the local search process. These schedules are not only complete as they involve all the precast components considered within the planning horizon, but also highly correlated to the specific schedule since they are all non-dominated according to the two objectives considered in the global search process. However, they may fail to meet specific constraints imposed during the local search phase.

A straightforward extension of local search, namely the multi-start local search (Aarts and Lenstra 1997; Congram *et al.* 2002), is adopted in this research to overcome a major drawback of local search, namely that it may stop at local optima that are of poor quality. This approach calls for running the search a number of times using different starting solutions. Therefore, the diversity of quality of the schedules generated by the global search, measured in terms of their deviations from the requirements imposed during the local search process, is desirable in the multi-start approach. When the search starts with a particular schedule fails in generating the specific schedule, another search can be initiated with a different schedule, thus

increasing the probability of success in finding the specific schedule.

6.2.4 Neighborhood Structure

For many combinatorial optimization problems whose solutions can be represented as sequences, partitions, or assignments, some type of k -exchange neighborhood structure ($k \geq 2$) is usually adopted, since it is both effective and easy to search. The k -exchange neighborhood contains all solutions that can be obtained by exchanging k elements in the sequence, partition, or assignment. Verifying local optimality for a k -exchange neighborhood requires $\Omega(n^k)$ time, where n is the total number of variables. As k increases, the computational effort required to search the neighborhood grows quickly, so that selecting larger values of k is often impractical.

In the scheduling problem, the simplest choice for the k value is 2, in which any two jobs irrespective of whether they are adjacent are exchanged. This kind of neighborhood structure is also called the swap neighborhood (Congram *et al.* 2002). For example, consider the sequence (A, B, C, D, E, F, G, H) in a schedule problem where there are eight jobs labeled from A to H. Thus, (A, F, C, D, E, B, G, H) is a neighbor by exchanging jobs B and F. This simplest form of neighborhood structure is adopted in this research.

As depicted in Figure 6.2, a two-dimensional integer array S is applied to represent the production schedule for precast building components. The rows represent molds numbered from 1 to M , and the columns represent workdays from 1 to T . The domain of the array $\{0, 1, 2, \dots, E\}$ is the set of precast element types to be produced. The element S_{mt} of the array therefore represents information on the production of mold m on day t : 0 denotes that the mold is idle, whereas other non-zero values denote the

production of a particular element type. Thus, the swap neighborhood is defined as the exchange of two elements S_{mt} and $S_{m't'}$ ($m \neq m'$ and $t \neq t'$) in the schedule array S , each of which represents the production of a different element type. An example of the swap neighborhood of the schedule array S is shown in Figure 6.2.

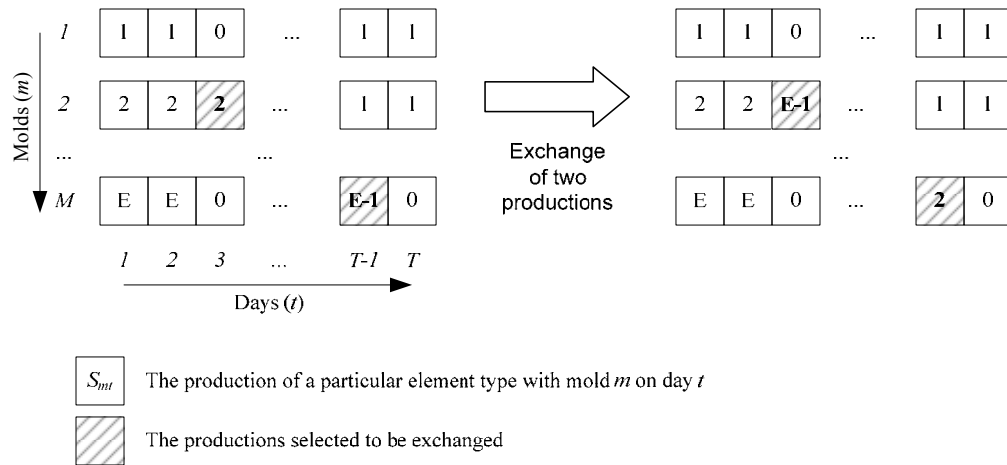


Figure 6.2 The swap neighborhood of the production schedule

6.2.5 Search Heuristics

How a given schedule is repaired certainly influences the efficiency of the iterative improvement technique. Repair methods differ in the amount of domain knowledge they exploit to modify a solution. One can consider a repair method as a generate-and-test process, where the generator takes as input a schedule and suggests possible modifications, and the tester then selects and performs one of the suggested modifications (Zweben *et al.* 1993). Thus, knowledge can be exploited in both the generator and the tester. For example, the generator incorporates constraint knowledge to greatly restrict the possible targets to be considered. In contrast, once a task is selected for repair, the tester selects the best modification that minimizes the number of remaining constraint violations. However, using knowledge is not free as computational overhead is incurred to evaluate and use repair knowledge. More

informed methods also tend to be more expensive. The following section describes some search heuristics used in this research.

6.2.5.1 Random search heuristic

In this heuristic, the pair of productions in the schedule that needs to be exchanged is decided at random. The new generated schedule is evaluated, and if no improvement is found, another modification is tried. Some experiments have shown that one can find good solutions with random techniques. Its success may be due to its simplicity since it does not require deep computational reasoning, nor a thorough search of all branches of the search tree (Dorn *et al.* 1996). The steps of the search procedure with the random search heuristic are described below:

Step 1: Given a set of constraints C_j and corresponding error functions f_j^{error} ($j = 1, 2, \dots, \rho$), and the objective function F_L ; select an initial schedule S^0 .

Step 2: Set $S \leftarrow S^0$.

Step 3: Search a neighboring schedule $S' \in N(S)$ by randomly exchanging a pair of production S_{mt} and $S_{m't'}$ ($m \neq m'$ and $t \neq t'$) in S , and calculate the objective function value $F_L(S')$.

Step 4: If $F_L(S') < F_L(S)$, set $S \leftarrow S'$; otherwise, go to Step 5.

Step 5: Check whether it meets any of the stopping criteria or not: (i) $F_L(S) = 0$ and (ii) a computing time bound is reached. If yes, stop the search and provide the current schedule S as the final solution; otherwise, go to Step 3.

6.2.5.2 Max-Min Conflicts heuristic

It seems reasonable that a successful iterative improvement technique applies a mixture of random and knowledge-based decisions to search the neighborhoods. The

Min-Conflicts heuristic (Minton *et al.* 1990) tries to repair the greatest constraint conflict. A system using Min-Conflicts exploits lookahead, which is quite effective at choosing the best repair. The original Min-Conflicts heuristic is as follows:

Given: A set of variables, a set of binary constraints, and an assignment specifying a value for each variable; two variables *conflict* if their values violate a constraint;

Procedure: Select a variable that is in conflict, and assign it a value that minimizes the number of conflicts. (Break ties randomly.)

The Min-Conflicts heuristic could be combined with other heuristics. For example, some researchers have considered a variation that uses “Max-Conflicts” as a variable ordering heuristic in conjunction with the Min-Conflicts value ordering heuristic (Zweben *et al.* 1994; Codognet and Diaz 2001). Instead of picking a variable randomly from the set of variables in conflict, the “Max-Conflicts” variation will randomly choose from the variables with the most-conflicts. The Min-Conflicts heuristic then selects the repair that minimizes the number of conflicts. This variation of Min-Conflicts heuristic is adapted in this research for the local search module, and is called the Max-Min Conflicts heuristic in the following sections.

The local search procedure with the Max-Min Conflicts heuristic is shown in Figure 6.3. The steps of the search procedure are described below:

Step 1: Given a set of constraints C_j and corresponding error functions f_j^{error}

($j = 1, 2, \dots, \rho$), and the objective function F_L ; select an initial schedule S^0 .

Step 2: Set $S \leftarrow S^0$.

Step 3: For each element type $e \in (1, 2, \dots, E)$, sum up all the constraint violations that it is involved with:

Step 3a: Set the element type counter $e = 1$ and the sum of constraint violations for each element type $F_L^{(e)} = 0$.

Step 3b: Set the constraint counter $j = 1$.

Step 3c: If element type e is involved in constraint C_j , let $F_L^{(e)} = F_L^{(e)} + f_j^{error}$; then increment j by one.

Step 3d: If $j \leq \rho$, go to Step 3c; otherwise, increment e by one and go to Step 3e.

Step 3e: If $e \leq E$, go to Step 3b; otherwise, return the sum of constraint violations for each element type $F_L^{(e)}$, ($e = 1, 2, \dots, E$).

Step 4: Identify the “culprit” element type k if $F_L^{(k)} = \max (F_L^{(e)} : e \in (1, 2, \dots, E))$. (Break ties randomly).

Step 5: Search all the neighboring schedules $S' \in N(S)$ by exchanging every possible pairs of productions in S which satisfy $S_{mt} = k$ and $S_{m't'} \neq k$ ($m \neq m'$ and $t \neq t'$), and calculate the objective function value $F_L(S')$.

Step 6: If $\exists S' \in N(S) : F_L(S') < F_L(S)$, select the best neighboring schedule S'_{best} for $F_L(S'_{best}) = \min (F_L(S') : S' \in N(S))$ and set $S \leftarrow S'_{best}$; otherwise, go to Step 7.

Step 7: Check whether it meets any of the stopping criteria or not: (i) $F_L(S) = 0$ and (ii) a computing time bound is reached. If yes, stop the search and provide the current schedule S as the final solution; otherwise, go to Step 3.

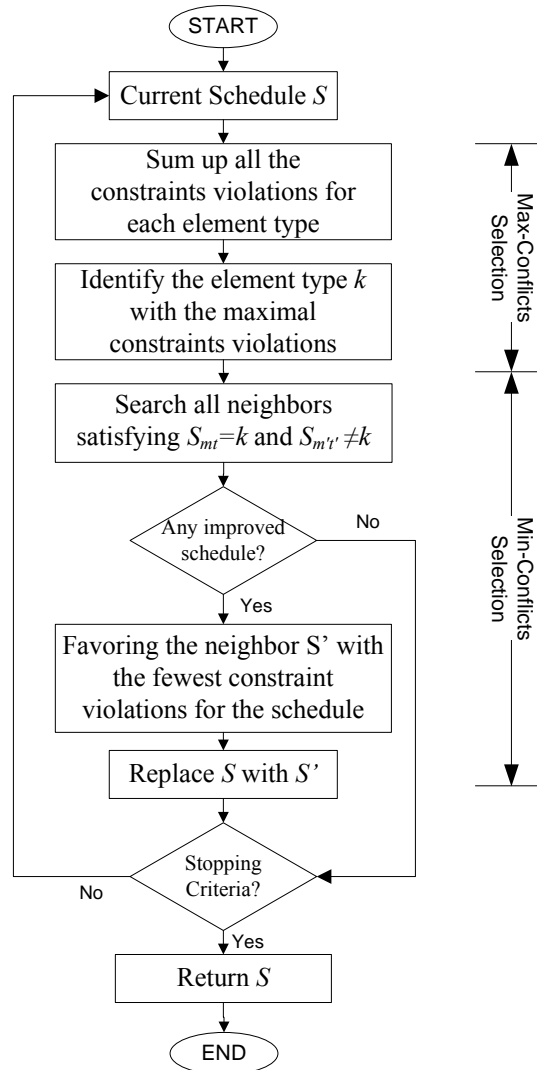


Figure 6.3 Search procedure of the Max-Min Conflicts heuristic

6.3 Case Study

6.3.1 Illustrative Examples

Two cases were tested in this study to verify the feasibility and applicability of the proposed local search algorithms. They were based on a particular NSGA run for the L-U example in the previous case study involving the global search module. In this NSGA run, alternative repaired schedules along the *Pareto* front provided several

options for the decision making. Detailed information associated with these schedules is shown in Table 6.2. Table 6.3 shows the details of the two cases used in this part of the study. Each case involving a particular schedule needs to be met by using local search. These two schedules provide one more option for the non-dominated schedules with the objective function vectors of (0, 7) and (4, 2) respectively.

Table 6.2 Information for available repaired schedules

No. of options	Number of units delivered late				Number of units out of the prescribed inventory band			
	E1	E2	E3	Total	E1	E2	E3	Total
1	0	0	0	0	3	3	1	7
2	1	0	0	1	2	2	0	4
3	0	1	0	1	1	2	1	4
4	0	0	1	1	1	2	1	4
5	0	2	1	3	2	1	0	3
6	1	2	0	3	1	1	1	3
7	1	2	1	4	1	1	0	2
8	2	2	0	4	0	1	1	2

Table 6.3 Specific constraints considered in two cases

Constraints	Specific Requirements	
	Case 1	Case 2
The number of units delivered late		
In total for all element types	=0	=4
Element type E1	=0	=0
Element type E2	=0	=2
Element type E3	=0	=2
The number of units out of the prescribed inventory band		
In total for all element types	=7	=2
Element type E1	=3	
Element type E2	=1	
Element type E3	=3	

Two search heuristics, namely the random search heuristic and the Max-Min Conflicts heuristic, were applied to search these two specific schedules. Eight repaired schedules were randomly selected from the solution set generated by the NSGA run as the initial solutions for the local search; each of them represents the available options

described in Table 6.2. For each combination of search heuristic and example, eight groups of experiments were executed by starting with each of the repaired schedules. Each group of experiments with the same initial solution consisted of three separate runs. All the experiments ran until (1) there were no outstanding constraint violations, i.e. an objective function value of 0; or (2) a 150-second CPU time bound was reached. All the experiments were implemented using MS Visual C++ on a MS Windows platform.

6.3.2 Results and Discussion

The results of the experiments performed on these two examples are discussed in the following sections.

6.3.2.1 Case 1

The search results by the random search heuristic and the Max-Min Conflicts heuristic with the eight selected initial solutions are shown in Figures 6.4 and 6.5 respectively. These figures display the step-by-step improvement in the objective function value with every acceptable modification to the current schedule against the CPU time. Information on the performance of the experiments starting with different initial solutions is shown in Table 6.4, such as the frequency of finding the specific schedule, the best and worst objective function value of the schedules found, and the fewest computing time and corresponding repair steps used to find the specific schedule.

For the random search heuristic, the experiments starting with seven out of the eight initial solutions found the specified schedule satisfying all the constraints imposed in Case 1 in at least one out of the three runs. As indicated in Table 6.4, the frequency of finding the specified schedule in the experiments with the initial solutions No. 5, 6

and 8 was 1, the experiments with the initial solution No.1 was 2, and the experiments with the initial solutions No. 2, 3 and 4 was 3. On the other hand, the experiments starting with the seventh initial solution did not find the specified one within the computing time bound of 150 seconds. The three runs generated the schedules with an objective function value ranging from 2 to 3. Therefore, the results show that the performance of the random search heuristic was highly variable not only in the experiments with different initial solutions, but also in the runs with the same initial solution. In contrast, except for the two runs involving the seventh initial solution, all the other runs with the Max-Min Conflicts heuristic successfully generated the specified schedule with an objective function value of 0. The results of the experiments with the Max-Min Conflicts heuristic show that this heuristic was consistent in generating the specified schedule considered in Case 1.

Table 6.4 Performance of search heuristics for Case 1

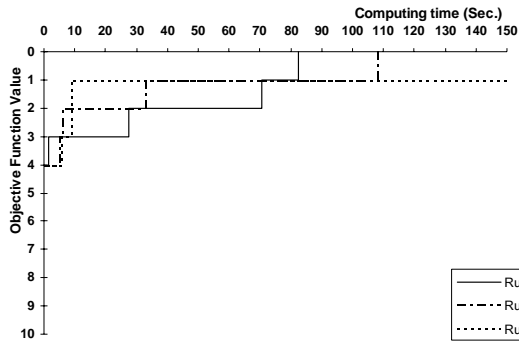
Initial solution		Random search heuristic					Max-Min Conflicts heuristic				
No	Obj. value	Fre.*	Final obj.		Repair steps ⁺	Computing times ⁺ (Seconds)	Fre.*	Final obj.		Repair steps ⁺	Computing times ⁺ (Seconds)
			Best	Worst (if any)				Best	Worst (if any)		
1	4	2	0	1	4	82.48	3	0	--	2	0.08
2	6	3	0	--	6	31.26	3	0	--	3	0.11
3	6	3	0	--	6	74.36	3	0	--	3	0.11
4	6	3	0	--	6	69.70	3	0	--	3	0.10
5	7	1	0	2	7	147.71	3	0	--	3	0.11
6	7	1	0	3	4	76.76	3	0	--	4	0.16
7	9	0	2	3	--	--	1	0	1	5	0.24
8	9	1	0	2	8	102.17	3	0	--	4	0.18
Sum[#]		14					22				

Note: * – The number out of the three runs made with the same initial solution that found the specified schedule.

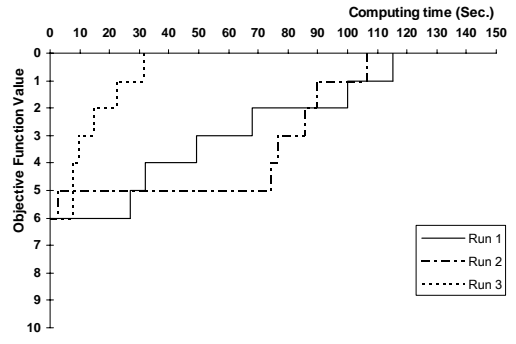
+ – The number of repair steps and computing time of the run that found the specified schedule in the least computing time.

– The number of all the runs with each heuristic that found the specified schedule.

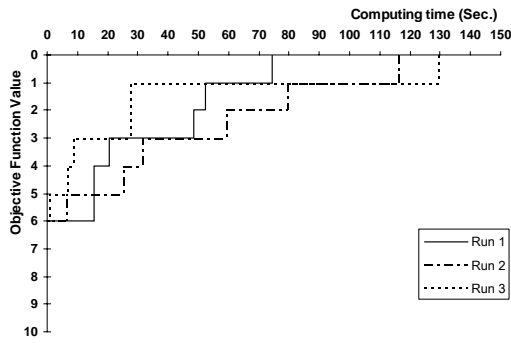
Therefore, over the 24 runs for each heuristic, the convergence frequency (Table 6.4) for the Max-Min Conflict heuristic (22) is much higher than that for the random search heuristic (14). In terms of computing time and repair steps used to find the specified schedule with the same initial solution, the results show that the Max-Min Conflicts heuristic was superior to the random search heuristic. For example, for the second initial solution, the former only required 3 repair steps and 0.11 seconds to find the specified schedule in the best case, whilst the latter required 6 repair steps and 31.26 seconds.



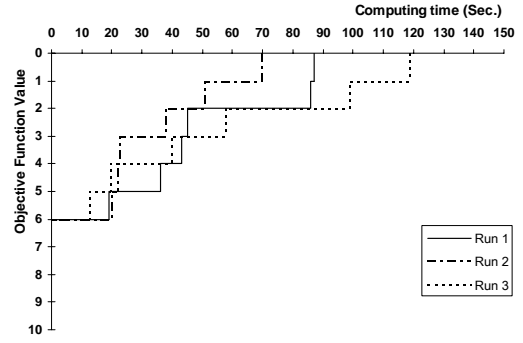
(1) Initial solution 1



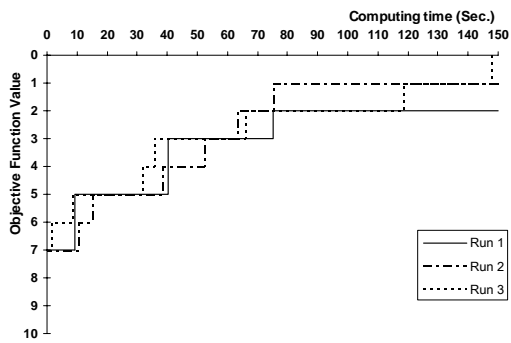
(2) Initial solution 2



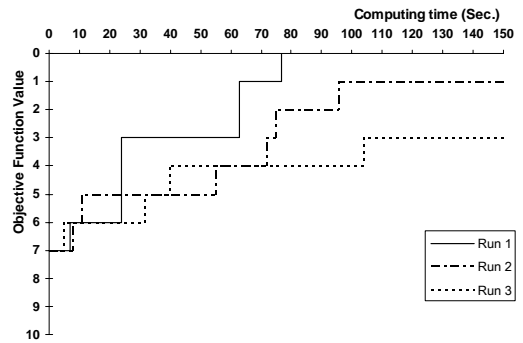
(3) Initial solution 3



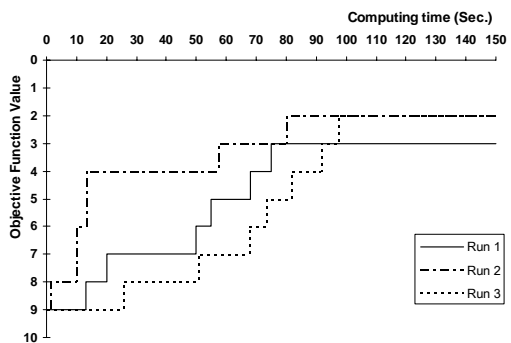
(4) Initial solution 4



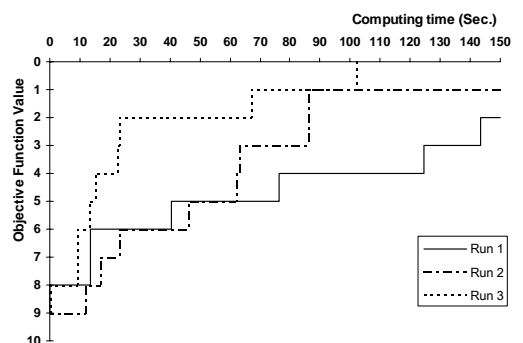
(5) Initial solution 5



(6) Initial solution 6



(7) Initial solution 7



(8) Initial solution 8

Figure 6.4 Results of the random search heuristic for Case 1

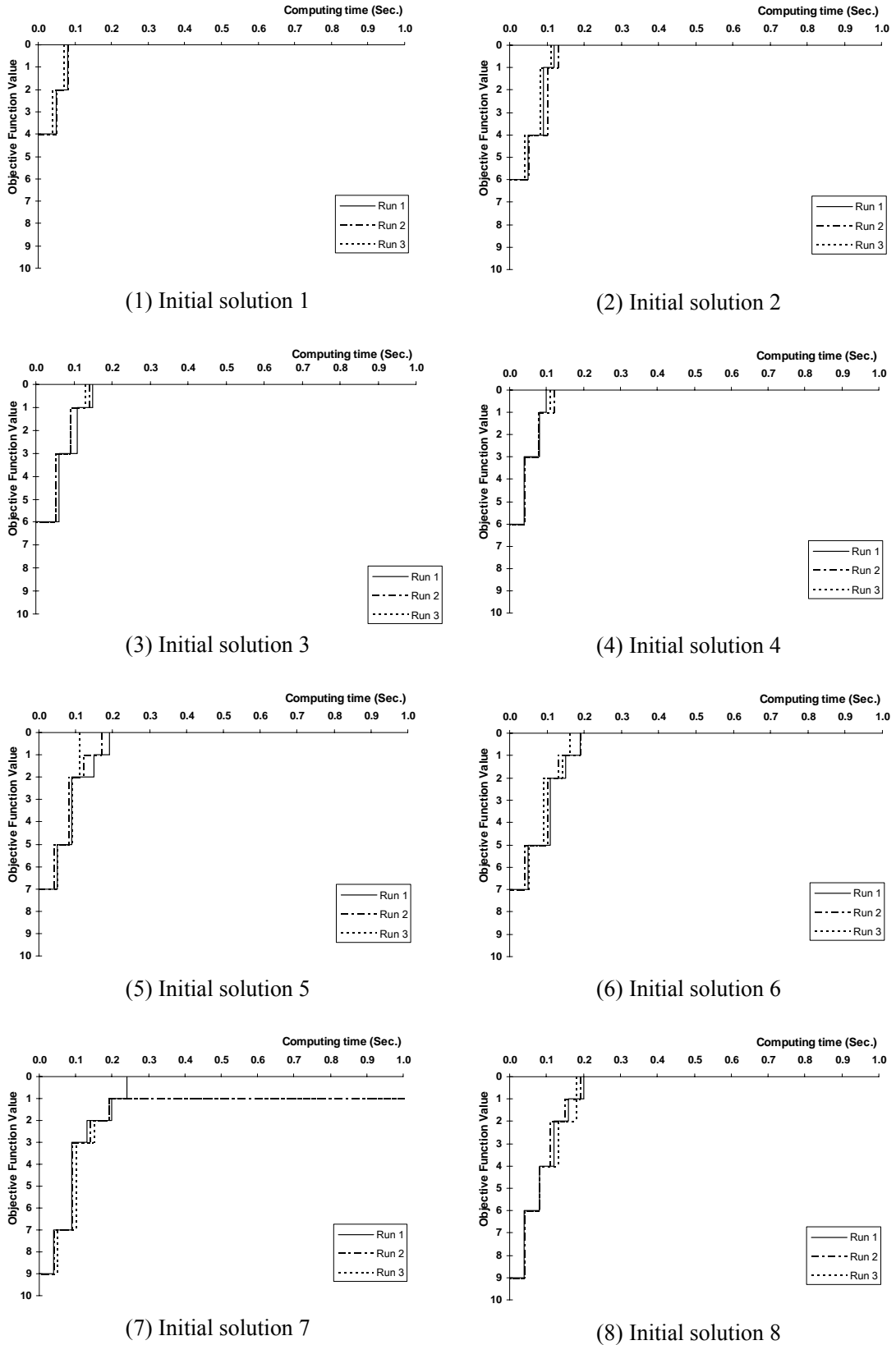


Figure 6.5 Results of the Max-Min Conflicts heuristic for Case 1

6.3.2.2 Case 2

Figures 6.6 and 6.7 display the search results with the eight initial solutions by the random search heuristic and the Max-Min Conflicts heuristic respectively. Table 6.5 shows the statistics on the performance of the heuristics in these experiments.

For the random search heuristic, the experiment involving the first five initial solutions generated the specified schedule considered in Case 2 in all the three runs, whilst the experiments starting with the sixth and eighth initial solutions converged to the specified one in one and two out of the three runs respectively. All of the schedules generated by the unsuccessful runs have an objective function of 1. Moreover, the random search heuristic failed to find the specified schedule starting with the seventh initial solution. All the three runs generated a schedule with an objective value of 1 at the end of 150 seconds. In contrast, all of the 24 runs with the Max-Min Conflicts heuristic successfully generated the specified schedule for the initial solutions tried.

These two heuristics required the same number of repair steps in generating the specified schedule with the initial solutions No. 3 and 5; however, the Max-Min Conflicts heuristic required fewer steps than the random search heuristic in the experiments involving other initial solutions. Moreover, the Max-Min Conflicts heuristic used less than 0.2 seconds to find the specified schedule in the experiments involving different initial solutions, whilst the random search heuristic used a computing time ranging from 4.17 to 80.84 seconds even in the fastest runs. Therefore, the results of experiments for Case 2 again show that the Max-Min Conflicts heuristic performed much better than the random search heuristic.

Table 6.5 Performance of search heuristics for Case 2

Initial solution		Random search heuristic					Max-Min Conflicts heuristic				
No	Obj. value	Fre.*	Final obj.		Repair steps ⁺	Computing times ⁺ (Seconds)	Fre.*	Final obj.		Repair steps ⁺	Computing times ⁺ (Seconds)
			Best	Worst (if any)				Best	Worst (if any)		
1	9	3	0	--	8	64.57	3	0	--	5	0.18
2	5	3	0	--	4	31.47	3	0	--	3	0.09
3	7	3	0	--	4	50.78	3	0	--	4	0.13
4	5	3	0	--	4	26.90	3	0	--	3	0.10
5	2	3	0	--	2	4.17	3	0	--	2	0.07
6	4	1	0	1	4	64.49	3	0	--	3	0.09
7	2	0	1	--	--	--	3	0	--	1	0.04
8	4	2	0	1	4	80.84	3	0	--	2	0.08
	Sum[#]	18					24				

Note: * – The number out of the three runs made with the same initial solution that found the specified schedule.

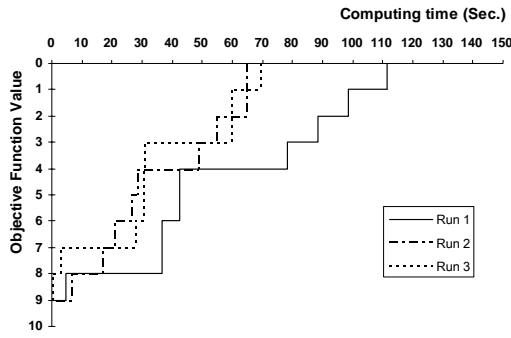
+ – The number of repair steps and computing time of the run that found the specified schedule in the least computing time.

– The number of all the runs with each heuristic that found the specified schedule.

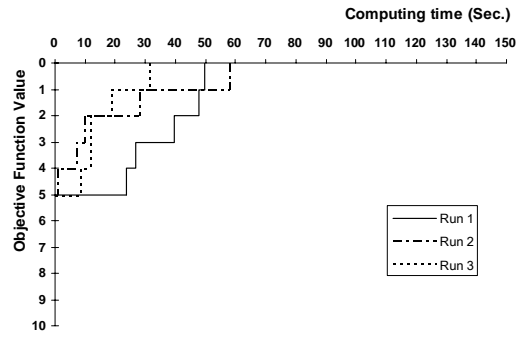
6.3.2.3 Summary

In summary, the results indicate that the Max-Min Conflicts heuristic performed better than the random search heuristic in both cases. It is apparent that the computing time for finding specified schedules with the Max-Min Conflicts heuristic was much less than with the random search heuristic. This is attributed to the difference between the ways these two approaches use in searching among the neighbors. Since neighborhoods are restricted to the element types involved in the greatest constraint violation for the Max-Min Conflicts heuristic, there are fewer candidate modifications to explore on average than for the random search heuristic. The Max-Min Conflicts heuristic searches all the possible neighbors deterministically and always selects the best modification operator with the minimal constraint violation. In contrast, it takes time to find an acceptable modification within a large neighborhood with the random search approach as its search is random. Furthermore, the modification is accepted no matter how much/little improvement it makes to the objective function value. This

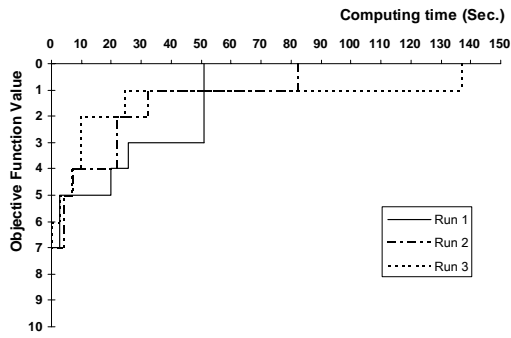
also explains why the number of repair steps taken by the Min-Max Conflicts heuristic is less than that in the random search heuristic for the same initial solution. Consequently, it is computationally less expensive to perform this form of lookahead for the problem considered in the local search module.



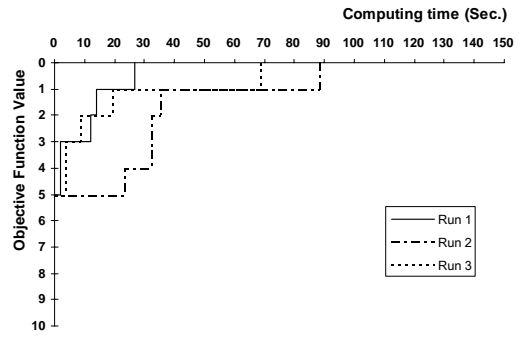
(1) Initial solution 1



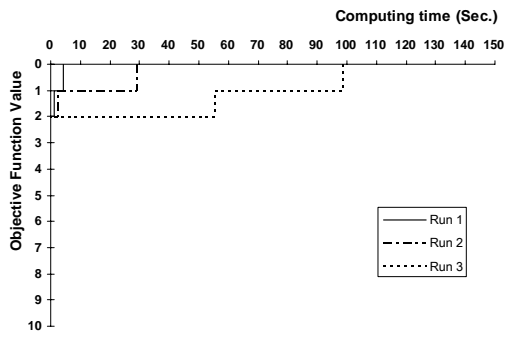
(2) Initial solution 2



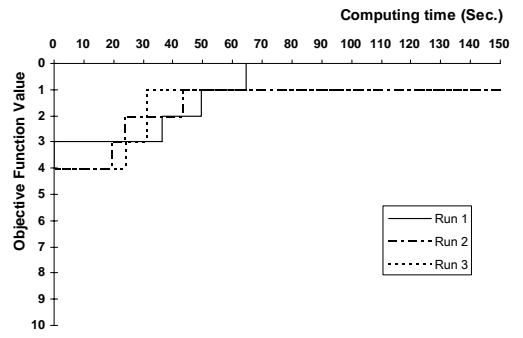
(3) Initial solution 3



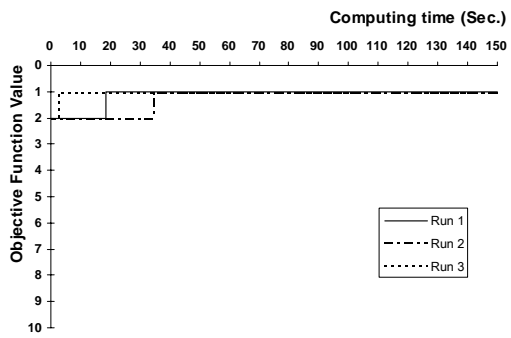
(4) Initial solution 4



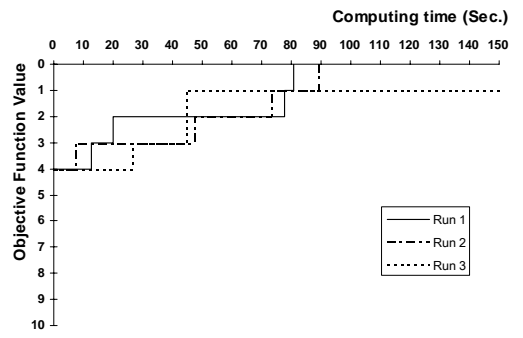
(5) Initial solution 5



(7) Initial solution 7

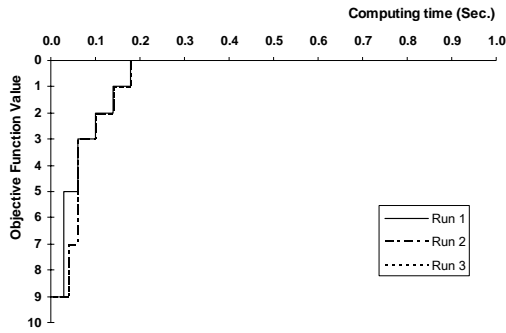


(7) Initial solution 7

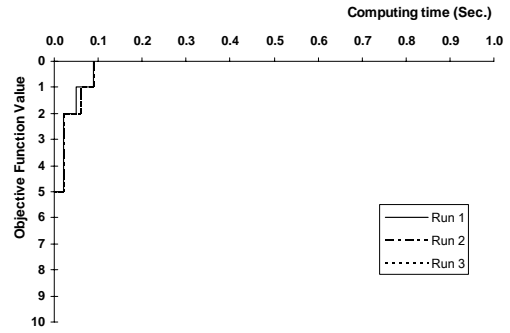


(8) Initial solution 8

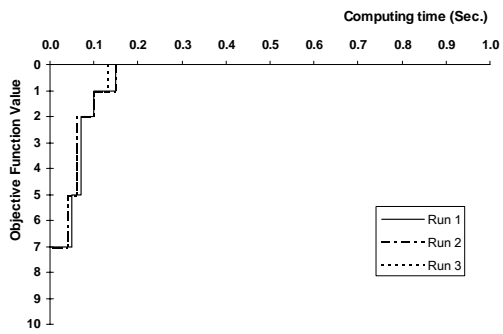
Figure 6.6 Results of the random search heuristic for Case 2



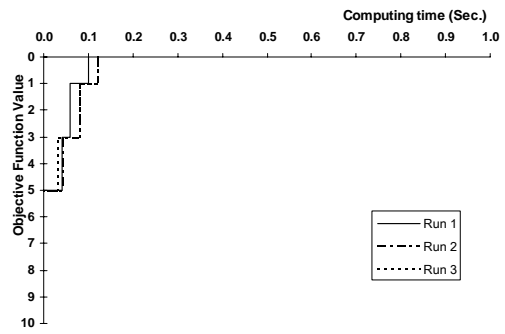
(1) Initial solution 1



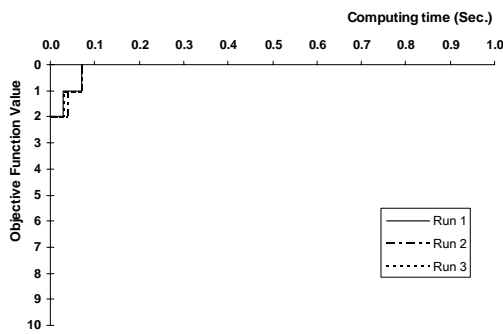
(2) Initial solution 2



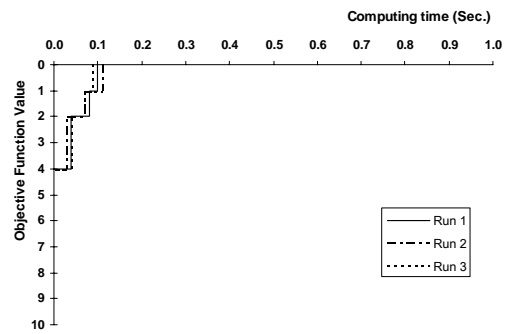
(3) Initial solution 3



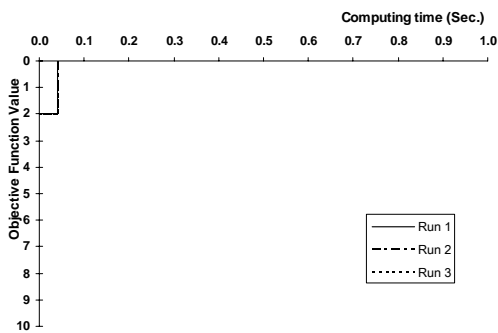
(4) Initial solution 4



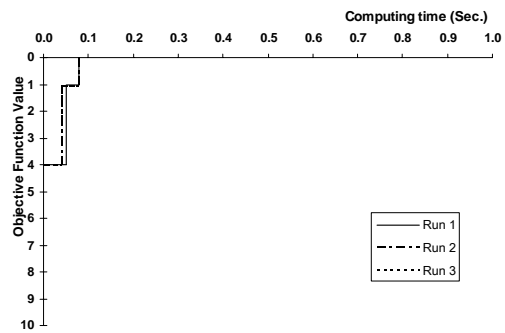
(5) Initial solution 5



(6) Initial solution 6



(7) Initial solution 7



(8) Initial solution 8

Figure 6.7 Results of the Max-Min Conflicts heuristic for Case 2

CHAPTER 7

CONCLUSIONS AND RECOMMENDATIONS

In this chapter, the work carried out in this research and the results obtained are first summarized; this is followed by a discussion of the limitations of the current research and recommendations for future research work.

7.1 Conclusions

The main contribution of this research is to develop a reactive scheduling model for precast production to complement earlier models that did predictive scheduling. This model recognizes the existence of schedule disturbances and the need to adapt the existing schedule in a practical way. This is achieved by using a repair-based strategy. The research also identifies effective global and local search methods to effect the repair-based strategy. In global search, the generation of alternative repaired schedules along the *Pareto* front using NSGA can be improved by including the elitist strategy, especially with the clustering selection that was created in the course of this research. This innovation helps to improve diversity of solutions along the *Pareto* front. On the other hand, the effectiveness and efficiency of local search can be improved by using the Min-Max Conflicts heuristic introduced in this research. This heuristic is capable of finding particular schedules with desired characteristics by exploiting domain knowledge associated with the specific constraints imposed on the local search. More specific conclusions about the model and search methods are highlighted in the following sections.

7.1.1 Development of a Coordinated Production Reactive Scheduling Model

A coordinated production reactive scheduling model (CPRSM) has been proposed in this research to address the need for repairing the current production schedule when disturbances arrive. This proposed CPRSM is based on a mathematical formulation of the precast production rescheduling problem, as well as an algorithmic solution procedure that incorporate advanced AI methods. It consists of four key elements for rescheduling, namely disturbance detection, global search with multiobjective optimization, local search with specific constraints, and ranking of outcomes for negotiation.

In the global search module, the precast production rescheduling problem is formulated as a true multiobjective optimization problem with quantitative evaluation criteria meaningful to both the precaster and the contractor. As the core of the proposed CPRSM, this module (1) repairs the existing schedule by resolving schedule disturbances with proper sequence and suitable heuristics, and (2) provides alternative repaired schedules that represent different degrees of trade-off between the multiple objectives considered. This module offers a more methodical and systematic approach of rescheduling production compared to that used by precasters in actual practice.

The local search module is developed to enhance the proposed CPRSM in case minimal adjustments are needed for the alternatives provided by the global search module. A local search mechanism is proposed, in which the schedules satisfying specific constraints imposed by either the precaster or the contractor are obtained by exploring the neighborhoods of available repaired schedules on the *Pareto* front. The local search module is useful in actual practice as it acts as a complementary search

process in the proposed CPRSM.

These two search processes provide a comprehensive means of searching for alternative repaired schedules in the rescheduling problem. These schedules form a valuable basis for the negotiation of a new delivery schedule between the precaster and the contractor. It is hoped that a final compromise solution can be found among the alternatives returned by the rescheduling process.

7.1.2 Generation of Repaired Schedules along a *Pareto* Front

Several multiobjective genetic algorithms were applied to generate alternative repaired schedules along a *Pareto* front defined by the multiple objectives considered in the global search module. The selected algorithms include the weighted sum genetic algorithm, non-dominated sorting genetic algorithm (NSGA) and two variations of NSGA incorporating different elitist strategies, namely NSGA-ESI and NSGA-ESII.

Three examples involving production schedules for precast building components were used to test the algorithms in the case study. These represent low, medium and high levels of mold utilization rates in the precast factory. The weighted sum genetic algorithm, in which varying values of the weights for the objectives were considered, is straightforward in application for the purpose of generating non-dominated solutions. However, this kind of algorithm has some shortcomings compared to the other three *Pareto*-based algorithms. The most significant disadvantage of this kind of algorithm is its sensitivity to the shape of the trade-off surface. The trial revealed that a non-dominated solution in the concave portion of the trade-off surface could not be found by the weighted sum GA in the L-U example. Although this algorithm

generated the optimal *Pareto* front in the M-U and H-U examples, it required more effort and computing time to do so. In contrast, the results for the three *Pareto*-based algorithms show that they are efficient in generating more than one non-dominated solution in their solution sets without the need for varying weights for the objectives. Moreover, the shape of the trade-off surface does not affect the ability of these algorithms in finding the non-dominated solutions no matter whether the surface is convex or non-convex.

The computational results of NSGA, NSGA-ESI and NSGA-ESII show that the performance of these *Pareto*-based algorithms was affected significantly with an increase in the utilization rate of the molds. However, the solution sets produced by NSGA-ESII were better than those obtained by NSGA and NSGA-ESI for all the examples. More significantly in the H-U example, half of the 10 NSGA-ESII runs converged to the optimal *Pareto* front, whilst none of the NSGA-ESI and NSGA runs were able to do so. The results indicate that the proposed elitist strategy is able to improve the performance of NSGA.

7.1.3 Exploration of Schedules with Specific Constraints

A local search process has been developed to support the incremental exploration for specific schedules. This local search uses the degree of satisfaction of specific constraints as the objective function. Starting with the alternative repaired schedules generated in the global search module, the local search process adopts the simplest form of a 2-exchange neighborhood structure and iteratively accepts modifications that improve the value of the objective function until the stopping criteria is met. Two search heuristics, namely the random search heuristic and the Max-Min Conflicts heuristic, were tried out in the search. A fundamental difference between these two

heuristics is the way in which the domain knowledge of specific constraints is applied in searching the neighborhood and accepting schedule changes. Experiments with these two heuristics were conducted on two cases. The results show that the Max-Min Conflicts heuristic performed better than the random search heuristic. Starting with the same initial solution, the Max-Min Conflicts heuristic was consistent in successfully finding the specified schedules, and required fewer repair steps and computing time than the random search heuristic.

7.2 Limitations of the Research

This research focused on generating alternative repaired schedules as a result of schedule disturbances. A reactive scheduling model for precast production has been developed and approved feasible. However, several limitations still exist in the proposed model.

Firstly, it is assumed that the number of molds (the critical resource considered in the model) is known in advance and remains constant during the planning horizon. However, this might not be the case in practice, since optimizing the mold groups and mold numbers are real concerns for the precaster. Besides the molds, other important resources such as the gantry crane, skilled workers and production space, are also assumed to be always available and enough to implement the production schedule. However, incorporating constraints related to the availability and capacity of these resources would make the proposed reactive scheduling model closer to the actual situation.

Secondly, in order to facilitate schedule coordination between the precaster and the

contractor, the present research focuses on optimizing two objectives simultaneously in the global search module. Although these two objectives represent the respective concerns of the precaster and the contractor, either party may consider other evaluation criteria in reality.

Thirdly, the scope of this research does not include the issue of how to rank alternative repaired schedules presented and reach a final compromise between the precaster and the contractor. This issue is necessary since the final aim of schedule coordination is to agree on a compromise schedule between these two parties before the precast production continues.

Fourthly, this research focuses on production rescheduling in precast factories. In reality, the contractor is also likely to review the construction schedule, and possibly, even adjust the construction schedule in the event of disruptions. Moreover, the values for a time buffer and a minimal inventory buffer need to be investigated. If these buffers are set too large, it would be wasteful; on the other hand, if these buffers are set too small, flexibility in responding to schedule disturbances is reduced.

Finally, the emphases of this research are the modeling of the rescheduling problem and the development of appropriate methods to solve it. Development of input and output interfaces that facilitate real life application in the rescheduling process is not considered in the present research.

7.3 Recommendations for Future Research

Based on the above discussions, the following issues are recommended for future

research work.

Firstly, development of a high-level resource replanning module is suggested to complement the proposed production rescheduling model. It could be used to consider issues, such as adjustment of quantities for different resources, resource assignment between different projects carried out in the factory, *etc.* Therefore, besides constraints on molds considered in this research, constraints related to other important resources (e.g. gantry crane and skilled workers) and related costs can be considered in the future development of the model.

Secondly, in order to meet the requirements of decision makers under different circumstances, more objective/evaluation functions can be included in the model. Practical considerations, such as the utilization level for different resources, the mold changeover cost, and the makespan for the production, are suggested for inclusion in future development.

Thirdly, an outcome ranking mechanism is recommended by incorporating negotiation techniques based on game theory. The precaster and the contractor can reach a “win-win” solution from alternatives presented by appropriately expressing their preferences with quantitative methods, such as converting their preferences into a real number with a suitable value function and plotting indifference curves for various degrees of the preferences.

Fourthly, a study of how the contractor responds to schedule disturbances is suggested for future research work. For example, if the adjustment of the construction schedule can not be done, the contractor can be more flexible in dealing with unexpected events by adjusting the time buffer between the delivery schedule and the construction schedule, as well as increasing the inventory buffer for precast components on the

site.

Finally, a friendly user interface would be beneficial to the application of the model in the industry. The input interface can facilitate users in entering the required data, such as disturbance characteristics, specific constraints, decision preferences, *etc.* The output interface can present a number of different alternative repaired schedules as generic Gantt charts and tables for easy reference.

REFERENCES

- Aarts, E. H. L. and Lenstra, J. K. (1997). Introduction. In Aarts, E. and Lenstra, J. K. (Eds.), *Local Search in Combinatorial Optimization*. pp. 1-17. New York: Wiley.
- Anderson, M. B. and Lawrence, W. R. (1996). Launch conditions and aerodynamic data extraction by an elitist pareto genetic algorithm. In *AIAA Atmospheric Flight Mechanics Conference*, San Diego, California. AIAA Paper 96-3361.
- Aytug, H., Lawley, M. A., McKay, K., Mohan, S. and Uzsoy, R. (2005). Executing production schedules in the face of uncertainties: a review and some future directions. *European Journal of Operational Research*, 161(1), pp. 86-110.
- Bagchi, T. P. (1999). *Multiobjective Scheduling by Genetic Algorithms*. Boston: Kluwer Academic Publishers.
- Bagchi, T. P. (2001). Pareto-optimal solutions for multi-objective production scheduling problems. In Zitzler, E. *et al.* (Eds.), *Proceedings of the First International Conference on Evolutionary Multi-Criterion Optimization (EMO 2001)*, pp. 458-471. Springer-Verlag. Lecture Notes in Computer Science No. 1993.
- Baker, K. (1974). *Introduction to Sequencing and Scheduling*. New York: John Wiley & Sons.
- Balbontín-Bravo, E. (1998). Simulation of large precast operations. In *Proceedings of the 1998 Winter Simulation Conference*, Vol.2, pp. 1311-1317.
- Bean, J. C. (1994). Genetic algorithms and random keys for sequencing and optimization. *ORSA Journal on Computing*, 6(2), pp. 154-160.
- Bierwirth, C. and Mattfeld, D. C. (1999). Production scheduling and rescheduling with genetic algorithms. *Evolutionary Computation*, 7(1), pp. 1-17.
- Brizuela, C. and Aceves, R. (2003). Experimental genetic operators analysis for the multi-objective permutation flowshop. In Fonseca, C. M. *et al.* (Eds.), *Proceedings of the Second International Conference on Evolutionary Multi-Criterion Optimization (EMO 2003)*, pp. 578-592. Springer-Verlag. Lecture Notes in Computer Science No. 2632.
- Brizuela, C., Sannomiya, N., and Zhao, Y. (2001). Multi-objective flow-shop: preliminary results. In Zitzler, E. *et al.* (Eds.), *Proceedings of the First International Conference on Evolutionary Multi-Criterion Optimization (EMO 2001)*, pp. 443-457. Springer-Verlag. Lecture Notes in Computer Science No. 1993.
- Brown, D. E., Marin, J. A., and Scherer, W. T. (1995). A survey of intelligent scheduling systems. In Brown, D. E. and Scherer, W. T. (Eds.), *Intelligent Scheduling Systems*, pp. 1-40. Boston: Kluwer Academic Publishers.

- Carlyle, W. M., Kim, B., Fowler, J. W., and Gel, E. S. (2001). Comparison of multiple objective genetic algorithms for parallel machine scheduling problems. In Zitzler, E. et al. (Eds.), *Proceedings of the First International Conference on Evolutionary Multi-Criterion Optimization (EMO 2001)*, pp. 472-485. Springer-Verlag. Lecture Notes in Computer Science No. 1993.
- Chan, W. T. and Hu, H. (1998). Process scheduling using genetic algorithms for construction industry. In *Proceedings of the 3rd International Conference on Management*, CHEP and Springer-verlag, 25-28 July 1998, Shanghai, China.
- Chan, W. T. and Hu, H. (2000). Precast production scheduling with genetic algorithms. In *Proceedings of Congress on Evolutionary Computation 2000*, pp.1087-1094, IEEE Press, 16-19 July 2000, San Diego, USA.
- Chan, W. T. and Hu, H. (2001). An application of genetic algorithms to precast production scheduling. *Computers and Structures*, Vol.79, pp.1605-1616.
- Chan, W. T. and Hu, H. (2002a). Production scheduling for precast plants using a flow shop sequencing model. *Journal of Computing in Civil Engineering*, 16(3), pp. 165-174.
- Chan, W. T. and Hu, H. (2002b). Constraint programming approach to precast production scheduling. *Journal of Construction Engineering and Management*, 128(6), pp. 513-521.
- Chan, W. T., Chua, D. K. H., and Xiong, L. (1999). Collaborative scheduling over the internet. *Computer-Aided Civil and Infrastructure Engineering*, Vol. 14, pp. 15-24.
- Chionh, C. K. (1999). The role of HDB in preparing Singapore's construction industry for the challenges of the new millennium. In *Construction Industry Development in the New Millennium: Proceedings of the 2nd International Conference on Construction Industry Development, and the 1st Conference of CIB TG 29 on Construction in Developing Countries*, 27-29 October, The Pan Pacific, Singapore.
- Chun, H. W. and Wong, R. Y. M. (2003). N* - an agent-based negotiation algorithm for dynamic scheduling and rescheduling. *Advanced Engineering Informatics*, 17(1), pp. 1-22.
- CIDB Construction Productivity Taskforce (1992). Raising Singapore's Construction Productivity: CIDB Construction Productivity Taskforce Report 1992. Singapore: Construction Industry Development Board.
- Codognet, P. and Diaz, D. (2001). Yet another local search method for constraint solving. In Steinhöfel, K. (Ed.), *Proceedings of the 1st International Symposium on Stochastic Algorithms: Foundations and Applications*, pp. 73-90. Berlin: Springer Verlag. Lecture Notes in Computer Science No. 2246.
- Coello, C. A. C. (2000). An updated survey of GA-based multiobjective optimization techniques. *ACM Computing Surveys*, 32(2), pp. 109-143.

- Coello, C. A. C. (2001). A short tutorial on evolutionary multiobjective optimization. In Zitzler, E. *et al.* (Eds.), *Proceedings of the First International Conference on Evolutionary Multi-Criterion Optimization (EMO 2001)*, pp. 21-40. Springer-Verlag. Lecture Notes in Computer Science No. 1993.
- Coello, C. A. C., Van Veldhuizen, D. A., and Lamont, G. B. (2002). *Evolutionary Algorithms for Solving Multi-objective Problems*. New York: Kluwer Academic.
- Collette, Y. and Siarry, P. (2003). *Mutiobjective Optimization : Principles and Case Studies*. Berlin; New York: Springer.
- Cochran, J. K., Horng, S. M., and Fowler, J. W. (2000). *A multi-population genetic algorithm to solve multi-objective scheduling problems for parallel machines*. Working Paper.
- Congram, R. K., Potts, C. N., and van de Velde, S. L. (2002). An iterated dynasearch algorithm for the single-machine total weighted tardiness scheduling problem. *Inforns Journal on Computing*, 14(1), pp. 52-67.
- Cui, X., Li, M., and Fang, T. (2001). Study of population diversity of multiobjective evolutionary algorithm based on immune and entropy principles. In *Proceedings of the Congress on Evolutionary Computation 2001*, Vol. 2, pp. 1316-1321. Piscataway, New Jersey, IEEE.
- Dandy, G. C. and Engelhardt, M. (2001). Optimal scheduling of water pipe replacement using genetic algorithms. *Journal of Water Resource Plan management*, 127(4), pp. 214-223.
- Davis, L. (1991). *Handbook of Genetic Algorithms*, New York: Van Nostrand Reinhold.
- Dawood, N. (1994). Developing a production management modeling approach for precast concrete building products. *Construction Management and Economics*, Vol.12, pp. 393-412.
- Dawood, N. (1995). An integrated knowledge-based/simulation approach to production planning: an application to the pre-cast industry. *Construction Management and Economics*, Vol.13, pp. 53-64.
- Dawood, N. (1996). A simulation model for eliciting scheduling knowledge: an application to the precast manufacturing process. *Advances in Engineering Software*, 25(2-3), pp. 215-223.
- Dawood, N. (1999). A proposed system for integrating design and production in the precast building industry. *International Journal of Construction Information Technology*, 7(1), pp.73-83.
- Dawood, N. and Neale, R. H. (1993). A capacity planning Model for precast concrete building products. *Building and Environment*, 28(1), pp. 81-95.

- Dawood, N. and Smith, M. (1996). Validation of an automated capacity planning in the concrete building materials industry using an industrial case study. *Building and Environment*, 31(2), pp. 129-144.
- Deb, K. (2001). *Multi-Objective Optimization Using Evolutionary Algorithms*. New York: John Wiley & Sons.
- Deb, K., Agrawal, S., Pratab, A. and Meyarivan, T. (2000). A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In Schoenauer, M. *et al.* (Eds.), *Proceedings of the Parallel Problem Solving from Nature VI Conference*, pp. 849-858, Paris, France. Springer. Lecture Notes in Computer Science No. 1917.
- Deb, K. and Goldberg, D. E. (1989). An investigation of niche and species formation in genetic function optimization. In Schaffer, J. D. (Ed.), *Proceedings of the Third International Conference on Genetic Algorithms*, San Mateo, CA, pp. 42-50. Morgan Kaufmann.
- De Jong, K. A. (1975). *An analysis of the behavior of a class of genetic adaptive systems*. Ph. D. Thesis. University of Michigan.
- Dorn, J., Girsch, M., Skele, G., and Slany, W. (1996). Comparison of iterative improvement techniques for schedule optimization. *European Journal of Operational Research*, Vol. 94, pp. 349-361.
- Fang, J. and Xi, Y. G. (1997). A rolling horizon job shop rescheduling strategy in the dynamic environment. *International Journal of Advanced Manufacturing Technology*, Vol.13, pp. 227-232.
- Fonseca, C. M. and Fleming, P. J. (1993). Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. In Forest, S. (Ed.), *Proceedings of the Fifth International Conference on Genetic Algorithms, San Mateo, CA*, pp.416-423. Morgan Kaufmann.
- Fonseca, C. M. and Fleming, P. J. (1995). An overview of evolutionary algorithms in multiobjective optimization. *Evolutionary Computation*, 3(1), pp. 1-16.
- Fonseca, C. M. and Fleming, P. J. (1998). Multiobjective optimization and multiple constraint handling with evolutionary algorithms: I. A unified formulation. *IEEE Transaction on Systems, Man and Cybernetics, Part A*. 28(1). pp. 26-37.
- Fox, M. S. and Smith, S. F. (1984). ISIS—a knowledge based system for factory scheduling. *Expert Systems*, Vol. 1, pp. 25-49.
- Fwa, T. F., Chan, W. T., and Hoque, K. Z. (2000). Multiobjective optimization for pavement maintenance programming. *Journal of Transportation Engineering*, 126(5), pp. 367-374.
- Garetti, M. and Taisch, M. (1995). Using neural networks for reactive scheduling. In Kerr, R. and Szelke, E. (Eds.), *Artificial Intelligence in Reactive Scheduling*, pp. 146-155. London: Chapman and Hall.

- Gembicki, F. W. (1974). *Vector Optimization for Control with Performance and Parameter Sensitivity Indices*. Ph. D thesis, Case Western Reserve University, Cleveland, Ohio.
- Gen, M. and Cheng, R. (1997). *Genetic Algorithms and Engineering Design*. New York: John Wiley & Sons.
- Goicoechea, A., Hansen, D. R., and Duckstein, L. (1982). *Multiobjective Analysis with Engineering and Business Applications*. New York: John Wiley and Sons.
- Goldberg, D. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*, Reading: Addison-Wesley.
- Goldberg, D. E. and Richardson, J. (1987). Genetic algorithms with sharing for multimodal function optimization. In Grefenstette, J. J. (Ed.), *Genetic Algorithms and their Applications: Proceedings of the Second International Conference on Genetic Algorithms*, Hillsdale, NJ, pp. 41-49. Lawrence Erlbaum.
- Graves, S. C. (1981). A review of production scheduling. *Operations Research*, 29(4), pp. 646-675.
- Hajela, P. and Lin, C. Y. (1992). Genetic search strategies in multicriterion optimal design. *Structural Optimization*. Vol.4, pp. 99-107.
- Halpin, D. W. and Riggs, L. S. (1992). *Planning and Analysis of Construction Operations*. New York: Wiley.
- Horn, J. (1997). Multicriteria decision making. In Bäck, T. *et al.* (Eds.), *Handbook of Evolutionary Computation*, pp. F1.9:1-15. Bristol: Institute of Physics Publishing and New York: Oxford University Press.
- Horn, J., Nafpliotis, N., and Goldberg, D. E. (1994). A niched pareto genetic algorithm for multiobjective optimization. *IEEE World Congress on Computational Intelligence*, Vol.1, pp. 82-87.
- Hu, H. (2000). *Production Scheduling for Precast Plants*. Ph.D. thesis. National University of Singapore.
- Hwang, C. L. and Masud, A. S. M. (1979). *Multiple Objective Decision Making – Methods and Applications: a State-of-the-art Survey*. Berlin: Springer-Verlag.
- Ishibuchi, H. and Murata, T. (1996). Multi-objective genetic local search algorithm. In *Proceedings of 1996 IEEE International Conference on Evolutionary Computation*, pp. 119-124. Piscataway, New Jersey. IEEE.
- Ishibuchi, H. and Murata, T. (1998). Multi-objective genetic local search algorithm and its application to flowshop scheduling. *IEEE Transactions on Systems, Man and Cybernetics*, 28(3), pp. 392-403.
- Jan, A., Yamamoto, M., and Ohuchi, A. (2000). Evolutionary algorithms for nurse scheduling problem. In *2000 Congress on Evolutionary Computation*, Vol.1, pp. 196-203. Piscataway, New Jersey. IEEE.

- Keeney, R. L. and H. Raiffa (1976). *Decisions with Multiple Objectives: Preferences and Value Trade-offs*. New York: John Wiley and Sons.
- Kim, C. O., Min, H. S., and Yin, Y. (1998). Integration of inductive learning and neural networks for multi-objective FMS scheduling. *International Journal of Production Research*, 36(9), pp. 2497-2509.
- Kjenstad, D. (1998). *Coordinated Supply Chain Scheduling*. Ph. D. Thesis. Norwegian University of Science and Technology, Trondheim, Norway.
- Knowles, J. D. and Corne, D. W. (2000). Approximating the non-dominated front using the Pareto archived evolution strategy. *Evolutionary Computation*, 8(2), pp. 149-172.
- Kursawe, F. (1991). A variant of evolution strategies for vector optimization. In Schwefel, H. P. and Manner, R. (Eds.), *Parallel Problem Solving from Nature. 1st Workshop, PPSN 1*, pp. 193-197, Dortmund, Germany: Springer-Verlag. Lecture Notes in Computer Science No. 496.
- Laumanns, M., Zitzler, E., and Thiele, L. (2001). On the effects of archiving, elitism, and density based selection in evolutionary multi-objective optimization. In Zitzler, E. et al. (Eds.), *Proceedings of the First International Conference on Evolutionary Multi-Criterion Optimization (EMO 2001)*, pp. 181-196. Springer-Verlag. Lecture Notes in Computer Science No. 1993.
- Lee, T. W. (1989). *Planning of Precast Component Production*. B. Sc. Thesis. National University of Singapore, Singapore.
- Leu, S. S., and Hwang, S. T. (2001). Optimal repetitive scheduling model with shareable resource constraint. *Journal of Construction Engineering and Management*, 127(4), pp. 270-280.
- Leu, S. S., and Hwang, S. T. (2002). GA-based resource-constrained flow-shop scheduling model for mixed precast production. *Automation in Construction*, Vol. 11, pp. 439-452.
- Levine, D. (1996). *Users Guide to the PGAPack Parallel Genetic Algorithm Library*. Argonne, Illinois: Argonne National Laboratory.
- Li, Y. and Man, K. F. (1998). Scheduling and planning problem in manufacturing systems with multiobjective genetic algorithm. In *Proceedings of 24th Annual Conference of the IEEE, IECON '98*, Vol. 1, pp. 274 -279.
- Li, E. Y. C. and Shaw, W. H. (1996). Flow-time performance of modified scheduling heuristics in a dynamic rescheduling environment. *Computers and Industrial Engineering*, 30(1-2), pp. 213-216.
- Li, E. Y. C. and Shaw, W. H. (1998). Simulation modeling of a dynamic job shop rescheduling with machine availability constraints. *Computers and Industrial Engineering*, 35(1-2), pp. 117-120.

- Madureira, A., Ramos, C., and do Carmo Silva, S. D. (2002). A coordination mechanism for real world scheduling problems using genetic algorithms. In *Proceedings of the 2002 Congress on Evolutionary Computation*, Vol. 1, pp. 175-180.
- Man, K. F., Tang, K. S., and Kwong, S. (1999). *Genetic Algorithms: Concepts and Designs*. London; New York: Springer.
- Marglin, S. (1967). *Public Investment Criteria*. Cambridge, Massachusetts: MIT Press.
- Marzouk, M. and Moselhi, O. (2004). Multiobjective optimization of earthmoving operations. *Journal of Construction Engineering and Management*, 130(1), pp. 105-113.
- Mattfeld, D. C. (1996). *Evolutionary Search and the Job Shop: Investigations on Genetic Algorithms for Production Scheduling*. Heidelberg: Physica-Verlag.
- Minton, S., Johnston, M., Philips, A., and Laird, P. (1990). Solving large-scale constraint satisfaction and scheduling problems using a heuristic repair method. In *Proceedings of the 8th National Conference on Artificial Intelligence*, pp. 17-24. Boston, Massachusetts: AAAI Press / The MIT Press.
- Montana, D., Brinn, M., Moore, S., and Bidwell, G. (1998). Genetic algorithms for complex real-time scheduling. In *Proceedings of the 1998 IEEE International Conference on Systems, Man, and Cybernetics*, pp. 2213-2218. La Jolla, California. IEEE.
- Murata, T., Ishibuchi, H., and Tanaka, H. (1996). Multi-objective genetic algorithm and its applications to flowshop scheduling. *Computers and Industrial Engineering*, 30(4), pp. 957-968.
- Nishimura, E., Imai, A., and Papadimitriou, S. (2001). Berth allocation planning in the public berth system by genetic algorithms. *European Journal of Operational Research*, Vol. 131, pp. 282-292.
- Noronha, S. J. and Sarma, V. V. S. (1991). Knowledge-based approaches for scheduling problems: a survey. *IEEE Transactions on Knowledge and Data Engineering*, 3(2), pp. 160-171.
- O'Grady, P. J. and Lee, K. H. (1989). An intelligent cell control system for automated manufacturing. In Kusiak, A. (Ed.), *Knowledge-based Systems in Manufacturing*, pp. 151-172. London: Taylor & Francis.
- Oh, J. and Wu, C. (2004). Genetic-algorithm-based real-time task scheduling with multiple goals. *Journal of Systems and Software*, 17(3), pp. 245-258.
- Osyczka, A. (1985). Multicriteria optimization for engineering design. In J. S. Gero (Ed.), *Design Optimization*, pp. 193-227. Orlando, Fla.: Academic Press.

- Osyczka, A. and Kundu, H. (1995). A new method to solve generalized multicriteria optimization problems using the simple genetic algorithm. *Structural Optimization*, Vol. 10, pp. 94-99.
- Paechter, B., Rankin, R., Cumming, A., and Fogarty, T. C. (1998). Timetabling the classes of an entire university with an evolutionary algorithm. In Eiben, A. E. *et al.* (Eds.), *Parallel Problem Solving From Nature – PPSN V*, pp. 865-874, Amsterdam, Holland. Springer-Verlag. Lecture Notes in Computer Science No. 1498.
- Parks, G. T. and Miller, I. (1998). Selective breeding in a multiobjective genetic algorithm. In Eiben, A. E. *et al.* (Eds.), *Parallel Problem Solving from Nature – PPSN V*, pp. 250-259. Berlin: Springer.
- Pendharkar, P. C. (1999). A computational study on design and performance issues of multi-agent intelligent systems for dynamic scheduling environments. *Expert Systems with Applications*, 16(2), pp. 121-133.
- Phan, H. T. (2000). *Constraint Propagation in Flexible Manufacturing*. New York: Springer.
- Prasad, T. D. and Park, N. S. (2004). Multiobjective genetic algorithms for design of water distribution networks. *Journal of Water Resources Planning and Management*, 130(1), pp.73-82.
- Raheja, A. S. and Subramaniam, V. (2002). Reactive recovery of job shop schedules – a review. *International Journal of Advanced Manufacturing Technology*, Vol. 19, pp. 756-763.
- Rangsaritratsamee, R., Jr. Ferrell, W. G., and Kurz, M. B. (2004). Dynamic rescheduling that simultaneously considers efficiency and stability. *Computers and Industrial Engineering*, Vol. 46, pp. 1–15.
- Rao, S. S. (1984). *Optimization: Theory and Applications*. New York: Wiley.
- Rovithakis, G. A., Perrakis, S. E., and Christodoulou, M. A. (2001). Application of a neural network scheduler on a real manufacturing system. *IEEE Transactions on Control Systems Technology*, 9(2), pp. 261-270.
- Rudolph, G. (1998). On a multi-objective evolutionary algorithm and its convergence to the Pareto set. In Fogel, D. B. (Ed.), *Proceedings of the 1998 International Conference on Evolutionary Computation*, pp. 511-516. Piscataway, New Jersey. IEEE.
- Sabuncuoglu, I. and Bayız, M. (2000). Analysis of reactive scheduling problems in a job shop environment. *European Journal of Operational Research*, 126(3), pp. 567-586.
- Sankar, S. S., Ponnambalam, S. G. and Rajendran, C. (2003). A multiobjective genetic algorithm for scheduling a flexible manufacturing system. *International Journal of Advanced Manufacturing Technology*, 22(3-4), pp. 229-236.

- Schaffer, J. D. (1984). *Multiple Objective Optimization with Vector Evaluated Genetic Algorithms*, Ph. D. Thesis. Vanderbilt University, Nashville, Tennessee.
- Schaffer, J. D. (1985). Multiple objective optimization with vector evaluated genetic algorithms. In *Genetic Algorithms and Their Applications: Proceedings of the First International Conference on Genetic Algorithms*, pp. 93-100, Lawrence Erlbaum.
- Shaw, K. J. and Fleming, P. J. (1996). An initial study of practical multi-objective production scheduling using genetic algorithms. In *Proceedings of the International Conference on Control '96*, University of Exeter, UK.
- Shaw, K.J. and Fleming, P.J. (1997). Including real-life problem preferences in genetic algorithms to improve optimization of production schedules. In *Proceedings of Genetic Algorithms in Engineering Systems: Innovations and Applications*, pp. 239-244.
- Smith, S.F. (1994). OPIS: A methodology and architecture for reactive scheduling. In Zweben, M. and Fox, M. S. (Eds.), *Intelligent Scheduling*, pp. 29-66. California: Morgan Kaufmann Publishers.
- Smith, S.F., Ow, P.S., Potvin, J.Y., Muscettola, N., and Matthy, D. (1990). An integral framework for generating and revising factory schedules. *Journal of the Operational Research Society*, 41(6), pp. 539-552.
- Srinivas, N. and Deb, K. (1994). Multiobjective optimization using non-dominated sorting in genetic algorithms. *Evolutionary Computation*, 2(3), pp.221-248.
- Steuer, R. E. (1986). *Multiple Criteria Optimization: Theory, Computation, and Application*. New York: John Wiley.
- Stützle, T. G. (1999). *Local Search Algorithms for Combinatorial Problems: Analysis, Improvements, and New Applications*. Sankt Augustin, German: Infix.
- Subramanyam, S. and Askin, G. (1986). An expert system approach to scheduling in flexible manufacturing systems. In Kusiak, A. (Ed.), *Flexible Manufacturing Systems: Methods and Studies*, pp. 243-256. Amsterdam: Elsevier Science Publishers.
- Sun, J. and Xue, D. (2001). A dynamic reactive scheduling mechanism for responding to changes of production orders and manufacturing resources. *Computers in Industry*, 46(2), pp. 189-207.
- Suresh, V. and Chaudhuri, D. (1993). Dynamic scheduling – a survey of research. *International Journal of Production Economics*, 32(1), pp. 53-63.
- Szelke, E., and Kerr, R.M. (1994). Knowledge-based reactive scheduling. *Production Planning and Control*, Vol. 5, pp. 124–145.
- Szelke, E., and Markus, G. (1995). A blackboard based perspective of reactive scheduling. In Kerr, R. and Szelke, E. (Eds.), *Artificial Intelligence in Reactive Scheduling*, pp. 60–77. London: Chapman and Hall.

- Talbi, E. G., Rahoual, M., Mabed, M. H., and Dhaenens, C. (2001). A hybrid evolutionary approach for multicriteria optimization problems: application to the flow shop. In Zitzler, E. et al. (Eds.), *First International Conference on Evolutionary Multi-Criterion Optimization*, pp. 416-428. Springer-Verlag. Lecture Notes in Computer Science No. 1993.
- Tam, C. M., Tong, T.K.L., and Chan, W.K.W. (2001). Genetic algorithm for optimizing supply locations around tower crane. *Journal of Construction Engineering and Management*, 127(4), pp. 315-321.
- Tamaki, H., Mori, M., Araki, M., Mishima, Y., and Ogai, H. (1994). Multi-criteria optimization by genetic algorithms: A case of scheduling in hot rolling process. In Fushimi, M. and Tone, K. (Eds.), *Proceedings of APORS'94*, Singapore, pp. 374-381. World Scientific Publishing.
- Tamaki, H., Nishino, E., and Abe, S. (1999). A genetic algorithm approach to multi-objective scheduling problems with earliness and tardiness penalties. In *1999 Congress on Evolutionary Computation*, pp. 46-52, Washington, D. C. IEEE.
- Tan, D. (2003). Prefabrication getting popular in Singapore. *PREFAB Architecture*, Issue No. 1, pp. 8-10.
- Todd, D. S. and Sen, P. (1997). A multiple criteria genetic algorithm for containership loading. In Bäck, T. (Ed.), *Proceedings of the Seventh International Conference on Genetic Algorithms*, San Francisco, CA, pp. 674-681. Morgan Kaufmann.
- Tranvouez, E., Ferrarini, A., and Espinasse, B. A. (2001). A multiagent modelling and simulation of workshop disruptions management by cooperative rescheduling strategies. In Giambiasi, N. and Frydman, C. (Eds.), *Proceedings of Simulation in Industry 2001*, pp. 917-924.
- Tsukada, T. K. and Shin, K. G. (1998). Distributed tool sharing in flexible manufacturing systems. *IEEE Transactions on Robotics and Automation*, 14(3), pp. 379-389.
- Tsukiyama, M., Mori, K., and Fukuda, T. (1992). Dynamic scheduling by using scheduling editor and distributed decision maker. In *IFAC Information Control Problems in Manufacturing Technology*, pp. 73-78.
- Valenzuela-Rendón, M. and Uresti-Charre, E. (1997). A non-generational genetic algorithm for multiobjective optimization. In Bäck, T. (Ed.), *Proceedings of the Seventh International Conference on Genetic Algorithms*, San Francisco, CA, pp. 658-665. Morgan Kaufmann.
- Van Veldhuizen, D. A. (1999). *Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations*. Ph. D. Thesis. Department of Electrical and Computer Engineering, Graduate School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, Ohio.

- Van Veldhuizen, D. A. and Lamont, G. B. (1998). *Multiobjective evolutionary algorithm research: A history and analysis*. Technical Report TR-98-03. Department of Electrical and Computer Engineering, Graduate School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, Ohio.
- Van Veldhuizen, D. A. and Lamont, G. B. (2000). Multiobjective evolutionary algorithms: Analyzing the State-of-the-Art. *Evolutionary Computation*, 7(3), pp.1-26.
- Vern, K. and Gunal, A. (1998). The use of simulation for construction elements manufacturing. In *Proceedings of the 1998 Winter Simulation Conference*, Vol.2, pp. 1273-1277.
- Warszawski, A. (1984). Production planning in prefabrication plant. *Building and Environment*, 19(2), pp. 139-147.
- Warszawski, A. (1990). *Industrialization and Robotics in Building*. New York: Harper & Row.
- Warszawski, A. and Ishai, E. (1982). Long range planning of prefabrication industry in a national economy. *Building and Environment*, 17(1), pp. 47-54.
- Yapo, P. O. (1996). *A Multiobjective Global Optimisation Algorithm with Applications to Calibration of Hydrologic Models*. Ph. D thesis. University of Arizona, Tucson.
- Yoshimura, K. and Nakano, R. (1998). Genetic algorithms for information operator scheduling. In Fogel, D. B. (Ed.), *Proceedings of the 1998 International Conference on Evolutionary Computation*, pp. 277-282. Piscataway, New Jersey. IEEE.
- Zadeh, L. A. (1963). Optimality and Nonscalar-Valued Performance Criteria. *IEEE Transactions on Automatic Control*, AC-8(1):59-60.
- Zeleny, M. (1973). Compromise programming. In J. Cochrane and Zeleny, M. (Eds.), *Multiple Criteria Decision Making*, pp. 262-301. Columbia, South Carolina: University of South Carolina Press.
- Zheng, D. X. M., Ng, S. T. and Kumaraswamy, M. M. (2004). Applying a genetic algorithm-based multiobjective approach for time-cost optimization. *Journal of Construction Engineering and Management*, 130(2), pp. 168-176.
- Zitzler, E. (1999). *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications*. Aachen: Shaker.
- Zitzler, E. and Thiele, L. (1998). *An evolutionary algorithm for multiobjective optimization: the strength Pareto Approach*. Technical Report 43, Zürich, Switzerland: Computer Engineering and Networks Laboratory, Swiss Federal Institute of Technology.
- Zitzler, E. and Thiele, L. (1999). Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4), pp.257-271.

- Zitzler, E., Deb, K., and Thiele, L. (2000). Comparison of multiobjective evolutionary algorithms: empirical results. *Evolutionary Computation*, 8(2), pp. 173-195.
- Zweben, M., Daun, B., Davis, E., and Deale, M. (1994). Scheduling and rescheduling with iterative repair. In Zweben, M. and Fox, M. S. (Eds.), *Intelligent Scheduling*, pp. 241-256. San Francisco, CA: Morgan Kaufmann.
- Zweben, M., Davis, E., Daun, B., and Deale, M. J. (1993). Scheduling and rescheduling with iterative repair. *IEEE Transactions on Systems, Man, and Cybernetics*, 23(6), pp.1588-1596.
- Zydallis, J. B., Van Veldhuizen, D. A., and Lamont, G. B. (2001). A statistical comparison of multiobjective evolutionary algorithms including the MOMGA-II. In Zitzler, E. *et al.* (Eds.), *Proceedings of the First International Conference on Evolutionary Multi-Criterion Optimization (EMO 2001)*, pp. 226-240. Springer-Verlag. Lecture Notes in Computer Science No. 1993.

APPENDIX I
Data and Schedules for the L-U Example

Table A.1 Site demands for the L-U example

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	T
	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	
E1					5						5					5					5				5					6		31
E2								3										3										2				8
E3					5						5					5					5				5					6		31


Table A.2 Original production schedule of the L-U example

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T
M1		E1	E1	E1	E1				E1	E1	E1	E1		E1	E1	E1	E1	E1	E1		E1	E1	E1	E1	E1	E1		E1			
M2					E2			E2		E1	E1			E1		E2	E2					E1									
M3				E3	E3			E2		E3	E3			E3	E3		E2				E3		E3								
M4			E3	E3	E3				E3	E3	E3			E3	E3	E3		E3	E3			E3	E3	E3	E3	E3	E3				


Table A.3 One repaired production schedule of the L-U example

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T
M1		E1	E1	E1	E1				E1	E1	E1	E1		E1	E1	E1	E1	E1	E1		E1	E1	E1								
M2		E2			E2			E2		E1	E1			E1			E1	E1	E1		E2	E1	E1								
M3				E3	E3			E2		E3	E3			E3	E3	E3	E2	E3	E3		E3	E2	E3								
M4			E3	E3	E3				E3	E3	E3			E3	E3	E3		E3	E3			E3	E3	E3							

Notes: 1. Priority of disturbance resolution and corresponding heuristics are D6(H2)->D5(H5)->D4(H5)->D2(H2)->D1(H4)->D3(H2).
 2. The repaired production schedule has an objective vector of (1, 4).

 Sundays and public holidays

 Disturbances

 Relocation of disturbances in the repaired schedule

APPENDIX II

Data and Schedules for the M-U Example

Table B.1 Site demands for the M-U example

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	T
	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	
E1					7						6					6					6				6					6		37
E2								3										3										3				9
E3					7						6					6					6				6					6		37


Table B.2 Production schedule of the M-U example

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T
M1		E1	E1	E1	E1			E1	E1	E1	E1	E1		E1	E1	E1	E1	E1	E1		E1	E1	E1	E1	E1	E1					
M2				E2			E1	E1		E2	E2	E2		E1	E1			E1	E1			E1	E1	E1							
M3							E1		E3	E3	E3	E3		E3				E2	E2		E3	E3	E3	E3	E3	E3					
M4		E3	E3	E3	E3		E3	E3	E3	E3	E3	E3		E3	E3	E3	E3		E2		E3	E3	E3	E3	E3	E3					

Table B.3 One repaired production schedule of the M-U example

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T
M1		E1	E1	E1	E1		E1	E1	E1	E1	E1	E1		E1	E1	E1	E1	E1	E1		E1	E1	E1	E2							
M2		E1		E2			E1	E1	E2	E2	E2	E2		E1	E1	E3	E1	E1	E1		E1	E1	E1								
M3							E1		E3	E3	E3	E3		E3	E2	E3	E3	E2	E3		E3	E3	E3								
M4		E3	E3	E3	E3		E3	E3	E3	E3	E3	E3		E3	E3	E3		E3	E3		E3	E3	E3								

Notes: 1. Priority of disturbance resolution and corresponding heuristics are D1(H5)->D2(H1)->D5(H2)->D6(H5)->D3(H5)->D3(H5).
 2. The repaired production schedule has an objective vector of (1, 12).

 Sundays and public holidays  Disturbances  Relocation of disturbances in the repaired schedule

APPENDIX III

Data and Schedules for the H-U Example

Table C.1 Site demands for the H-U example

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	T
	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	
E1					8						7					8					8				8					6		45
E2							2											3										2				7
E3					8						7					8					8				8					6		45


Table C.2 Production schedule of the H-U example

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T
M1			E1	E1	E1		E1	E1	E1	E1	E1	E1		E1	E1	E1	E1	E1	E1		E1	E1	E1	E1	E1	E1					
M2		E1	E1	E1	E1		E2	E2	E2	E1	E1	E1		E1	E1	E1	E1	E1	E1		E1	E1	E2		E1	E1					
M3		E3	E3	E3	E3		E3	E3	E3	E3	E3	E3		E3	E3	E3	E3	E3	E2		E3	E3	E3	E3	E1	E3					
M4			E3	E3	E3		E3	E3	E3	E3	E3	E3		E3	E3	E3	E3	E3	E3		E3	E3	E3	E3	E3	E3					


Table C.3 One repaired production schedule of the H-U example

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T
M1		E1	E1	E1	E1		E1	E1	E1	E1	E1	E1		E1	E1	E1	E1	E1	E1		E1	E1	E1								
M2		E2	E1	E1	E1		E2	E2	E2	E1	E1	E1		E1	E1	E1	E1	E1	E1		E1	E1	E2								
M3		E3	E3	E3	E3		E1	E3	E3	E3	E3	E3		E3	E3	E3	E3	E3	E2		E3	E3	E3								
M4			E3	E3	E3		E1	E3	E3	E3	E3	E3		E3	E3	E3	E3	E3	E3		E3	E3	E1								

Notes: 1. Priority of disturbance resolution and corresponding heuristics are D6(H7)->D1(H3)->D4(H5)->D3(H7)->D5(H5)->D2(H7).
 2. The repaired production schedule has an objective vector of (10, 19).

 Sundays and public holidays

 Disturbances

 Relocation of disturbances in the repaired schedule

