Material Cutting Plan Generation Using Multi-Expert and Evolutionary Approaches

Chang-Yu Hung

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
In partial fulfillment of the requirements for the degree of

Doctor of Philosophy
In
Industrial and Systems Engineering

Approved:

Dr. C. Patrick Koelling, Co-Chair	Dr. Robert T. Sumichrast, Co-Chair
Dr. Kimberly P. Ellis	Dr. Wolter J. Fabrycky

Dr. Dinesh Verma

July 5, 2000
Blacksburg, Virginia

Key Words: Cutting Plan Generation, Multi-expert System, Evolutionary Algorithms, Grouping Genetic Algorithm

Material Cutting Plan Generation Using Multi-Expert and Evolutionary Approaches

Chang-Yu Hung

ABSTRACT

Firms specializing in the construction of large commercial buildings and factories must often design and build steel structural components as a part of each project. Such firms must purchase large steel plates, cut them into pieces and then weld the pieces into H-beams and other construction components. The details of the order and the production operation are specified in the "cutting plan." This dissertation focuses on solving this "cutting plan generation" problem with the goal of minimizing cost.

Two solution approaches are proposed in this dissertation: a multi-expert system and an evolutionary algorithm. The expert system extends the field by relying on the knowledge of multiple experts. Furthermore, unlike traditional rule-base expert systems, this expert system (XS) uses procedural rules to capture and represent experts' knowledge. The second solution method, called CPG_{EA}, involves development of an evolutionary algorithm based on Falkenauer's grouping genetic algorithm.

A series of experiments is designed and performed to investigate the efficiency and effectiveness of the proposed approaches. Two types of data are used in the experiments. Historical data are real data provided by a construction company. Solutions developed manually and implemented are available. In addition, simulated data has been generated to more fully test the solution methods. Experiments are performed to optimize CPG_{EA} parameters as well as to compare the approaches to each other, to known solutions and to theoretical bounds developed in this dissertation.

Both approaches show excellent results in solving historical cases with an average cost 1% above the lower bound of the optimal solution. However, as revealed by experiments with simulated problems, the performance decreases in

cases where the optimal solution includes multiple identical plates. The performance of the XS is affected by this problem characteristic more than that of CPG_{EA}. While CPG_{EA} is more robust in effectively solving a range of problems, the XS requires substantially less processing time. Both approaches can be useful in different practical situations.

ACKNOWLEDGMENTS

Numerous people have helped me throughout the course of this research. I am in debt to them and want to take this opportunity to express my sincerest acknowledgments.

The completion of this dissertation would not have been possible without the guidance and support of my major advisor, Dr. Sumichrast. Words cannot describe my sincerest thanks to him. As an associate dean in Pamplin College of Business, he has spent numerous hours giving me help in every aspect of this research even in his busiest schedule. I will always remember his patience, encouragement, insight, advice, and the trust he gives me. He has given me much more than I can conceive of from an advisor. It is a blessing to have him as my advisor and mentor.

I thank Dr. Koelling for being my advisor in the ISE department. His support, encouragement and trust through the years have given me the strength in completing this research.

I appreciate other committee members' invaluable advice, help and encouragement. In particular, I thank Dr. Ellis for her cheerful spirit and encouragement; her advice and input on the mathematical formulation and other aspects of the research. I thank Dr. Fabrycky for his support and guidance through my study in the ISE department. His insight and tireless pursuit on numerous aspects of the systems engineering has given me an example of the professional excellence. I thank Dr. Verma for his invaluable advice through out the course of my study in Virginia Tech. His experience and directions have expanded my perception on different aspects of the systems engineering.

There are many people outside my research committee that contribute to this dissertation. I thank Dr. John E. Kobza for his help and support when I needed him most in the ISE department. I thank Professor Benjamin S. Blanchard for being a special friend to me, for his encouragement, and his example of great accomplishment, integrity and humility. I thank Dr. Evelyn Brown from

Management Science and Information Technology department for her input on the GGA and review of this dissertation. I thank Dr. George R. Terrell from Statistics Department for his advice on the experimental design. I thank my friend, Hongjie Wang, in the ISE department for his input on the cluster problems. I am also grateful to have invaluable input from many friends in LK Company, in particular, manager Shiang-Ray Su, Kuo-Chen Huang, Willy Zeng, and Shien-Rong Huang.

Thanks also go to my friends in the ISE department, Engineering Dean's Office, Public Service Program, and Chinese Bible Study for their support and great friendship.

Finally, my ultimate gratitude goes to God and my family. I thank God for guiding, providing and sustaining me through all my life and His blessing to my family. I thank my wife, Huan Yang, for her trust, help, sacrifice, and love through the years. She has made me live more fully in life. I thank my parents, brother and sisters for their unconditional love and support, in particular, for the unreserved provision from my parents. I thank my parents-in-law for their encouragement and support, especially for the help from my mother-in-law on our family during the most critical period of this research. I thank my beloved daughter, Eliza, for the everyday joy she brings to our family. It is a blessing to have them in my life. My dedication of this work is to all of them.

To the Almighty

To my family

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1 INTRODUCTION

Every successful manufacturing company must plan the efficient use of raw material. This planning has many facets and implications. In a custom-product make-to-order industry, material plans can impact customer service, the accuracy and speed of cost estimation, and possibly the firm's ability to secure contracts. In every industry it has a fundamental influence on cost and processing needs.

This dissertation focuses on a firm's ability to purchase raw material in bulk and subsequently to divide and process it. The problem shares general characteristics with well known problems such as cutting stock problems, cluster problems and grouping problems. Many of the models and solution techniques for these related problems are useful starting points for this research but none of them provide an acceptable approach to the problem of interest.

This chapter provides an overview of the problem of interest, the cutting plan generation or CPG problem. It describes the CPG problem in the context of other operations research models and techniques. The scope of this research is also discussed.

1.1 Overview of Cutting Plan Generation (CPG)

In the steel construction industry, preparing and ordering raw material is crucial to the efficiency and control of cost within the whole process operation. The cost of raw material may constitute 45% to 60% of the total cost (Huang 1998). Raw material is ordered in bulk to get volume discounts or to avoid extra costs applied to small orders. In ordering raw material, a crucial design operation is generating the "cutting plan" that specifies the details of the order and how the production operations will use the steel to construct the building. The cutting plan is important because it links architectural design, material preparation, manufacturing, and final assembly. The following sections explain the

background of cutting plan generation in the steel construction industry and the motivation for this research.

1.1.1 Process of Steel Construction

Figure 1 depicts an overview of a building project in a typical steel construction company. The process begins when the company learns of a construction project opportunity that fits its capabilities. Obviously, the company is interested in obtaining contracts that will provide enough revenue to pay for its costs and still make a profit. As shown in Figure 1, success at this phase requires learning about the requirements from the potential customer, negotiating with the customer, and possibly bidding on the project. During this first stage, engineers prepare a preliminary design and use it to estimate costs. If the contract for the construction is awarded, the design work becomes more detailed. This second step is highlighted in Figure 1, and due to its importance, is displayed in more detail in Figure 2.

Design engineers often decompose the project into more manageable parts such as the building floor. Based on the blue prints and other design specifications, final requirements for steel construction components (e.g., H-beams) are determined. Raw material requirements are calculated based on the necessary construction components. The raw material used in the steel construction industry includes standard and made-to-order size plates. Management's design philosophy and the project lead time determine whether standard or made-to-order sizes are more appropriate for a project. This research assumes that made-to-order sizes will be used.

In this environment, steel elements are cut from custom-sized plates and welded into construction components. So prior to ordering material, the design engineers must determine the construction elements required for the project and combine them into large steel plates. These engineers attempt to do this in an economical way. The arrangement of how the elements are cut from the steel plates is called the material cutting plan. The cutting plan is the bridge between the design

engineers and the production operators and represents the end of the detailed design phase.

The order for material plates is released. After the plates are delivered, they are cut into steel elements and welded to form the necessary construction components. Cutting and welding the elements into components is the beginning of the production phase.

Although many steps of a construction project are affected by the work reported in this dissertation, the focus is on two processes shown in Figure 2: determining the requirements for material elements and generating cutting plans. These processes are very time consuming if completed manually and are critical in determining final construction cost.

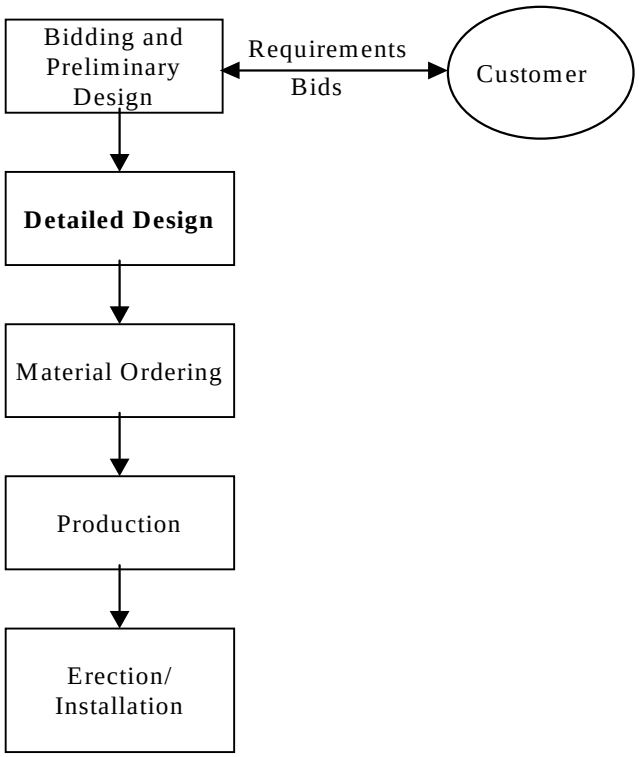

Figure 1. Overview of the steps of a construction project.

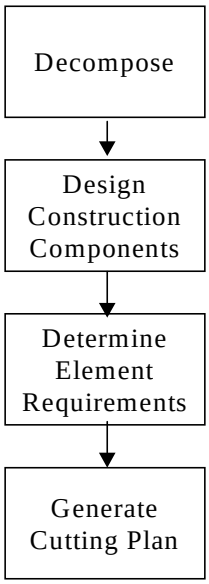

Figure 2. The steps of detailed design of a construction project.

1.1.2 Cutting Plan

As discussed above, cutting plan generation is the central process in the detailed design phase. It provides information that impacts all aspects of operations such as marketing, pricing, cost, production and material ordering. It is important to generate a cost saving cutting plan in a timely fashion so that the whole system can be synchronous and efficient. As the material costs make up most of the project cost, a small improvement in the cutting plan generation process can lead to a significant cost savings and a competitive advantage. However, cost is not minimized simply by minimizing material waste. The unit cost of raw material varies with the width of the plate ordered, in addition to other factors. Often, the material unit cost increases for both small and large sizes so a construction company must attempt to group its material requirements to achieve plates that have a width between the bounds defining the lowest unit cost. Since the steel plates can be ordered in any desired size within a range, the need to address material waste is a secondary factor.

Provided with raw material requirements and unit cost information, engineers attempt to generate a cutting plan that fulfills the demand at the minimum cost.

Without a systematic generating procedure, engineers have to use their experience to solve the problem raised from the custom-order size plates. Subsequently, they become experts in solving the CPG problem. The experts often use rules of thumb and trial-and-error to solve the problem. Based on an individual's experience and the characteristics of the problem, the results generated by different experts may not be the same.

It is desirable to automate the solution of the CPG problem so that the cutting plans are better in quality, faster in time, more accessible, and more informative. Improved computation methods will benefit the operation in several aspects.

(1) Minimize material and design labor cost.

(2) Reduce necessary lead-time in material ordering.

(3) Improve project cost estimation that would enable timely analysis and evaluation of different design alternatives so that project bidding can reflect more accurate cost figures.

(4) Speed the detail design process so that architects can generate more design alternatives early in the design stage.

(5) Estimate the impact of uncertainty in various parameters so that contingency plans can be developed.

(6) Monitor production and provide feedback for continuous improvement.

1.2 Related Operations Research Models and Solution Techniques

In search of computation models and solution methods for the CPG problem, several classic operations research (OR) models are studied. The CPG problem shares some characteristics with these problems. These problems include the well-known cutting stock problem (or, more broadly, the "cutting and packing problems") and the cluster problem.

1.2.1 Classic OR Problems

Superficially, the CPG problem looks like a variation of the two-dimensional cutting stock problem. Both problems satisfy demand by cutting small elements from a number of large stocks and both seek to minimize the material (stock) cost. However, two aspects distinguish the CPG problem from the cutting stock problem. (1) CPG does not have a fixed stock size or even a set of fixed stock sizes. The made-to-order plates can have an infinite number of sizes within a certain range. (2) The material unit cost is not fixed. It depends on the size of stocks and the ordered quantity. These two distinctions hinder the application of modeling and solution techniques for cutting stock problems to the CPG problem.

In further study, the CPG problem seeks to group required elements into groups (plates) that minimize cost. This characteristic of grouping different entities is similar to the cluster problem. However, the typical cluster problem seeks to group entities that share similar characteristics. If applied to CPG, it means grouping elements with similar dimensionality. This is not the objective of interest.

1.2.2 Other Solution Techniques

In search of solution techniques for the CPG problem, two approaches show particular promise: expert systems and evolutionary algorithms. Expert systems use a knowledge base to preserve experts' knowledge and have been used in many areas to assist or to replace experts. The application of expert systems can provide better performance and reduce cost at the same time. Engineers responsible for generating the cutting plans become experts over time and can serve as the knowledge source in developing an expert system.

Traditional expert systems often use one single expert as the major knowledge source to avoid opinion conflicts and to save development cost. Recently, systems based on multiple experts have been developed to solve more complex

problems. These multi-expert systems have the advantages of preserving different strengths of multiple experts, sharing knowledge and better rationalization. However, the development of multi-expert systems poses challenges in synthesizing different reasoning, reconciling different opinions and preserving different strengths among experts.

The other solution technique that shows promise in solving the CPG problem is evolutionary algorithms (EAs). EAs encompass a great number of computational techniques that use the principles of biological evolution in modeling and processing. EA is adopted increasingly in real world applications (Davis 1991; Goldberg 1994). It is found to be an efficient and robust method in many large, complex problems, and has demonstrated promising results in many areas including optimization, control and classification (Beasley 1997). In recent years, researchers have also shown that an EA produces superior results in a grouping problem (Falkenauer 1998) compared to conventional techniques.

The efficient use of EAs requires extension and specialization of algorithms. The adjustment can improve the effectiveness of EAs by utilizing specific knowledge about the problem and combining other optimization methods, such as local search and heuristics (Ibaraki 1997). The parameters used within an EA have to be "optimized" so that the EA can reach its potential performance.

1.3 Scope and Limitation

It is well known from experience and research that most of the system life-cycle cost is committed based on the engineering design and decision made in the early stages of system life-cycle process (Blanchard and Fabrycky 1998; Wood III and Agogino 1996). A procedure to quickly generate high quality solutions to the CPG problem is the key to better design and decisions in the detailed design. The need was recognized in the steel construction industry long ago, and various systems have been proposed (Bellack 1984; Jármai 1990; Pasley and Roddis 1996; Yusuf et al. 1997). There are also some computer integrated

manufacturing systems available and implemented in the steel industry[1] (DynaSys Inc. 1998; INS Engineering Corporation 1998).

In this research, however, the focus is limited to finding better solution methods for the CPG problem. The broader problem, including the whole life cycle, is not considered. Those considerations include bidding of projects, the ability of manufacturing to efficiently carry out the plan, inventory control, material handling, labor, etc. The scope of this research has other limitations as well.

1. This dissertation is not intended to develop a decision support system that provides what-if capability and other interactive functionality for the steel construction industry.
2. The performance of the multi-expert system developed in this dissertation may depend on the patterns of the input data. Furthermore, the heuristic within the multi-expert system is problem-specific and may not be applicable in other similar problems that have different constraints. However, a realistic set and simulated data are tested to investigate the effectiveness of the heuristic.
3. Although the EA solution approach may be applicable in other highly constrained grouping problems with minor modification, its performance cannot be guaranteed outside the tested problem domain.

Even though there are limitations pertaining to this research, the solution approaches developed in this dissertation solve the practical problem effectively and add new knowledge for solving similar grouping problems.

1.4 Plan of Presentation

The following chapters present the study of and proposed solution approaches to the CPG problem. They include the related literature, results and analysis of the experiments, and conclusions.

[1] However, neither the proposed decision support systems nor commercial computer integrated systems address the CPG problem. Instead, the cutting plan is usually generated either manually or by a nesting algorithm which is a special type of cutting stock problem with elements of irregular shapes. This is the result from the assumption that companies are using standard size steel plates as their raw material.

Chapter 2 presents an in-depth review of literature on similar problems' solution techniques and the proposed solution approaches. It provides the mathematical formulation and solution approaches used in the cutting and packing problems, and the cluster problems. These problems in the existing OR area share some characteristics of CPG problem. The review explains those similarities and differences. Two proposed solution approaches are reviewed in Chapter 2. The chapter also explains the significance of the modeling concepts and solution techniques adapted from these fields.

Chapter 3 explains the CPG problem in more detail and develops a mathematical formulation. Two solution approaches developed in this dissertation are explained and demonstrated with examples. These approaches include an expert system developed from multiple experts' knowledge and an evolutionary algorithm developed based on Falkenauer's grouping genetic algorithm.

Chapter 4 presents a series of experiments including their results and analysis. These experiments are designed and performed to investigate the performance of both solution approaches as well as the effects of other factors that influence performance. The observation, analysis, and implication based on the results of the experiments are also discussed.

Finally, Chapter 5 presents the contributions of this research to the steel construction industry, the development of multi-expert systems and evolutionary algorithms. Future research concerning the advancement of the solution approaches, and the exploration of new knowledge with related methods and problems, are also discussed.

CHAPTER 2 LITERATURE REVIEW

Much research has focused on problems that share some of the characteristics of the CPG problem. In many industries (paper, glass, steel construction, etc.), companies face the problem of cutting required elements from one or more large objects or packing a limited space with small items. The problems faced by these industries are called "cutting and packing problems." The need to efficiently cut material or divide space is an important element of the CPG. Cutting and packing problems are reviewed in this section since those problems have some properties similar to the CPG problem.

The CPG problem can also be treated as a grouping problem with a geometric characteristic and complex cost structure. The most recognized grouping problems involve cluster analysis that seeks to group similar entities into distinctive groups.

Despite the similarities between those two problems and the CPG problem, several factors in design and cost constraints in the CPG problem hamper the adoption of solution techniques from these problems. Existing techniques can only serve as a starting point. Two solution techniques, evolutionary algorithms and an expert system approach, are proposed to solve the CPG problem. These techniques are also reviewed in this chapter, with an emphasis on evolutionary algorithms.

2.1 Cutting and Packing Problems

Companies in many industries are faced with the need to purchase standard sizes of material and subsequently cut them to fit their needs. Others need to pack or load small items into large containers for transportation or storage. While these problems may seem very different on the surface, they are actually closely related. The next section clarifies their commonalties. This cutting and packing problem appears under different names in literature and has been studied extensively in one or more dimensions. Though several papers discussed the

problem before 1961, Gilmore and Gomory's articles on linear programming approaches to one-dimensional cutting stock problems were the first practical techniques published (Gilmore and Gomory 1961; Gilmore and Gomory 1963; Sweeney and Paternoster 1992). Since then, the number of published articles related to the cutting and packing problem has increased every year (Sweeney and Paternoster 1992). For a review of cutting and packing problems, see Dyckhoff (1990), Dyckhoff and Finke (1992), Haessler & Sweeney (1991), Hinxman (1980), and Sweeney and Paternoster (1992).

The CPG problem is different from the cutting and packing problem because its objective function is unusually complex and because the large item (or space) is not restricted to a finite set or sizes. The process of selecting patterns in the cutting and packing problem is similar to the selection of different stock sizes in the CPG problem. In this section, the cutting and packing problem and the related assortment problem are investigated to explore possible formulation and solution techniques that may be used in the cutting plan generation problem.

2.1.1 Characteristic and Classification of Cutting and Packing Problems

The cutting and packing problem appears under different names in the literature. Dyckhoff (1990) lists some examples: "

- cutting stock and trim loss problems
- bin packing, dual bin packing, strip packing, vector packing, and knapsack (packing) problems,
- vehicle loading, pallet loading, container loading, and car loading problems,
- assortment, depletion, design, dividing, layout, nesting, and partitioning problems,
- capital budgeting, change making, line balancing, memory allocation, and multi-processor scheduling problems."

While the problems listed above have different names and applications, they share a logical structure. In particular, all have two fundamental properties: (Dyckhoff 1990)

1. “There are two groups of basic data whose elements define geometric bodies of fixed shapes in a one- or more-dimensional space of real numbers.” The group of large objects is called “the stock”; the other group of small objects (called items) is the list of ordering or required items.
2. The cutting or packing process performs geometric allocation of small items to large objects. Residual space not belonging to small items is usually called “trim loss”.

In this dissertation, the term “cutting and packing problem” is used as the general problem domain including all problems sharing the above two properties.

Since these two characteristics of the cutting and packing problem are common in many situations, the problem can be extended to cover many problems sharing the same properties in the abstract dimensions. Dyckhoff lists some that share the properties with the dimensions of various natures:

1. Dimension of weight: knapsacking, and vehicle loading

2. Dimension of time: assembly line balancing and multiprocessor scheduling

3. Financial dimension: capital budgeting and change making

4. Other dimension: computer memory allocation.

Based on the common logical properties, more specific characteristics can be deduced. Some important characteristics include:

1. Dimensionality: minimum number of relevant dimensions for a geometric description of a pattern
2. Assignment restrictions: assignment of large objects to small items.
3. Assortment of the stocks and the items: type and number of existing figures of the objects/items.

4. Quantity measurement: type of quantity measurement for objects/items with identical measurements in the relevant dimensions; it includes discrete and continuous.
5. Shape of figures: figure of the object/item is determined by its form, orientation and size, e.g. rectangular, non-rectangular, parallel orientation, 90 degree turns parallel.
6. Availability: number, sequence and dates of objects/items
7. Pattern restrictions: including geometrical and operational characteristics.
8. Objectives: explicit formulated goals.
9. Status of information and variability: reliability of data (e.g. deterministic or stochastic)
10. Solution methods: basic structure of the applied solution approach, including object-orientated methods and pattern-orientated methods.

The above properties show the large variety of applications of cutting and packing problem. Dyckhoff (1990) developed a classification scheme for cutting stock problems by classifying the four most important characteristics:

1. Dimensionality
   (1) One-dimensional.
   (2) Two-dimensional
   (3) Three-dimensional
   (N) N-dimensional with N>3
2. Kind of assignment
   (B) All objects (stocks) and a selection of items
   (V) A selection of objects and all items.
3. Assortment of large objects
   (O) One object.
   (I) Many objects of identical figure
   (D) Different figures
4. Assortment of small items
   (F) Few items (of different figures).
   (M) Many items of many different figures.

(R) Many items of relatively few different (non-congruent) figures.

(C) Many identical items.

By combining the distinguished major types of the above four characteristics, denoted by fourth-tuple of respective symbols, many cutting and packing problems can be classified into 96 different categories. For example, 3/B/O/F denotes all three-dimensional cutting and packing problems where one large object has to be packed with a selection out of a few small items. Using Dyckhoff's classification scheme, the classic knapsack problem is one-dimensional with one large object that has to be packed with a selection from the set of small items (1/B/O/). The pallet loading problem is two-dimensional with identical small items (2/B/O/C).

In this research, Dyckhoff's typology is used for the classification of different cutting and packing problems. The CPG problem is similar to one of the two-dimensional cutting and packing problem and the related assortment problem with some differences in the restriction of availability of large object's sizes. (See section 2.1.4). The following sections address these two types of problems and their solution approaches.

2.1.2 Two-Dimensional Cutting and Packing Problems

Recalling the problem description, the CPG can be treated as the two-dimensional cutting stock or assortment problem with the following properties:

1. two-dimensional 2-stage guillotine cut pattern (2^2)
2. all small items must be assigned to a selection of stocks (V)
3. large objects (stocks) are the groups formed by the small items (D^3)
4. small items are of different sizes (M).

[2] A cut goes from one edge of the large object to its opposite edge is called "guillotine" cut.

[3] In our research problem, the sizes of large objects (stocks) are not fixed, but determined by the combination of small items.

Therefore, the CPG is similar to the two-dimensional problem with a selection of stock sizes and many items of many different figures (2/V/D/M). The following are some previous solution approaches to this type of problem.

The two-dimensional cutting and packing problem is NP-hard so exact solution approaches are not practical when the number of objects/items is as large as is commonly found in real problems (Garey and Johnson 1979). The solution methods to the cutting and packing problems can be classified into two general approaches: pattern-oriented and object-oriented (Dyckhoff 1990; Haessler and Sweeney 1991). Most of the solution methods use the pattern-oriented approach in solving the problem. The pattern-oriented approach, in contrast to object-oriented, first constructs patterns and assigns large objects and small items to some of these patterns. Table 1 shows different categories of solution approaches in cutting and packing problem.

Object/Item Oriented Approach		Pattern-Oriented Approach		
Exact Methods	Approximation Algorithms	Sequential Heuristics Procedures	Single Pattern	Multiple Pattern
Branch and bound, dynamic programming (Golden, 1976)	Bin Packing Algorithms (Coffman, 1984)	Heuristic (Haessler, 1971)	Knapsack Algorithms (Hinxman, 1980; Dowsland, 1985, Terno et al., 1987)	LP-Based and general heuristics (Hinxman, 1980; Stadtler, 1988; Farley, 1988)

Table 1. Different Approaches in Solving Cutting and Packing Problem
(Adapted from Dyckhoff 1990; Haessler and Sweeney 1991)

In the two dimensional cutting stock problem, the pattern generating process is much more difficult than that in the one-dimensional problem (Dyckhoff 1990; Sarin 1983). The pattern generating process is a sub-problem within the cutting stock problem.

The CPG problem does not have a fixed size stock; the objective is to generate several different patterns that will include all the demand items. In essence, it is a problem of pattern generation or grouping of small items. In fact, the pattern generation problem is a special grouping problem with geometric constraints as is clarified in the next section.

2.1.3 Pattern Generation in Two-Dimensional Cutting Stock Problems

Since the CPG problem is similar to pattern generation, we will focus on the solution techniques of pattern generation. Many different methods have been proposed for pattern generation (Hinxman 1980; Sarin 1983). The following are some major solution techniques.

2.1.3.1 Two-Stage Method

Gilmore and Gomory (1965) present a two-stage procedure to solve the guillotine cut pattern generation problem. The knapsack problem is solved in stages to obtain feasible patterns. The first stage cuts a stock into strips and the second stage cuts these strips into required pieces. The one-dimensional knapsack problems are then formulated accordingly. In stage 1, for each width w_i, feasible patterns of fitting rectangles of width $w_j \leq w_i$ in strips of size $w_i \times L$ are determined by solving the following knapsack problem:

$$\text{Max:} \qquad v_i^* = v_1 q_1 + ... + v_t q_t$$

$$\text{Subject To:} \qquad l_1 q_1 + ... + l_t q_t \leq L$$

Where l_1, ..., l_t and q_1, ..., q_t are respectively the lengths and number of pieces for which $w_j < w_i$, $j = 1, ..., t$, v_i is the value of piece type i, and L is the stock length. The Stage 2 knapsack problem is then to fit strips of size $w_i^* L$ into rectangular stock of size $W \times L$ so as to

$$\text{Max:} \qquad v_1^* q_1 + ... + v_m^* q_m$$

$$\text{Subject To:} \qquad w_1 q_1 + ... + w_m q_m \leq W$$

Marconi (1970) also considers a two-stage problem. In his models, the unit cost of stock sheets depends upon their size. He seeks to deal with difficulties which include the limited availability of stock of certain sizes and constraints imposed by the cutting machinery. He adapts the solution model of Gilmore and Gomory with the order list of different sizes.

2.1.3.2 Dynamic Programming

Gilmore and Gomory (1966) developed dynamic programming recursions for generating patterns both for staged and general guillotine cuts. The recursion for generating patterns with general guillotine cuts is:

$$G(X,Y) = \underset{X_0,Y_0}{Max}\{H(X,Y), G(X_0,Y) + G(X - X_0,Y), G(X,Y_0) + G(X,Y - Y_0)\}$$

where

$$X_0 \le \frac{1}{2}X \quad \text{and} \quad Y_0 \le \frac{1}{2}Y$$

G(X, Y) is the maximum value that can be obtained from an X by Y rectangle using W_i by L_i rectangles at shadow price U_i and any succession of guillotine cuts, and

$H(X, Y) = Max_i \{U_i \mid W_i < X \text{ and } L_i < Y\}$.

Beasley (1985b) improved the computation of the Gilmore-Gomory recursion and demonstrated the procedure could solve moderate problem in a reasonable time.

2.1.3.3 Tree Search Method

Christofides and Whitlock (1977) developed a tree search procedure that is effective for medium-sized problems. The tree procedure is as follows. At a root node of a tree, the emanating branches correspond to all possible cuts that can be made on the rectangular stock. A node at the end of the branch represents the rectangles produced by the corresponding cut. At a subsequent node, one of these rectangular pieces is selected to make further cuts. So a node in the tree represents all the rectangles produced by cuts according to the paths from the root node to that node.

This is a depth-first branch and bound algorithm to find optimal patterns for the general guillotine cut case with limits on the number of times a size can appear in the pattern. A procedure is used to eliminate duplicate cuts.

This algorithm took more than 2 minutes to generate a pattern with 20 ordered sizes on a CDC 7600 computer. This suggests that only small to medium sized problems can be solved since the number of iterations required to find an optimal linear programming solution could exceed 2 or 3 times the number of ordered sizes (Haessler and Sweeney 1991).

2.1.3.4 Heuristic Methods

Many heuristic rules are used to generate feasible patterns, e.g. (1) starting by packing largest sizes first, then next to largest and so on, (2) starting to pack smallest size first, then next to smallest and so on, (3) starting from a corner, (4) packing along the edge first and move inward, (5) packing from the center and move outward. These pattern generating procedures can also be used in the object-oriented solution approach to solve the cutting and packing problem without generating pattern first.

Beasley (1985c) proposed a heuristic algorithm for the guillotine cutting version of the problem based on a greedy procedure for generating two-dimensional cutting patterns. It also included a linear program for choosing the cutting patterns to use and an interchange procedure to decide the best subset of stock rectangles to cut. This version of the problem allows different sizes of available stocks.

Wang (1983) developed an alternative approach to generate general guillotine cutting patterns with limits on the number of times a size appears in a pattern. The ordered rectangles are combined either horizontally or vertically. Instead of the shadow price of the ordered sizes, an acceptable value for trim loss is used to drive the procedure. Viswanathan and Bagchi (1988) and Vasko (1989) improved Wang's algorithm by using better bounds to select the rectangles to be combined. Viswanathan and Bagchi used the unconstrained dynamic programming solution to the guillotine problem in a best-first search algorithm

that generates optimal solutions while requiring substantially fewer nodes than Wang's algorithm.

Vasko used the solution to the two-stage cutting pattern problem to find initial upper bounds on the general guillotine problem with restrictions on the number of times a size can appear in a pattern. The procedure is said to be 25 times faster than Wang's algorithm in generating optimal solutions to the pattern generation problem.

Adamowicz and Albano (1976) group order rectangles into strips one rectangle wide. They use a problem reduction method to find an arrangement of strips which forms a cutting pattern that satisfies an aspiration level for a single sheet, then repeat the process with a reduced order list for the next sheet. The total set of cutting patterns so produced is used as the initial state in a state-space search in which the arcs of the state-space graph are labeled with manipulations designed to group together the unused areas.

Hinxman (1976) also uses a problem reduction method involving the grouping of similarly shaped pieces. At each iteration of a repeated exhaustion reduction, the problem reduction method is applied. An attempt is made to form an economic pattern including as many pieces as possible of the most "awkward" type for which demand is not yet satisfied.

2.1.4 Two-Dimensional Assortment Problem

Relatively little literature considers the assortment problem which seeks to determine the best portfolio of stocks to keep in fulfilling the demand. From the above review, it is clear that the number of possible two-dimensional cutting patterns for any stock rectangle can be very large. So while the two-dimensional assortment problem can be formulated as an integer program, solution of all but small problems is impractical (Beasley 1985a). Even with the limitation of a specified set of stock rectangle sizes and simplifications to the objective function, the formulation that involves explicit consideration of all possible cutting patterns

is still very large. Hence it is not possible to solve the assortment problem except for very small problems using exact solution approaches (e.g. integer programming).

Most solution approaches for assortment problem in the literature use some heuristic combined with an algorithm that solves the pattern generation problem. Typically, the heuristic utilizes bounds in limiting the stock sizes to be investigated. The pattern generation method is used under the assumption of known stock sizes, thus it becomes a pattern generation problem.

2.1.4.1 Combination of Heuristic and Linear Programming

Chambers and Dyson (1976) describe the selection of stock sizes in the glass industry. The possible widths and lengths for stock sizes are integers in given ranges. The “best” set of *k* stock sizes is to be chosen. They proposed two methods with a common first stage. The first stage is solving a trim-loss problem for each possible width under the assumption that all feasible lengths are available. The heuristic method can be regarded as a state-space search in which the initial state is that all feasible lengths are used, each arc of the graph is labeled with an operation of dropping a length from the set to be used. The successors to only one node labeled with a given number of lengths are generated, and the search ceases when the generated nodes are labeled with sets of *k* length.

The other method explores a larger number of combinations. An integer programming method of partial enumeration using a branch and bound tree is set up to refine the heuristic solution. Lower bounds to curtail the tree search are calculated from consideration of the available lengths at any tree node.

Beasley (1985a) used a heuristic algorithm for the guillotine cutting stock problem. The heuristic includes a greedy procedure to generate cutting patterns, a linear program for choosing the cutting patterns to use, and an interchange procedure to decide the best subset of stock rectangles to cut.

2.1.4.2 Dynamic Programming

Page (1975) considered techniques for cutting rectangular steel bars. In his problem, there are orders for a large number of rectangular cross-sections. An order for a non-stock section is met by cutting down a larger stock section, so one order rectangle will be produced from each stock rectangle. Note that one stock produces only one item. A two-dimensional dynamic programming formulation is used with cross-section as the variable. The formulation is a relaxed form of the original problem, and the solution to the dynamic program gives the minimum feasible cost and provides the starting point for the heuristic method that follows. The heuristic uses the result of the dynamic program as the lower bound, and iterates dimensional changes of the stock sizes to find the best set of stock sizes to hold.

2.1.5 Differences Between the Research Problem and the Cutting Stock Problem

As stated in the previous section, the material cutting plan problem faced by many companies in the steel construction industry has limited similarity to a standard, two-dimensional cutting and packing problem. In both, an area is divided into pieces with the objective of minimizing cost and the constraint of satisfying the demand for particular size elements. However, the details of the objective functions and the constraints differ between these problems.

In a standard, two-dimensional cutting stock problem, the primary objective is to minimize material cost or equivalently, material waste. Material waste is caused by the need to cut from standard-sized stock. Although multiple stock sizes (as a cutting stock "portfolio") are considered in the assortment problem, only a limited number of different sizes of stock is considered in the solution methods (Beasley 1985a; Chambers and Dyson 1976; Gemmill and Sanders 1990; Hinxman 1980; Page 1975). Other approaches developed to select stock sizes involve analytical heuristics using iteration. These heuristics are defined to minimize trim loss. This

limitation of different solution approaches is due to the enormous number of combinations of the small items, i.e. number of possible patterns.

The solution approaches in the pattern generation problem within the two-dimensional cutting problem may be considered as alternative solution methods for the CPG problem. However, the pattern generation problem requires certain bounds or ranges in setting up the formulation (e.g. knapsack problem.) The material cutting plan problem of the steel construction industry is not constrained by standard stock sizes. Recall that CPG problem involves purchasing steel plates in any desired dimensions within a wide-range of sizes. In the width dimension, the only waste is caused by technological factors--a very small amount of material is lost (3 mm) for each cut made in a plate. Differences in lengths of elements combined in a plate also cause some material waste. However, the costs due to waste material are secondary considerations. The primary need is to form plates with dimensions associated with low unit cost. This makes standard two-dimensional cutting stock solutions inappropriate.

The heuristic approaches used in assortment problems are better candidates in solving our problem, e.g. Beasley (1985a), but the CPG cost structure is more complex and therefore even the assortment problem heuristics are not entirely adequate.

From the other prospective, the CPG problem can be treated as the need to group the required items so that the cost of resulting stocks (or groups) is minimized. This is the same characteristic as seen in grouping problems that are reviewed in the next section.

2.2 Grouping Problems and Cluster Analysis

Many practical problems consist of partitioning a set of objects into a collection of mutually disjoint subsets such that the resulting “groups” exhibit certain properties or satisfy certain criteria. Such problems are called cluster problems or grouping problems. In this dissertation, the problem of generating the cutting

plan involves grouping individual elements together to form the ordering plates. It can be characterized as a special type of grouping problem. The conventional cluster analysis methods are not suitable in approaching the CPG problem since the objective of joining similar elements may not produce the desirable plates with widths that fall within the minimum unit cost range. However, despite the different objective, formulations of the cluster problem serve as a valuable reference in forming a mathematical model of the CPG problem.

In this section, the cluster and grouping problems are defined and discussed. Mathematical formulations of the problem are introduced. Some of the solution approaches for grouping problems are discussed. Also, differences between the cutting plan generation problem and the typical cluster problems are addressed.

Clustering is a general term for formal, planned, purposeful, or scientific classification. Many natural objects need to be named and classified, and there is often the need to group similar entities together based on their characteristics. Almost any discipline faces this problem: biology, medicine, psychology, geography, applied economics, marketing, computer science, image processing, manufacturing, etc. The applications of classification are numerous: facilitating data organization, exploration of data sets for discovery, prediction of new objects, etc.

The cluster problem has been formally defined by Duran (1974)

> Given a set $I = \{I_1, I_2, ..., I_n\}$ and m, an integer less than or equal to n. Determine m clusters (subsets) of individuals in I such that I_i belongs to one and only one subset and those individuals which are assigned to the same cluster are similar but individuals from different clusters are different.

The homogeneity within each cluster and the separation between clusters can be defined in many ways. Numerous studies have been done in the past. Furthermore, many problems impose a priority on the clusters to be found; thus, various paradigms of cluster analysis are defined. To show the vast research

done in this area, Seber (1984) reported the number of references sited in two books that focus on this problem: Sneath and Sokal (1973) contains approximately 1,600 references and Duran and Odell (1974) includes over 400 references.

Despite the extensive application of cluster analysis, many problems of partitioning a set into disjoint subsets do not demand homogeneity and separation as the objective. These types of problems, called grouping problems, usually exists in man-made environments where grouping is desired to minimize a complex cost function. Grouping problems include the bin packing problem, workshop layout problem, graph coloring, etc. (Falkenauer 1998). The CPG problem belongs to this category. In this dissertation the term “grouping problem” is used to describe problems with any objective while “cluster problem” follows Duran’s definition that requires the homogeneity and separation as the objective (i.e. the cluster problem is a subset of the grouping problem).

2.2.1 Solution Methods for the Cluster Problem

Many solution techniques have been proposed for solving the subset of the grouping problem called the cluster problem. The technique used in the cluster problem is called cluster analysis. For the grouping problem without objective of homogeneity and separation, the usual approach is to formulate the problem as a mathematical programming model. Since most grouping problems, including cluster problems, are NP-hard, heuristic techniques are used in most cases to solve the mathematical model (Bhaskar and Narendran 1996; Cowgill 1993).

In this section, the solution techniques developed to solve cluster problems are briefly reviewed. Exact solution methods based on mathematical formulations are also discussed. These formulations are valuable for the composition of the CPG problem

2.2.2 Cluster Analysis

As explained previously, cluster analysis is “a generic name for a variety of mathematical methods, numbering in the hundreds, that can be used to find out which objects in a set are similar” (Romesburg 1984) or, in short, “the art of finding groups in data” (Kaufman and Rousseeuw 1990). Some important factors in cluster analysis include determination of the correct number of clusters, achieving coverage of all objects, identifying of irrelevant variables, limiting the degree of cluster overlap, controlling cluster size, controlling evenness of cluster size, defining similarity measures, etc. (Cowgill 1993). Among these factors, the most fundamental ones are concerned with attributes or measurements of objects, and the similarity measures. Attributes are measurements that describe the objects, such as height, weight, etc. The similarity measure is the mathematical description of closeness of different objects. It is often expressed in terms of distance or dissimilarities between two objects. Dissimilarities can be quantified in several ways. They can be computed from an individual attribute or from a combination of several attributes. Also, dissimilarities can be subjective ratings of how much certain objects differ from each other, from the point of view of one or more observers.

Many algorithms for cluster analysis are described in the literature; the choice of an algorithm depends both on the type of data available and on the particular purpose. Since cluster analysis is primarily used as a descriptive or exploratory tool, it is legitimate to use several algorithms on the same data. Many statistical tests can be carried out to measure the results from different algorithms for inferential or confirmatory purposes.

Clustering algorithms can be categorized into two classes: partitioning and hierarchical (Hansen et al. 1995; Kaufman and Rousseeuw 1990). It would be impractical to provide a complete review of the specific techniques proposed for each of these classes. Instead, an overview of the two classes of clustering algorithms is presented below.

2.2.2.1 Partitioning Methods

A partitioning method constructs a predetermined number of clusters, i.e., it classifieds the data into *k* groups, which satisfy the requirements: (a) each group must contain at least one object; and (b) each object belongs to exactly one group. The number of clusters to form, *k*, is specified by the user. The quality of the overall solution can be highly dependent on *k*. Therefore, the method is often carried out by trying different values of *k*.

One popular partitioning method called Convergent K-means has been shown to be one of the most robust partitioning method techniques (Cowgill 1993). The method selects *k* objects in the data set. The variables are scaled so that euclidean distance between objects represents the dissimilarity. The corresponding clusters are then found by assigning each remaining object to the nearest representative object. The objective is to minimize the average squared distance.

2.2.2.2 Hierarchical Methods

The other class of clustering method is hierarchical. Unlike partitioning methods, hierarchical methods do not construct a single partition with *k* clusters. Hierarchical methods deal with all values of *k* in the same run; i.e., including the situations of $k = 1$ (all objects belong to the same cluster) and $k = n$ (each object as a cluster). In between, all values of *k* are considered in a kind of gradual transition, i.e. the only difference between $k = r$ and $k = r+1$ is that one of the *r* clusters splits up in order to obtain $r+1$ clusters.

There are two kinds of hierarchical techniques: the agglomerative and the divisive. Agglomerative methods start when all objects are apart; then in each step two clusters are merged until only one is left. In contrast, divisive methods start when all objects in one cluster and in each following step split up a cluster until there are n (number of objects) of them. For a comparison of the performance of various hierarchical methods see Milligan and Cooper (1987).

2.2.3 Grouping by Complete Enumeration

Regardless of the objective function, one way to solve the grouping problem optimally is to evaluate the objective function for each grouping alternative and choose the one producing the optimal value. However, this strategy is not practical unless the number of objects and the number of clusters is small. Duran and Odell (1974) showed the number of grouping alternatives grows exponentially as the number of objects increases:

Let *n* be the number of objects and *m* be the number of clusters. If *m* is not specified, the total number of grouping alternatives is

$$\sum_{m=1}^{n}\left[\frac{1}{m!}\sum_{j=0}^{m}\binom{m}{j}(-1)^{j}(m-j)^{n}\right]$$

$S(n,m)=\frac{1}{m!}\sum_{j=0}^{m}\binom{m}{j}(-1)^{j}(m-j)^{n}$ is a Stirling's number of the second kind, and it can be shown that $\underset{n\to\infty}{Lim}\, S(n,m)\approx m^{n-1}$. Table 2 shows the number of grouping alternatives for different values of *n* and *m*.

n/m	1	2	3	4	5	6	7	8
1	1							
2	1	1						
3	1	3	1					
4	1	7	6	1				
5	1	15	25	10	1			
6	1	31	90	65	15	1		
7	1	63	301	350	140	21	1	
8	1	127	966	1701	1050	266	28	1

Table 2. Number of Grouping Alternatives for Different Values of n and m. (Duran and Odell 1974 p.41)

As previously stated, most grouping problems are NP-hard. Thus exact solution methods are not practical in real situations that often involve a large value for *n*. Take an example given by Anderberg (1973): if 25 objects are grouped into 5

clusters, there are over 10^{15} different cluster alternatives ($2.44 * 10^{15}$ using the above equation.) Only in a few cases, if some prior constraints can be imposed to reduce the possible grouping alternatives significantly, can exact solution methods be used. Nevertheless, the mathematical formulations of these methods are useful in constructing the CPG problem.

There are two major mathematical formulations: dynamic programming and integer programming. These formulations are presented in the following sections.

2.2.4 Dynamic Programming

There is no standard mathematical formulation for the dynamic programming problem. The equations and formulas pertinent to a dynamic programming problem depend on the particular situation at hand. The problem is usually reduced to a recursive relationship or equation which reflects the multiple interrelated decisions inherent in the dynamic programming procedure and which result in the final optimal result.

In the context of the grouping problem, dynamic programming is a procedure that seeks the optimum grouping in stages such that at each stage, the objective function is computed in such a way that redundant calculations inherent in the complete enumeration procedure are eliminated. In this way, the optimal solution will be attained in stages. Consequently, the dynamic programming approach requires large amounts of rapid access storage.

For illustration, we present a general dynamic programming formulation related to cluster analysis given by Jensen (1969):

$$W_k(z) = \begin{cases} 0 \quad \text{if } k = 0 \\ \min_y \left[T(z-y) + W_{k-1}(y) \right], \quad \text{if } k = 1, 2, ..., m_0 \end{cases}$$

where

m = number of disjoint and non-empty subsets into which the n objects are to be partitioned

k = index or stage variable

m_0 = m if $n \geq m$, and n-m if $n < m$

z = state variable representing a given set of objects at stage k

y = state variable representing a given set of objects at stage k-1

$z - y$ = subset of all objects contained in z but not in y

$T(z-y)$ = the "transition cost" of the objects in the cluster of objects in $(z-y)$.

The variables y and z represent 2 states (sets of objects) in stages k-1 and k, respectively. The difference z-y represents those objects contained in the stage k state but not in the stage k-1 state. $T(z-y)$ then represents the "transition cost" for those objects which are combined with the stage k-1 state objects and $W_k(z)$ gives the minimum value for the cost in partitioning the objects represented by z into k disjoint and nonempty subsets.

2.2.5 Integer Programming

The other mathematical formulation that can result in an optimal solution is integer programming. Many grouping problems (e.g. in manufacturing: printed circuit board grouping problem, group technology) can be formulated as an integer programming model (Askin et al. 1994; Bhaskar and Narendran 1996; Stecke 1983). In most cases, a heuristic solution is used due to the complexity (e.g. some are non-linear) or large search space (NP-hard) of the integer programming model.

A formulation given by Vinod (1969) is presented below for illustration:

Suppose there are n objects. Let n_j denote the number of objects in the jth group (j=1,..., m) with $n = \sum_{j=1}^{m} n_j$. The size of the largest cluster will be m_0. If m_0 is unspecified then m_0=n. The cost involved in placing the ith object in the jth group is denoted by c_{ij}.

Let a_{ij} = 1 or 0 according as the *i*th object is or is not contained in the *j*th group. For the purpose of identifying the groups the idea of group leader is created and the *j*th object is taken to be the leader of the *j*th group. Let $y_j = 1$ if the *j*th object is a leader and $y_j = 0$ otherwise. The integer programming model is

Minimize

$$\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}c_{ij}$$

Subject to

$$\sum_{j=1}^{n} a_{ij} = 1, \quad i = 1, 2, \ldots, n \qquad (1)$$

$$\sum_{j=1}^{n} y_j = m \qquad (2)$$

$$y_j \geq a_{1j},\ y_j \geq a_{2j},\ \ldots..,\ y_j \geq a_{nj},\ j = 1, 2, \ldots, n. \qquad (3)$$

Constraint (1) states that each object must be put into one group. Constraint (2) states that there are exactly m groups. The last constraint (3) says that the jth object must be a leader before any objects can be placed in the group corresponding to it. This constraint is also used to avoid repetition in the matrix A = $\{a_{ij}\}$. For example, suppose there are 3 objects. Without constraint (3), grouping of {1}, {2, 3} can be represented as $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix}$ or $\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}$, etc. (The row represents the objects to be grouped, and the column denotes groups.) The second one will violate constraint (3).

The above matrix representation of cluster membership of each object is adapted in most of integer programming formulations. We will also adapt this representation in modeling CPG problem.

2.3 Expert Systems

Expert systems (XS[4]) are "intelligent computer programs that use knowledge and inference procedures to solve problems that are difficult enough to require significant human expertise for their solution. The knowledge of an expert system consists of facts and heuristics" (Feigenbaum 1979). An expert system solves problems in a specific domain of expertise (i.e. field or specialty) and cannot be applied to general problem solving. This section provides a brief review of XS including characteristics, applications, and major components. The end of this section considers problems involving multiple experts.

2.3.1 Application and Characteristics of Expert Systems

Expert systems have been implemented in a wide-range of problems which involve interpreting, identifying, predicting, diagnosing, designing, planning, monitoring, debugging, testing, training and controlling (Badiru 1992). The applications of XS include business, education, manufacturing, medicine and law. The use of XS rises from the need to compensate for some of the limitations of human expertise. These limitations include the scarcity of human expertise, physical and mental limits, inconsistency and bias. From management's point of view, XS may provide some or all of the following (Badiru 1992; McCart 1991):

(1) Assistance for experts: XS can help experts make, check and document their decisions.
(2) Expert replacement or replication: Since human expertise is scarce, organizations may want to capture experts' knowledge and use that knowledge in the absence of experts.
(3) Sensitivity analysis: XS can facilitate the decision making by providing the probable outcomes of different courses of action.
(4) Training: XS can be used to train less experienced personnel.

[4] The abbreviation (XS) is used to differentiate from that of Evolution Strategy (ES).

(5) Performance enhancement: XS can increase the probability, frequency, and consistency of making good decisions. They also provide objective decision without bias and emotional reactions.
(6) Cost control: Once developed, XS provide real time, low cost and expert level solutions by the non-expert.

From the users' point of view, XS typically provide answers in a specific domain (field or area) and have the following characteristics (McCart 1991; Sumichrast 1995; Tzafestas et al. 1993).

(1) The most distinguishing feature of an expert system is its ability to perform like an expert in a specific domain. An XS can act like an expert in solving problems, giving advice, or providing information. The performance of an XS depends on how domain specific knowledge is captured, organized and accessed within the XS. Some sophisticated expert systems have the ability to learn and expand their own knowledge base as new information (or knowledge) is added.

(2) Expert systems often include a multi-faced user interface. The interface asks relevant questions, accepts input and displays the processing result. Many expert systems also have the ability to explain the reasoning followed in reaching a solution. The ability of an expert system to explain its reasoning is one of the advantages of expert systems over other intelligence modeling approaches (e.g. neural networks).

(3) Many problems addressed by expert systems can be formulated as state space search problems. Although the state space is often large, expert systems routinely search it with limited processing resources and obtain a good solution. Exhaustive search methods are not used since they require excessive memory and processing time. Expert systems use specific information of the domain to guide the search and reduce the searching time. This information is called heuristic information and the search methods are called heuristics. Heuristics do not always cover every possible situation; thus, occasionally expert systems fail to obtain a good solution.

(4) The knowledge involved in solving complex, real world problems is usually uncertain, imprecise and incomplete. XS, like human experts, often have the ability to deal with the uncertainty.

2.3.2 Expert Systems Structure

As defined above, an XS simulates the decision making and problem solving capability of a human expert in a certain domain and is expected to perform at a level comparable to an expert. In order to retrieve facts and apply heuristics efficiently, the knowledge within the expert system must be organized in an easily accessible format. Further, researchers have found that many problems require identical reasoning and processing, and differ only in the domain specific knowledge required for problem solution. For these reason, expert systems usually are organized into three components: (Badiru 1992; Tzafestas et al. 1993)

(1) Knowledge base: This contains declarative descriptions of expert information, problem solving rules, procedures or intrinsic data necessary for problem solving in the domain.
(2) Inference engine: This a generic control mechanism that solves a problem by interpreting the axiomatic knowledge in the knowledge base.
(3) Working memory: The inference engine records the current problem in a workspace called working memory. Working memory may also contain other modules for the purpose of knowledge acquisition (knowledge acquisition module), reasoning explanation (explanation module), and common interaction with the users (user interface) (Nikolopoulos 1997).

The division of the knowledge base and inference engine has the advantage of versatility - the knowledge base can be replaced with a new one. However, the inference engine and knowledge base are not completely independent. The content of the knowledge base has to take into account the built-in control mechanism of the inference engine. The performance of expert systems is highly affected by the structure of the knowledge base, or specifically, by the

knowledge representation scheme. The following sections discuss the knowledge acquisition and representation, and the reasoning mechanisms adapted in an inference engine as well as special concerns for problems involving multiple experts.

2.3.3 Knowledge Elicitation and Representation

The knowledge base preserves human experts' knowledge relevant to solving problems in the domain. The knowledge is captured through a knowledge acquisition process. The captured knowledge must be converted (or encoded) to a knowledge representation formalism that can be stored and manipulated by a computer. Some researchers use the term “knowledge acquisition” to include the knowledge elicitation and knowledge representation (Nikolopoulos 1997); some (Tzafestas and Adrianopoulos 1993) use “knowledge acquisition” as knowledge elicitation only, apart from the knowledge representation issue. Others emphasize the creative part of knowledge acquisition that “gives the person acquiring the knowledge the opportunity to independently generate new approaches to solving problems in the domain under consideration” (Badiru 1992). To avoid confusion, the more narrow and consistently defined term knowledge elicitation is used in this section.

The research of the problems and techniques for eliciting human knowledge is an on-going effort. To acquire knowledge, the question “what is knowledge?” must be addressed. One definition of knowledge is “an awareness of and an understanding of the facts, relationships, representations and processes of some subject” (Sumichrast 1995). Based on this definition, there is no clear representation for expert systems to possess such awareness. For computer systems, knowledge may mean “storing, retrieving and creating facts, relationships, representations and processes related to some subject.” (Sumichrast 1995) There are many representation schemes in expert systems, including: (Badiru 1992; Sumichrast 1995).

(1) Semantic networks: Semantic networks consist of a collection of nodes and arcs. The nodes are linked with arcs to form object relationships. Arcs carry notations that indicated the type of relationships between the nodes. Semantic networks have the advantages of flexibility in editing new components and ability to inherit relationships from other nodes. However, semantic networks lack a formal definitive structure which is required for implementation in an operational setting.

(2) Rules: Rules may be the most common and versatile of all representation schemes. Rules use the template “if X then Y” to represent knowledge. X is the premise, input, or antecedent while Y is the conclusion, output, or consequent. For example, “IF A is inactive AND B is unavailable THEN use cutting method C”. The antecedent typically contains several clauses linked by the logical connectives AND/OR. The advantages of using rules are (a) flexibility – individual rules can be easily added, deleted or changed; (b) ease of interpretation; (c) ability to represent the interaction between declarative and procedural knowledge. However, rules do not always provide an efficient means of representing knowledge. For example, procedural knowledge is not efficiently represented as rules. Even when rules efficiently represent domain knowledge, the number of rules required for some problems pose challenges in designing an efficient search strategy.

(3) Propositional Logic/Predicate Logic: Propositional logic is an elementary logic that is used to determine whether a given proposition is true or false. Predicate logic adds the capability to specify certain relationships and make generalizations about propositions. Propositional logic, predicate logic and an extension of predicate logic called first-order logic are commonly used in representing knowledge. Propositional logic has the advantages of simplicity, conciseness and modularity. However, it has difficulty representing procedural and heuristic knowledge and in managing large knowledge bases due to restricted organizational structure.

(4) Others: Other representation schemes include Cases, Scripts, Frames and Object-Attribute-Value triplets.

Knowledge representation is very important to an expert system's performance. Different knowledge representation methods have different strengths and limitations. In practice, rule-based systems are the easiest to implement. Rule-based systems have acceptable performance overall but their efficiency decreases as the knowledge volume increases. The choice of representation scheme should be considered with the specific application domain.

2.3.4 Reasoning in Expert Systems

The knowledge stored in a knowledge base must be processed through logical reasoning to produce solutions. An expert system's reasoning and control mechanisms are usually embedded in the inference engine. Presented below are some important concepts and models for inference.

(1) Deductive and inductive reasoning: Induction and deduction are useful general concepts in expert systems. Deduction draws specific conclusions about a given member of a class from general information about that class. Induction, on the contrary, draws a general conclusion based on specific facts.

(2) Modus Ponens: Modus ponens is a logical reasoning that allows conclusions to be drawn from rules and facts. This is the formal name for allowing that when the premise of a rule is true, it is valid to believe the conclusion is true. For example, given the rule of "if A is true then B is true" and the fact "A is true", modus ponens allows the conclusion of "B is true" to be drawn.

(3) Modus Tollens: Modus tollens is the converse of modus ponens. Its reasoning states that if the premise of a rule must be true for the rule's conclusion to be true, then if the conclusion is false, the premise is false. For example, given the rule "if A is true then B is true" and the fact "B is false", modus tollens concludes "A is false".

(4) Forward Chaining: Forward chaining is a control mechanism for inferring new facts from given facts. A forward chain begins with known facts and infers new facts until a goal is reached or no new facts can be inferred.

(5) Backward Chaining: Backward chaining is a control mechanism. It starts from a goal state and backtracks to the paths that may lead to the goal. The process is backward since it begins with a potential solution and searches for facts that support the solution.

Many expert systems use more than one reasoning and control method. For example, rule-base systems may use both forward chaining and backward chaining.

2.3.5 Multi-Expert Systems

Multiple experts may be needed to solve very large and complex problems. Many problems arise while dealing with multiple experts. These problems include different opinions about the same problem, different knowledge backgrounds and expertise, etc. The major challenges are the control of conflicting opinions and coordination between different area of expertise. An expert system based on multiple experts has potential advantages including: (Badiru 1992) (1) the ability to share experience, knowledge, and resources; (2) increased credibility; (3) improved morale due to participation; (4) better rationalization by observing others' views. O'Leary (1998) also suggests that, through empirical study, knowledge acquisition from multiple experts provides more correct orderings to the probabilities than knowledge acquisition from an individual.

There are two different schemes for the development and structure of multi-expert systems. First, the expert system may contain many sub-systems (often called agents) that tackle different portions of the problem. One challenge in developing this type of system is the coordination and cooperation of the sub-systems (Gasser 1991). The other scheme is using a single knowledge base containing knowledge from more than one expert. Since the CPG problems

involve multiple experts in solving one problem, only the second scheme will be considered.

Various communication techniques have been proposed to elicit knowledge from multiple experts. These techniques include brainstorming, the Delphi method, nominal group techniques, etc. (Badiru 1992). These techniques facilitate knowledge extraction from multiple experts but were originally developed for general use rather than for the development of expert systems. Hanachi (1996) proposed a technique specifically for XS development. This technique uses an active and deductive database as a cooperative information system that facilitates the development of the multi-expert systems. This system maintains information about the development project and a set of rules that coordinate and control the different participants' activities.

The reconciliation and incorporation of conflicting opinions is one major problem for knowledge representation. In a multi-expert system, Li et al. (1995) suggest a fuzzy reasoning method that will combine experts' knowledge and keep the independence of individual experts. The method uses a technical value of different experts as the "competition degree of experts" in determining which expert's opinion will be adapted. This approach "identifies" the input cases and finds the most suitable expert's knowledge to use for the individual case. Similar in their approach, Rahman and Firhurst (1996) use a hierarchical decision-making structure to "filter" the input data; then apply different classifiers in recognition of handwritten characters. They show the multi-expert system outperforms individual experts. Also in character recognition, Tsutsumida et al. (1996) show improvement by using a multi-expert system that utilizes three algorithms. Ng (1990) proposed a multiple knowledge base system architecture that uses mathematical aggregation to reach a consensus of multiple experts.

In the CPG problem, multiple experts were consulted to extract their knowledge. A heuristic involving tests of different combinations of rules is developed through synthesis of different opinions and logical deduction.

2.4 Evolutionary Algorithms

Evolutionary Algorithms (EAs) are based on the principles of biological evolution: genetic inheritance and survival of the fittest. EA is taken from the analogy of a natural system's evolution to model, optimize and solve problems. It applies rules of reproduction, recombination, and mutation to solutions of model systems so that beneficial and survival-enhancing traits are passed on to new solutions. Since its debut in the 1960s, EA has been adapted increasingly in solving complex problems, including optimization problems. Its greatest advantage is robustness: it is applicable in many situations where the goal or constraints change over time (e.g. financial investment problem), where parameter adjustments and fitness measurements are changing (e.g. industrial control problem), and where the search space is discontinuous or multimodal (Goldberg 1994; Schwefel 1997).

The CPG problem is a complex grouping problem with discrete values, non-linear constraints and an objective function that includes a penalty cost that can only be calculated after a solution is formed. These characteristics make efficient solution impractical using conventional optimization methods (e.g. mathematical programming methods). For solution of the CPG problem, EA is robust and open to the incorporation of both existing and new solution methods for further sophistication, specialization and hybridization.

This section presents a brief history of EA, its basic operation, and several issues in constructing an EA.

2.4.1 Three Forms of Evolutionary Algorithms

There are three main forms of evolutionary algorithms, namely, Genetic Algorithms, Evolution Strategies, and Evolutionary Programming. Other techniques based on the evolutionary principle (e.g. Genetic Programming) as well as many hybrid approaches are also called EAs. The three major branches of EAs were developed independently of each other until the early 1990s. The

name “Evolutionary Algorithms” or “Evolutionary Computation” was established in the 1990s after organized efforts of interaction among the various EA research communities. For more background, including the history of EA, see De Jong, Fogel and Schwefel (1997).

The earliest work using an evolutionary process in problem solving can be traced back to the 1950s. However, the basis of the three main forms of EA was not established until the mid-1960s. Evolutionary Programming (EP) was established by Lawrence Fogel in San Diego, California, and the Genetic Algorithm (GA) was developed at the University of Michigan by John Holland. In Europe, Evolution Strategies (ESs) were developed by Bienert, Rechenberg, and Schwefel in Berlin.

As stated above, there are several variants of evolutionary algorithms and also many hybrid systems which comprise various features of these paradigms. However, the structure of any evolutionary method is similar (Bäck 1997; Dasgupta and Michalewicz 1997). Figure 3 shows the basic structure. As will be explained in more detail in the following sections, the order of these steps varies slightly among implementations of evolutionary algorithms.

```
t = 0
initialize P(t)
evaluate P(t)
while (not termination-condition) do
        t = t+1
        select P(t) from P(t-1)
        alter P(t)
        evaluate P(t)
end do
```

Figure 3. Basic Structure of an Evolutionary Algorithm. Adapted from Dasgupta and Michalewicz *(1997)*

The EA maintains a population of individuals, $P(t) = \left\{x_1^t,...,x_\mu^t\right\}$ for each iteration *t*. Each individual represents a solution to the problem and is evaluated to give some measure of its quality or fitness. A selection step based on the fitness measure is performed to obtain the new population of iteration *t+1*. Part of the

new population undergoes the alter step by means of well-defined operators to form new solutions. The computation is terminated when the new generation meets a predetermined quality, a specified number of iterations is reached, or some other termination condition is achieved.

The following is a brief review of the three main forms of EA. The operation of the GA is emphasized since our solution approach is largely drawn from it.

2.4.1.1 Genetic Algorithms

Genetic Algorithms (GAs) typically represent solutions as binary strings. New strings are formed primarily by recombining parts of the current solutions (i.e. a recombination operator). However, mutation is used as a secondary operator (Holland 1975). The following example illustrates the basic representation and operations used by a GA. This example builds on the more general EA structures of Figure 3. In addition, the need to select parameters is highlighted. In this example, the function $f(x)=x^2$, where x is defined between 0 and 31, is to be maximized.

(1) Define a solution as a string of five binary digits or genes. The feasible solutions can be represented as: 00000, 00001, … , 11111. In this context, each solution is called a chromosome.

(2) Set t, the iteration number, equal to 0. Select an initial population of p chromosomes by randomly selecting each of their five digits. Here p is called the population size. An initial population of four chromosomes might consist of the following.

01101
11000
01000
10011

(3) Evaluate each solution through the objective function and use this value to assign the solution's probability of being chosen as a parent so that this

selection probability is related to the quality or fitness of the solution. In this example, the selection probability P(x) is calculated as $P(x)=f(x)/\Sigma f(x)$.

Table 3 shows the objective function value and the probability of being chosen as a parent.

String No.	Initial Population	x value	f(x)	Probability (f/Σf)
1	01101	13	169	0.14
2	11000	24	576	0.49
3	01000	8	64	0.06
4	10011	19	361	0.31

Table 3. GA Example

(4) Set t = t +1 for counting the number of iterations.

(5) Randomly select two parents such that the probability of selection is as shown in Table 3. Suppose the first two parents chosen are 01101 and 11000.

(6) Randomly select a site, *s*, for crossover; suppose we choose *s* = 4.

(7) All genes after site s are switched between the original chromosomes. In this example, s = 4 so only the last gene position is switched. Both of the modified chromosomes are added to a new generation of solutions.

Before Crossover	After Crossover
0 1 1 0 : 1	0 1 1 0 0
1 1 0 0 : 0	1 1 0 0 1

(8) Repeat above steps 5, 6, and 7 to generate more children to form the next generation.

(9) Evaluate the new generation as in step 3. The old generation is discarded in this case.

(10) Repeat steps 4 through 9 until the termination condition is satisfied.

2.4.1.2 Evolution Strategies

Evolution Strategies (ESs) were developed to solve parameter optimization problems (Bäck 1996). A chromosome represents an individual as a pair of real-valued vectors, $\mathbf{v} = (\mathbf{x}, \boldsymbol{\sigma})$. The first vector $\mathbf{x}$ is a point in the search space. The second vector $\boldsymbol{\sigma}$ is a vector of standard deviations. The earliest ESs are based on a population with only one chromosome. Only mutation is used as an operator. The mutation is implemented by replacing $\mathbf{x}$ with $\mathbf{x}^{t+1} = \mathbf{x}^{t} + \mathbf{N}(0, \boldsymbol{\sigma})$, where $\mathbf{N}(0, \boldsymbol{\sigma})$ is vector of independent random Gaussian numbers with a mean of zero and standard deviations $\boldsymbol{\sigma}$. The offspring is accepted as a new member of the population if and only if it has better fitness than its parent and all constraints are satisfied.

Current implementations of ES differ from the initial implementations in terms of how a population is defined and in their control parameters. ESs now use more than one individual in the population. This change was made to provide mechanisms for control parameters (e.g. mutation variance) to self-adapt. Previously, control parameter values were either static or modified by deterministic algorithms. ES implementations with more than one individual use the symbolic notation introduced by Schwefel (Rudolph 1997). The abbreviation $(\mu+\lambda)$-ES denotes that μ individuals produce λ offspring. The new temporary population of $(\mu+\lambda)$ individuals is reduced by a selection process to μ individuals. On the other hand, in (μ, λ)-ES, the μ individuals produce λ offspring and the selection process selects a new population of μ individuals from the set of λ offspring only. In $(\mu+\lambda)$-ESs and (μ, λ)-ESs, the production of the offspring is done by either crossover or mutation. The control parameter σ is no longer a constant; it is incorporated in the structure of the individuals and goes through the evolution process. For more detail, see Schwefel (1995).

The following list is adapted from Michalewicz (1992) to highlight differences between common implementations of ES and GA.

1. ESs were developed for numerical optimization originally (e.g. optimal shapes of bodies in a flow in aerodynamics) and only have been applied to discrete optimization problems recently. On the other hand, GAs were applied in many different domains.
2. In representation, ESs use real-value vectors while GAs typically use binary strings.
3. ESs produce an intermediate population and then apply the selection process. The selection process is deterministic – only the best μ out of $\mu+\lambda$ (($\mu+\lambda$)-ES) or λ ((μ, λ)-ES) individuals are selected. In GA, a selection procedure is applied randomly and repetitively. Strong individuals are likely to be chosen several times while the weakest one also has a chance to survive.
4. The operator parameters differ between ESs and GAs. In GAs, the parameters (e.g. probability of crossover, probability of mutation) are constant while the parameters of ESs change during the evolution process.
5. ESs and GAs enforce constraints differently. In ESs, an offspring is discarded if not feasible since ESs assume a set of inequalities as part of the optimization problem. In GAs, a "penalty cost" is typically added to infeasible solutions so that the probability of retaining a significant number of infeasible solutions in the population is small.

2.4.1.3 Evolutionary Programming

The original Evolutionary Programming (EP) techniques were developed for the purpose of evolving an artificial intelligence with the ability to predict changes in an environment. EP uses the evolution principle to refine the algorithm or solutions with regard to an environment and payoff function. The environment was defined as a sequence of symbols and the evolving algorithm was supposed to produce a new symbol. The output symbol is then evaluated according to some payoff function, which calculates the effectiveness of the algorithm in prediction. EP has a wide range of applications. Common uses include pattern

recognition, system identification and parameter optimization (Porto 1997). It is also utilized in several medical applications, such as optimizing drug design.

For example, with a series of events a_1, a_2, a_3, ...; an algorithm will predict the next event, a_{n+1}. The EP is to evolve such an algorithm. The algorithm usually is represented as a finite state machine (FSM) since FSM offers a meaningful representation of the algorithm based on the interpretation of symbols. Figure 4 shows an example of FSM and its offspring. The nodes represent different states while the edges indicate the transition form one state to another with input and output values (notation *a/b* beside the edge represent the input value of *a* and output value of *b* while the machine operates from state S_1 to state S_2). A population of FSMs representing solutions is maintained within EP. Each FSM is evaluated by exposing to the environment which is represented by a series of symbols. The FSM makes predictions each time and the predicted symbol is compared to the next observed symbol. So, if there are n symbols, the FSM makes n predictions. The fitness value of the FSM is evaluated by taking the overall performance into account such as weighted average of the accuracy of all predictions.

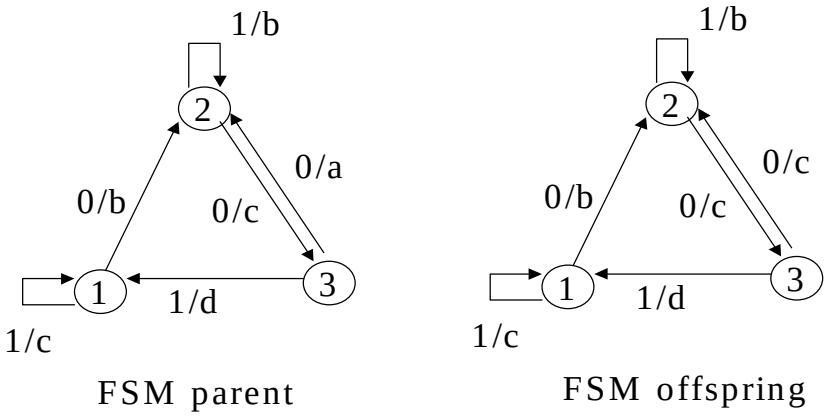

Figure 4. A FSM and its offspring. (Adapted from Dasgupta and Michalewicz *(1997)*).

The EP technique first generates offspring and then selects individuals to form the next generation. Only mutation is used as an operator, so each parent produces a single offspring. The emphasis of EP is on the use of one or more

mutation operations which generate diversity among the population of solutions, but at the same time maintain a high degree of correlation between parent and offspring behavior. For the common FSM representation described above, mutation can take five possible forms (Porto 1997): "(i) change an output symbol; (ii) change a state transition; (iii) add a state; (iv) delete an existing state; and (v) change the initial state." These mutations are chosen with respect to some probability distribution. In strict EP, recombination operator is not used.

Since each individual goes through mutation to produce its offspring, the size of the intermediate population is doubled. The surviving offspring are generally chosen from the intermediate population by choosing the best *m* individuals. Details of selection processes are deferred until section 2.4.3.

Compared to GAs, EP is different in its philosophy of optimization approach (Porto 1997). GAs orient a bottom-up approach by assuming that combining the building blocks will form better solution. On the other hand, EP makes no attempt to attribute credit to individual components of the solutions. Various operators in EP simultaneously modify all variables at the same time. Crossover operators that recombine building blocks are not used.

Compared to ESs, EP also differs in several areas (Michalewicz 1992):

1. EPs do not use recombination operators.
2 EPs use probabilistic selection, whereas ESs select the best μ individuals for next generation.
3. In EP, fitness values are obtained from objective function values by scaling them and possibly by imposing some random alternation.

Though following the same framework, the above three paradigms emphasize different aspects in modeling evolution, and have different strengths. For example (Dasgupta and Michalewicz 1997), one should use ES or GA with real-valued representation for numerical optimization problems. GAs are best for

combinatorial optimization problems. And EP is used most successful in modeling system behavior.

Table 4 summarizes differences between the three major paradigms of EA. These differences are not absolute. For example, GA may use representations other than binary and can achieve self-adaptation. In practice, researchers often add problem specific knowledge to an algorithm making it more effective and efficient. These hybrid approaches are not easily classified into any categories of EAs.

Historically, the development of these different forms of EA suggests that a strength of one algorithm is adapted by the other and vice versa. For example, to deal with the difficulty of performing local search for the numerical applications, researchers may represent a solution through real-value numbers (typical in ES and EP) but use crossover operations typically associated with GA.

In this dissertation, EA is used to describe our solution methodology in order to avoid confusion and loss of generality.

	ES	EP	GA
Representation	Real-valued	Real-valued	Binary-value
Fitness is	Objective function value	Scaled objective function value	Scaled objective function value
Self-adaptation	Standard deviations and rotation angles	None (standard EP), variances (meta-EP), correlation coefficients	None
Mutation	Gaussian, main operator	Gaussian, only operator	Bit-inversion, background operator
Recombination	Discrete and intermediate, sexual and panmictic, important for self-adaptation	None	z-point crossover, uniform crossover, only sexual, main operator
Selection	Deterministic, extinctive or based on preservation	Probabilistic, extinctive	Probabilistic, based on preservation
Constraints	Arbitrary inequality constraints	None	Simple bounds by encoding mechanism
Theory	Convergence rate for special cases, $(1+1)$-ES, $(1+\lambda)$-ES, global convergence for $(\mu+\lambda)$-ES	Convergence rate for special cases, $(1+1)$-EP, global convergence for $(1+1)$-EP	Schema processing theory, global convergence for elitist version

Table 4. Comparison of three forms of EA (Adapted from Bäck *(1996)*)

There is much variation in constructing an EA, depending upon the nature of the problem. Some important decisions involve the following questions:

(1) How would a solution be represented?

(2) What would be a suitable fitness function for evaluating each solution?

(3) How can solution feasibility be assured?

(4) What operations are permissible?

Standard answers to these questions have been superficially discussed in connection with the presentation of the three main EA paradigms. The following section explores these issues more generally.

2.4.2 Representation

Representation in EA is complicated by the problem of “encoding brittleness.” Encoding brittleness means that (Ronald 1997) “slightly different problems require completely different genetic encodings for good solutions to be found.” It is because a solution of a particular problem can be represented in a number of ways, and a search method is most efficient while using a particular representation but not other representations. As the research of EA matures, the robust encoding of EA becomes more important. This section begins with a brief explanation of terminology used in EA; then it presents a review of several important encoding schemes and vital factors in choosing a good solution representation.

2.4.2.1 Terminology Used in EA

Since EA uses the natural evolution process as a search method, it borrows much vocabulary from genetics. The terms population, chromosome and gene have already been introduced. This section defines more advanced terms. In genetics, the genotype is the “abstract collection of genes possessed by an individual” (Stansfield 1991) or “information contained in the genetic code” (Michalewicz 1992). So, genotypes (structures) represent individuals in a population as abstract forms; often these genotypes are called strings or chromosomes. Chromosomes are made up by genes (also called features, characters) arranged in linear order. A gene has a value (allele) and a position (loci) in the chromosome. Each genotype represents a possible solution to the problem. The genotype must be interpreted or decoded to realize the individual. That realized individual with many properties or characteristics influenced by

genes is called the phenotype. The phenotype's performance or fitness within the environment is determined not by its properties alone but also by the environment.

Michalewicz (1992) gives an analogy between nature and a GA illustrating genotype, phenotype and fitness. The lower half of Figure 5 shows that the information encoded within representation strings in a GA as the genotype, the actual solution of the problem as the phenotype, and the fitness value of the solution evaluated by a certain objective function as the individual's fitness within the environment. The upper half of Figure 5 displays the natural world analogy. The chromosome of a flower contains encoded gene information that, when manifested, produces a flower – the phenotype. Flowers are evaluated by the environment where only the most fit are likely to survive. In nature, the mapping between gene and phenotype is complex (a single gene can affect multiple properties in the phenotype, and a single character in the phenotype is controlled by multiple genes.) In EA literature, most implementations use a one-to-one mapping between phenotype and genotype for simplicity.

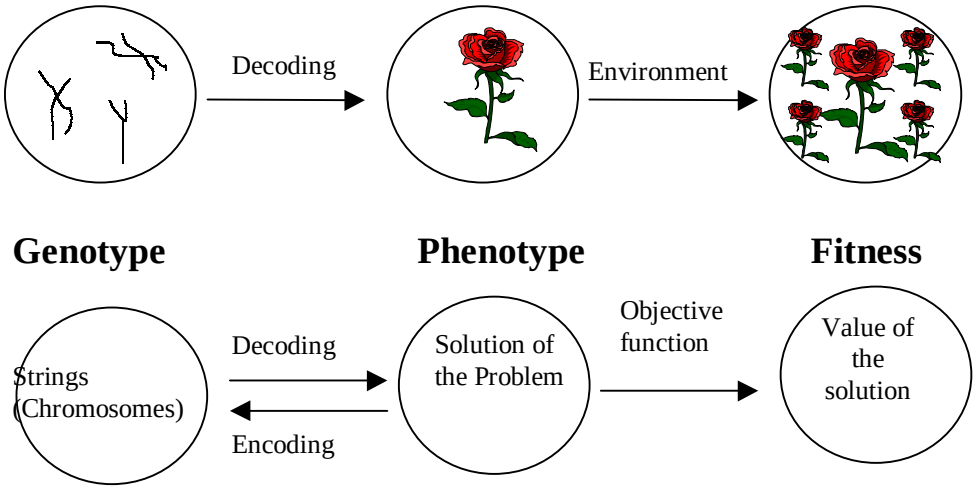

Figure 5. The Analogy of nature and GAs (Adapted from Michalewicz *(1992, p. 28)*).

Another term related to the representation in GAs is "building block". Holland developed a concept called schema to explain the process of a GA. A schema is a similarity template describing a subset of strings with similarities at certain

string positions. In Holland's notation, the symbol * is used to represent an arbitrary substring with zero or more genes. In the example of Section 2.4.1, the schema *0000 matches two strings {10000, 00000}; the schema *111* describes a subset with four members {01110, 01111, 11110, 11111}. Holland has shown that the expected representation of a given schema in the next generation increases or decreases in proportion to its fitness in the current generation (Holland 1975). From this concept, Goldberg (1989) developed the "Building Block Hypothesis". According to Goldberg, building blocks are highly fit, short-defining-length, low-order schemata. Based on Holland's finding, Goldberg suggested that building blocks are propagated from generation to generation with increasing frequency while schemata with lower fitness values appear in a decreasing number of chromosomes. (According to Goldberg, "despite the processing of only n structures each generation, a genetic algorithm processes something like n^3 schemata".) Since the process goes in parallel without extra bookkeeping or memory other than the strings of population, it is called "implicit parallelism".

2.4.2.2 Encoding Schemes

Below the most common encoding schemes are described and their appropriate use explained.

(1) Binary Encoding

Binary encoding is used most often in GA. It is an efficient choice for problems that naturally map into a string of zeros and ones. The coding of the variables in string structures make the search space discrete. In solving a continuous search space problem, the binary representation transforms the problem into a discrete programming problem. In this encoding scheme, many discrete programming problems which are difficult to solve by traditional methods, can be solved conveniently by GAs.

Goldberg (1989, p.80) offered two principles in choosing an encoding scheme: (1) "principle of meaningful building blocks", which is "the user should select a coding so that short, low-order schemata are relevant to the underlying problem

and relatively unrelated to schemata over other fixed positions"; and (2) "principle of minimal alphabets", which said that "The user should select the smallest alphabet that permits a natural expression of the problem". The first principle is to design the encoding scheme conforming to the building block hypothesis. However, designing a coding scheme with meaningful building blocks is an art. The second principle is to exploit the information related to the short schemata and the fitness. It can be shown that binary coding offers the maximum number of schemata per bit of information of any coding.

However, in many situations, a binary encoding is not appropriate. For example:

(i) The value of a bit may restrain the fitness contributions of other bits.

(ii) The problem requires a higher-order than binary symbol set, and these symbols can be arranged together to form necessary schemata (building blocks).

(iii) The genetic operators may produce infeasible solutions with binary encodings or a binary encoding may not describe a solution naturally.

According to the "principle of minimal alphabets" and the building block hypothesis in GA, the smallest alphabet generates the greatest implicit parallelism. However, no empirical evidence indicates that binary codings will be most effective or efficient in solving real-valued optimization problems (Fogel 1997b).

(2) Vector Encodings

ESs and EPs typically operate directly on a real-valued vector. However, vector encodings can also be adapted in GA. For example, a three-dimensional problem, $f(x, y, z)$, a solution might be $f(x_1, y_1, z_1)$ where x_1, y_1, z_1 can be encoded as three binary substrings or as a floating point number.

For real-valued optimization problems, research shows that maximizing the number of schemata is not necessarily useful and floating-point representations outperform binary representations because they are more consistent, more

precise, and produce faster execution (Fogel and Stayton 1994; Michalewicz 1992). This may be evidence for rejecting the building block hypothesis in GAs.

(3) Ordered and Permutations Encodings

Permutation problems seek the optimal arrangement of a set of symbols in a list. For example, the sequence of traveling to five cities can be expressed as x = {4, 3, 0, 1, 2}. Permutations are a popular representation for many combinatorial optimization problems, such as the traveling salesman problem, because they exhibit the following (Ronald 1997):

(i) prevent duplicate or missing gene values

(ii) allow the use of certain high-performance genetic operators, (e.g. edge recombination operator) that use the ordering information in the encoding. (See Whitley (1997) for details of the edge recombination operator).

(iii) have a simple decoding mechanism from the genotype to the phenotype, e.g. in traveling salesman problems the numbers represent the order of the cities being visited.

(4) Other Representations

Many other encoding schemes exist in EA, including: finite-state representations, parse trees, matrices and structured heterogeneous encodings. Finite-state and matrix representation are popular in EP while parse trees are used most in genetic programming. For surveys, see Fogel (1997a) and Angeline (1997).

2.4.2.3 Guidelines for Encoding

The encoding of a complex problem is a difficult problem itself. The following are guidelines for choosing the proper encoding scheme for GAs: (Ronald 1997) "The encoding:

1. embodies the fundamental building-blocks that are important for the problem type;

2. is amenable to a set of genetic operators that can propagate these building blocks from parent genotypes through to the children genotypes during child generation;
3. is not subject to epistasis where the effect of one gene suppresses the action of one or more other genes;
4. allows a tractable mapping to the phenotype, allowing fitness information to be calculated in the minimum number of steps;
5. exploits an appropriate genotype-phenotype mapping process if a simple mapping to the phenotype is not possible;
6. embodies feasible solutions if possible – penalty systems or repair strategies are options if illegal solutions are produced;
7. suppresses isomorphic forms, i.e. many genotypes that map to the one problem point;
8. uses gene values taken from an alphabet of the smallest possible cardinality, with binary encodings considered the best if suited to the problem;
9. represents the problem at the correct level of abstraction ranging from a completely specified point in the problem space or a set of aggregate qualities that describe a family of possible solutions."

The choice for representation is a subjective one. It differs among different investigators. In any event, a suitable representation should be (Fogel and Angeline 1997) "as complex as necessary and no more so, and should explain the phenomena investigated, which here means that the resulting search should be visualizable or imaginable to some extent."

2.4.3 Selection

Selection is the operation that emphasizes better solutions and chooses among individuals to mimic Darwinian survival of fittest. The selection operator does not produce any offspring, but selects better solutions from a population and

discards the rest. The identification of good or bad individuals is usually calculated by a fitness function. The principle of selection is to have a greater probability of selection assigned to an individual with a better fitness value.

Some selection operators choose the best individuals deterministically; some assign a probability of selection to each individual according to fitness value and select offspring using that probability distribution. Either way, the emphasis is on selecting better solutions and, at the same time, avoiding pitfalls such as early convergence to a local optimum, rejecting good solutions, and totally random selection. Several selection operators are discussed in this section.

Selection operators are often described using the term “selective pressure”. Selective pressure is a measurement concept of how much intensity the operator puts on a population to allow only the best individuals to survive. It is often measured by “takeover time” which is “the speed at which the best solution in the initial population would occupy the complete population by repeated application of the selection operator alone” (Deb 1997). The shorter the takeover time, the stronger the selective pressure. Using an operator with a high selective pressure results in losing diversity in the population quickly and should be compensated for by maintaining a large population or employing a highly disruptive recombination and mutation operator. Failure to do so may result in premature convergence to a low quality solution (or a local optimum). On the other hand, a selection operator with low selective pressure allows the recombination and mutation operators to properly search the space but may require excessive computation time.

2.4.3.1 Proportional Selection

Proportional selection is a popular approach first proposed by Holland (1975). This operator selects a number of offspring in proportion to an individual’s fitness. To show it clearly, the selection process can be decomposed into three steps: (Deb 1997)

(i) evaluate individuals according to a “fitness function”,

(ii) create a probability distribution proportional to the fitness value,

(iii) select offspring from the distribution.

The last two steps are relatively straightforward. The probability of selecting any individual is the ratio of its fitness to the sum of the fitness values of all individuals. Selection of offspring is simply drawing individuals from the population according to the probability distribution. This selection scheme is sometimes called “roulette-wheel” selection where each individual takes up a space on the roulette wheel proportional to its fitness. The offspring are selected by “spinning” the roulette wheel as many times as the population size.

The selection process is influenced by the method used to assign a fitness value to each individual. An intuitive approach is to use the objective function value as the fitness value so the space one individual occupies on the roulette wheel is proportional to its objective function value. This approach has some problems.

(i) Negative fitness values are not allowed.

(ii) It cannot be applied to minimization problems directly. Minimization problems must be transformed into maximization problems before using the operator.

(iii) If an individual has an exceptionally high objective value as compared to the rest of the population, it will occupy a very large portion of the roulette wheel. The selecting process is likely to select this “super-solution” most of the time. The result is lost of diversity and the EA may converge to a sub-optimal solution prematurely.

(iv) If most of the individuals in the population have similar objective values, the roulette wheel is divided almost equally among all the members. The selection process becomes random. This often happens later in the execution of EA.

To resolve the above problems, most of the proportional selection operators use some sort of fitness function to transfer the objective function to a non-negative interval, then use a scaling scheme to maintain a productive level of selective

pressure that subdues the “super-solution” effect at the early runs of the EA. Using a scaling scheme can preserve solutions with small differences in their objective function values until late in the run of the EA.

Some important fitness scaling functions include Grefenstette’s time-varying linear transformation of the objective value (Grefenstette 1986), Goldberg’s “Sigma scaling” which is based on the distribution of objective values (Goldberg 1989), and “Boltzmann selection” proposed by de la Maza and Tidor (de la Maza and Tidor 1993).

2.4.3.2 Tournament Selection

In the tournament selection scheme, survival of individuals is determined by “playing a tournament”; i.e., *q* solutions are selected and the best individual is selected either deterministically or probabilistically. After the tournament, all q individuals may be replaced into the population for next tournament or they may not be replaced until several tournaments have been played.

The tournament selection scheme has several advantages:

(i) It can be used in both minimization and maximization problems without change or transformation in the fitness function.

(ii) There is no restriction on negative objective values.

(iii) It is well suited for parallel implementation since it does not require global information (e.g. total of the fitness values, sorting of whole population).

Because of the above advantages, the tournament selection is becoming popular selection scheme in EA research.

2.4.3.3 Rank-Based Selection

Ranking individuals in a population for selection eliminates the need for fitness scaling because selection pressure is kept by the relative ranks among individuals. Rank-based selection assigns the probability of selection according to the ranking of individuals. The ranking can be either linear or nonlinear. In

linear ranking, a selection probability of each individual is proportional to its rank where the rank of the least is zero and the rank of the most fit is μ-*1* given μ as the population size. In linear ranking, the selection operator is designed to avoid premature convergence caused by a "super-solution" so that the expected number of offspring of the best individual is less than twice the expected number of the population average. In nonlinear ranking, the selection probabilities are based on individual's rank but not proportionally. It may be proportional to the square of the rank or even more aggressive. In this case, the selection pressure is much greater than that in the linear ranking scheme. The risk of premature convergence must be compensated for by more disruptive recombination or mutation operators.

Rank-based selection has several advantages:

(i) It can prevent premature convergence caused by a "super-solution".

(ii) It simplifies the mapping of objective function to the fitness function.

(iii) It can be used in situations when the precise objective function is difficult to specify (e.g. subjective preference for different solutions.)

2.4.3.4 Other Selection Methods

There are many other variations of selection operators that are different in certain operations but follow the same concept of the above schemes. These include soft brood selection (Altenberg 1994), Boltzmann selection and disruptive selection.

2.4.3.5 Overlapping/Non-overlapping Population and Elitist Strategies

In selection, there are two competing schemes with regard to different generations, namely overlapping and non-overlapping populations. In a non-overlapping scheme, parents and offspring never compete with one another; i.e. the entire parent generation is replaced by the offspring generation. On the other hand, an overlapping population forces the parents and offspring to compete for

survival. The amount of overlap between parents and offspring is called the “generation gap”.

A selection operation in an EA involves a selection pool and a selection distribution of that pool. A selection pool is required in selecting for reproduction and for deletion. For reproduction, parents are selected according to one of the schemes explained above. In deletion, individuals are removed from a population to make room for new offspring. In a non-overlapping population scheme, the entire parent generation is selected for deletion. In an overlapping scheme, the selection pool consists of parents and offspring. Selection for deletion is processed on the combination population. Early implementations of EP and ES used overlapping populations while early implementations of GA used non-overlapping populations; the distinction is no longer universal.

Two systems, steady state and generational EAs, represent these two typical schemes. Steady state systems usually produce one or two offspring in an operation. To make room for the new offspring, some individuals are selected for deletion in the population that may contain parents only or which may be augmented with offspring. This is an overlapping population system since not all the parents are deleted from the population. On the other hand, generational systems have the entire parent population replaced by the offspring population. It is a non-overlapping system.

Individuals have a lifetime of one generation in simple generational EAs. In order to preserve good individuals longer than one generation, “elitist strategies” can be used to extend the lifetime of certain individuals. The strategy can be useful in both overlapping and non-overlapping systems. In an overlapping system, the combined population is usually ranked according to fitness and then truncated to form the new population. This method ensures that individuals with higher fitness survive. In generational systems where all parents are replaced by their offspring, there is no guarantee that better individuals will survive into the next generation. An elitist strategy in generational genetic algorithms is the method used to preserve the best individual in every generation. Usually, in generational

GA, only the best individual survives while in EP and ES more than just the best survive. One study shows that preserving more than one best individual (multiple elitist) in GA may yield better results than single elitist strategy (Soremekun et al. 1996).

2.4.4 Variations of Recombination

Recombination mechanisms use two (or more) parents from the population to create a new individual. In GA, the most common recombination operator is crossover. In a natural biological system, crossover is a complex process that occurs between pairs of chromosomes. In EA, crossover may take on many forms depending upon the problem and representation scheme.

Holland (1975) introduced a three-step procedure that forms the basis for many types of crossover operation.

(i) Two individuals are chosen from the population of parent strings.

(ii) One or more string locations are chosen as crossover points. The points define the string segments to exchange.

(iii) Parent string segments are exchanged to produce two offspring.

2.4.4.1 Single Point Crossover

Single point crossover was introduced by Holland. One crossover point is chosen randomly and the bits to the right of that point between two parents are exchanged. It was illustrated in Section 2.4.1.

The traditional single-point operator is rarely used in practice because of inferior performance compared to other crossover operators. Research shows that single-point crossover suffers from “positional bias”, i.e., the bit toward to the right in the string has higher exchange probability.

2.4.4.2 n-Point Crossover

This operator uses the number of crossover points as a parameter. The n-point crossover exchanges each second segment between subsequent positions.

Figure 6 shows 3-point crossover. The value n = 2 is commonly used in the literature. There is no clear theoretical or empirical consensus concerning the ideal number of crossover points to use while n ≥ 3.

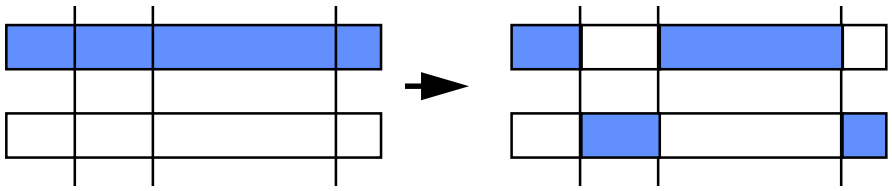

Figure 6. 3-point Crossover

2.4.4.3 Uniform Crossover

With this operator, each bit is randomly and independently exchanged or not exchanged. Therefore, the number of crossover points is not determined in advance. The probability of crossover is usually set to 0.5; however, some study shows advantage in using smaller values (Spears and De Jong 1991).

2.4.4.4 Punctuated Crossover

This operator is one of the attempts to incorporate self-adaptation in GA in the same way as in ES and EP. It uses a binary string of “punctuation marks” to indicate the location of crossover points for a multipoint crossover operation. The string is appended to the chromosome to carry extra information. The appended string is also subject to crossover and mutation. Although the self-adaptation idea behind the operator is intuitively appealing, empirical studies do not show any conclusive advantage over traditional methods (Booker et al. 1997).

There are also many other crossover operators (e.g. segmented crossover, shuffle crossover, etc.) In practice, details within these operators are often modified to enhance performance. The crossover operators described above are all following the principle of Mendelian inheritance, i.e., (Booker et al. 1997) “the requirement that every gene carried by an offspring is a copy of a gene inherited from one of its parents.” Some researchers indicated that this need not be the case for artificial recombination. For robust exploration of the opportunities

represented by the parents, it is important to note that single-point and n-point crossover cannot generate all possible offspring that have some combination of genes in the parents while shuffle crossover and uniform crossover can. Also, the large number of different crossover operators in empirical studies suggests that, in some cases, the crossover operator may not yield satisfactory performance. Some results even suggest that the dominating role of crossover in GA may give way to mutation in the future (Bäck 1996). For a more extensive review of recombination mechanisms, see Booker, Fogel, Whitley, and Angeline (1997).

2.4.5 Variations to Avoid Infeasible Solutions

Infeasible individuals are often encountered during some stage of the evolution process. For example, the simple GA illustrated in the example of section 2.4.1 has no means of assuring feasibility if applied to a problem with constraints. Some mechanism must be added to the simple GA in these cases. It is also true that many encoding schemes cannot rule out the production of infeasible individuals because of problem constraints. Traditionally, EAs applied to constrained optimization problems follow one of two paradigms:

1. modification of the genetic operators, or
2. penalizing individuals that violate constraint(s).

Recently, many heuristics have been proposed to handle the feasibility issue. Dasgupta and Michalewicz (1997) summarized these into different categories:

2.4.5.1 Rejection of Infeasible Individuals

This method eliminates all infeasible solutions from the population. It is popular in many techniques (e.g. ESs). The advantage of this method is simplicity: there is no need to evaluate infeasible solutions or compare them to feasible ones. It also may work well if the search space is convex. However, in many situations, this approach has a serious limitation; the system may reach the optimum

solution more efficiently by “crossing” an infeasible region, especially in non-convex feasible search spaces.

2.4.5.2 Penalizing Infeasible Individuals

The most common way that constraints are enforced in EA optimization is by penalizing infeasible solutions. The major problem in this approach is determining how the penalty should be applied. Many factors influence the performance of the penalty function. So, the choice of such a function may depend on “(1) the ratio between sizes of the feasible and the whole search space, (2) the topological properties of the feasible search space, (3) the type of the objective function, (4) the number of variables, (5) the number of constraints, (6) the types of constraints, and (7) the number of active constraints at the optimum.” (Dasgupta and Michalewicz 1997, p. 19)

Some guidelines were proposed by researchers on the study of penalty functions design: (Richardson et al. 1989; Siedlecki and Sklanski 1989)

1. “penalties which are functions of the distance from feasibility are better performers than those which are merely functions of the number of violated constraints,
2. for a problem having few constraints, and few full solutions, penalties which are solely functions of the number of violated constraints are not likely to find solutions,
3. good penalty functions can be constructed from two quantities, the maximum completion cost and the expected completion cost,
4. penalties should be close to the expected completion cost, but should not frequently fall below it. The more accurate the penalty, the better will be the solutions found. When penalty often underestimates the completion cost, then the search may not find a solution”;
5. “the genetic algorithm with a variable penalty coefficient outperforms the fixed penalty factor algorithm.”

2.4.5.3 Maintaining a Feasible Population by Special Representations and Genetic Operators

Using problem specific heuristics to maintain feasible solutions is a popular trend. Many specialized systems have been developed for particular optimization problems using a unique chromosomal representations and specialized “genetic” operators which alter their composition. In many areas, problem-specific representations and operators result in very successful evolutionary algorithms. For examples of such systems, see Davis (1991).

2.4.5.4 Repair of Infeasible Individuals

For many combinatorial optimization problems (e.g. traveling salesman problem, knapsack problem, etc.) repairing an infeasible individual is relatively easy. The repaired individuals can be used in evaluation only, or they can replace the original individuals. The disadvantage of this method is that it is problem dependent. For each particular problem, a specific repair algorithm must be designed, and there are no standard heuristics for the design of such algorithms. For other problems (e.g., nonlinear transportation problem, scheduling problems), the process of repairing infeasible individuals may be very complex. However, the recently developed Genocop III system which uses repair algorithms and maintains two populations shows promising results (Michalewicz and Nazhiyath 1995).

2.4.5.5 Use of Decoders

In using the techniques of decoders, a chromosome is interpreted in such a way that it can be decoded to a feasible solution. For example, a sequence of items for the knapsack problem can be interpreted as “add an item if possible"; such interpretation will always lead to feasible solutions. Several conditions should be satisfied to use decoders: (Palmer and Kershenbaum 1994) (1) there should be a one-to-one relationship between each decoded solution and the corresponding feasible solution, (2) the transformation should be computationally fast, and (3) it should have the locality feature , i.e. small changes in the decoded solution result in small changes in the solution itself.

2.4.6 Variations of Using Problem Structure to Improve Efficiency

To improve the performance of a EA, it is natural to consider combining EA with other existing optimization techniques, such as linear programming, integer programming, dynamic programming, branch and bound, as well as heuristics specifically developed for the given problem. Some popular hybrid approaches are presented below.

2.4.6.1 Local Search

GA combined with local search has been proposed from the beginning. Numerous applications can be found in the literature (Ibaraki 1997). The local search improves a solution by searching its neighborhood. The neighborhood is a set of solutions that are usually obtained by perturbing its components in specific ways. The solutions in the neighborhood can be searched randomly or systematically. The adaptation of a solution in the neighborhood can be the first improved solution found or the local optimum.

The adaptation of local search in GA is called Genetic Local Search (GLS) (Ibaraki 1997). Local search is usually performed before the selection step in GA. GA's ability to capture an entire solution space and local search's sharp optimization make the combination a powerful algorithm. GLS has been adapted in many problems, especially in combinatorial optimization problems such as Traveling Salesman Problem.

2.4.6.2 Use of Problem-Specific Heuristics

Many real world problems fall into the category of NP-hard problems (Garey and Johnson 1979). For NP-hard problems, it is reasonable to develop heuristics that will get good suboptimal solutions in a short time. In combining GA/EA with problem-specific heuristics, usually, heuristics are used in producing an initial population or to fix infeasible solutions, while GA diversifies the search direction. The combination approach has produced very good results in many popular

optimization problems such as knapsack problems, job-shop scheduling problem, etc.

2.4.6.3 Combination with Other Methods

EA can also be combined with other search methods such as simulated annealing (SA), tabu search, and dynamic programming. Consider simulated annealing as an example. SA is a search method based on random local search. It is different from traditional local search in that it allows worse solutions to be accepted by a non-zero probability. The acceptance probability is based on the "temperature"; when the temperature is higher, the probability of accepting worse solutions is higher. The search procedure starts with highest temperature, then gradually reduces the temperature.

SA and GLS share many similarities but differ in the following ways:

(i) SA maintains only one solution while GLS maintains a population of solutions.

(ii) GLS uses genetic operators and local search to produce and select offspring while SA uses random local search to generate a candidate solution and may accept a new worse solution with a non-zero probability.

SA and GLS may be combined by maintaining a population of solutions, generating new solutions using crossover, mutation and local search, and selecting offspring by schemes of SA and GA. Some studies have shown very good results (Chen and Flann 1994).

2.4.7 Parameter Selection Methods

Given a problem, the researchers of EAs face the task of selecting operator and control parameter values to produce the best result. These decisions include the parent selection mechanism, crossover and mutation operators as well as the corresponding parameters (e.g. crossover probability, mutation probability, and population size). The proportion of parents that experience crossover within one

generation is defined as the crossover rate. Setting the crossover rate depends on the choices made regarding other aspects of the overall algorithm including the population size and mutation rate. Usually, the crossover rate is set between 0.45 and 0.95. The decision may be based on (Freisleben 1997):

"- systematically checking a range of operators and/or parameter values and assessing the performance of the EA

- the experiences reported in the literature describing similar application scenarios

- the results of theoretical analyses for determining the optimal parameter settings."

None of the above approaches can guarantee satisfactory results in any given case, not to mention universal validity. So, the search for the best EA for a given problem is an optimization problem itself. This is called "metaoptimization problem". Several optimization methods have been proposed in the literature to solve this metaoptimization problem, including simulated annealing and tabu search. The approach of using EA to solve the metaoptimization is called "metaevolutionary approach". In any event, study (Hart and Belew 1991) has shown that there is no universally valid optimal parametrization of a GA because the optimal parameter values are strongly dependent on the particular optimization problem, its representation, population model, and genetic operators used in the GA.

Although finding an optimal set of parameter values may not be possible, several theoretical studies on simple function optimization problems can be used to suggest the population sizes, mutation probabilities (Hesser and Männer 1990), crossover probabilities, and the relationships between operators (Freisleben 1997).

Table 5 shows some empirically determined GA parameter settings that may be useful starting points in practice. Gates et. al. (1995) suggest a heuristic in choosing optimal genetic operators setting in a simple GA case: "

1. Choose a relatively small population size (10 - 30).
2. Start with a very small mutation rate (one that will make the GA converge prematurely.
3. Run the GA.
4. When convergence is detected, stop.
5. Increase the mutation rate slightly.
6. Repeat from step 3 until you have exhausted the preset number of function evaluations."

Author	Population Size	Crossover Rate	Mutation Rate
Schaffer	20-30	0.75 - 0.95	0.005 - 0.01
De Jong	50 - 100	0.60	0.001
Grefenstette	30	0.95	0.01

Table 5. Empirically determined GA parameter settings (Adapted from *(Gates et al. 1995)*)

2.5 Summary

This chapter reviews several models related to the CPG problem either in the problem domain or solution approaches. The CPG problem has some characteristics similar to the cutting and packing problems found in many industries, especially in terms of the dimensionality and the objective of avoiding cutting waste. Upon further investigation, however, CPG's emphasis on developing plates so that the lowest unit cost is achieved and its lack of a constrained stock size limit the usefulness of the existing solution approaches . Nevertheless, the heuristic methods used in the assortment problem serve as a good reference in synthesizing the knowledge of multiple experts, which is the first attempt in solving the CPG problem.

In further analysis of the CPG problem, its nature of grouping small elements points to the problem domain of grouping problems. The most prominent type of grouping problem is called the cluster problem, which seeks to group similar

elements so that elements within the same group are relatively homogeneous and elements in different groups share little resemblance. Since grouping similar elements is not the objective of the CPG problem, the solution approach applied to the cluster problem (i.e. cluster analysis) is not favorable. However, the mathematical formulation with its matrix representation in the integer programming approach toward cluster problem is adopted and modified in the EA solution approach.

Two solution techniques adopted in this dissertation are also reviewed. The first technique involves developing an expert system based on multiple experts' knowledge. The advantages and challenge of multi-expert systems are discussed in this chapter. Finally, the second technique, EA, with its different forms and design consideration is investigated. The XS approach is the first attempt in solving the CPG problem. It preserves the knowledge of multiple experts and has many advantages of expert systems. On the other hand, the EA approach is more robust and may be applicable in other grouping problems with different constraints and objective functions. Both approaches are presented in the next chapter.

CHAPTER 3 CPG PROBLEM AND SOLUTION METHODOLOGY

In this chapter, the CPG problem is described in detail and formulated mathematically. Small examples are provided to illustrate the problem and the mathematical model. As stated in the previous chapter, existing solution techniques are not suitable for solving the CPG problem. In this dissertation, an expert system and an evolutionary algorithm are proposed as CPG solution approaches. These approaches are created by synthesizing knowledge of multiple experts and by drawing techniques from different modeling and solution methodologies. Both approaches are described in detail. This chapter also explains the experiment designed to investigate the efficiency and effectiveness of the solution approaches as well as the effects of different EA parameters.

3.1 Problem Description

As stated in Chapter 1, the cutting plan specifies the raw material to order and how the raw material should be cut and processed into building construction components. The raw material is a steel plate. It is available from vendors in either standard or made-to-order sizes. There are a small number of standard sizes for a steel plate. Industry experience shows that if the standard sizes are used, the cutting waste is approximately 15% - 20%. The waste is much greater than that expected from made-to-order plates. Since there is no reduction in unit cost for using a standard size plates, the cutting waste represents a very significant additional cost. Made-to-order plates, with their lower waste, also save cost by reducing the need for waste processing and inventory handling. In this dissertation, only made-to-order plates are considered.

The construction components necessary for the building project determine the steel element requirements. Consider, as an example, the H-beam illustrated in Figure 7. The dimensions describing an H-beam are typically listed in the form: H $\times$ B $\times$ t1 $\times$ t2 $\times$ L (where H is the height, B the width, t1 the thickness of web, t2 the thickness of flange, and L the length). The convention for steel plate

dimension specification is Thickness × Width × L. As illustrated in Figure 7, the required elements for building an H-beam are one piece of plate with the size of t1 × (H-2t2) × L and two pieces of plate with the size of t2 × B × L.

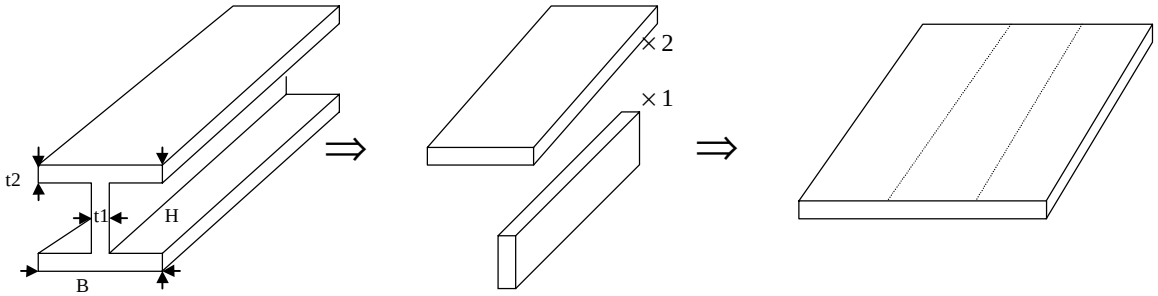

Figure 7. The dimensions of an H-beam are shown along with how such a construction component can be separated into elements and then reformed into a steel plate.

In preparing the ordering quantity and the specifications of raw materials (in this case, steel plate), the engineers are limited by the minimum and maximum width plate available from the vendor. In addition, they must know unit prices for different types of steel. Typically, the unit price of a type of steel with a given thickness is a step function of its width; prices are updated periodically by the vendor. Some steel prices have a "convex" shape with higher prices for pieces which are either "too wide" or are "too narrow." Table 6 is an example of the unit price for steel that varies in this way. Table 7 shows the unit prices for another type of steel that increases in price as the width increases. While the unit price of each steel type varies by thickness, this information is not used to help form economical plates because it is not practical to combine the thickness of required steel elements into one plate. In all cases, an additional "penalty cost" is added by vendors for plates of the same size which are too small as measured by weight.

T: Thickness W: Width (in mm)	1219 ≤W< 1524	1524 ≤W< 1829	1829 ≤W< 2000	2000 ≤W< 2500	2500 ≤W< 2750
35.0<T≤36.0	12.64	12.36	12.36	12.27	12.38
36.0<T≤40.0	12.84	12.56	12.38	12.27	12.38
40.0<T≤50.8	13.10	12.84	12.66	12.56	12.66

Table 6. Example[5] showing material unit prices for SS330 varying by thickness and width. All prices are in New Taiwan Dollars (NT$) per Kg.

T: Thickness W: Width (in mm)	1524 ≤W< 2000	2000 ≤W< 2500	2500 ≤W< 2750	2750 ≤W< 3000	3000 ≤W< 3300
28.0<T≤35.0	11.85	11.90	11.95	12.00	12.20
35.0<T≤50.8	11.95	12.00	12.05	12.10	12.30
50.8<T≤65.0	12.05	12.10	12.15	12.20	12.40

Table 7. Example showing material unit prices for SS400 increasing by thickness and width. All prices are in NT$ per Kg.

As has been explained, the engineers arrange the components to form a plate order with low unit cost and low cutting loss. Figure 8 illustrates four different scenarios of how pieces could be cut from steel plates. The labels UB and LB indicate the upper and lower bound on the plate width associated with the lowest unit cost. Notice that only the arrangement on the right side of this figure takes advantage of the lowest possible unit price and avoids cutting loss associated with combining elements of different lengths in the same plate.

[5] Source: China Steel Corporation.

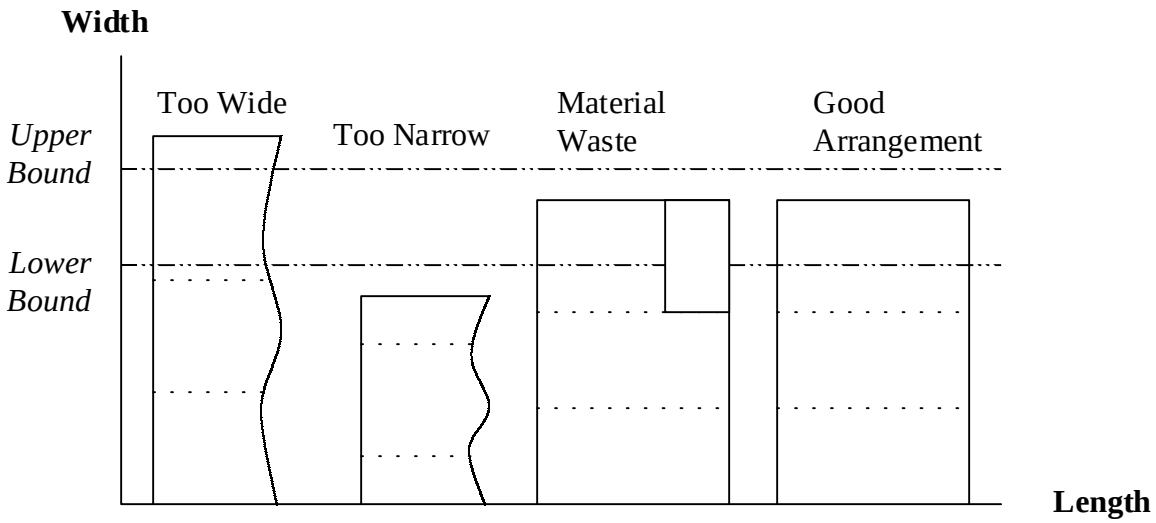

Figure 8. Construction elements should be arranged so that the plate width is between the lower and upper bound associated with the lowest cost per unit and so that no cutting loss is present. Above are three poor arrangements and one efficient one.

The following example will further clarify this problem. Consider a company that needs three H-beams with the following dimensions (H $\times$ B $\times$ t1 $\times$ t2 $\times$ L in mm)

$530 \times 400 \times 40 \times 40 \times 8000$

$780 \times 500 \times 40 \times 40 \times 8000$

$780 \times 500 \times 40 \times 40 \times 10000$

This company must order steel plates, cut them into elements and weld them into H-beams with the required dimensions. The first H-beam is decomposed into three elements (thickness $\times$ width $\times$ L):

$40 \times 400 \times 8000$

$40 \times 400 \times 8000$

$40 \times 450 \times 8000$

Similarly, the second and the third components are also decomposed. The resulting required elements are listed in Table 8. (In all cases, the thickness is 40 mm.)

Element	1	2	3	4	5	6	7	8	9
Width (mm)	400	400	450	500	500	700	500	500	700
Length (mm)	8000	8000	8000	8000	8000	8000	10000	10000	10000

Table 8. Required Elements

Assume the unit prices of the steel plates vary according to different widths and thicknesses as listed in Table 9. As stated above, assume there is a penalty unit cost NT$ 1 per Kg added if the total weight of all plates of a specific size is less than 2600 Kg. Given the requirement and the unit cost, the CPG problem in this example is to group the nine elements to form the ordering plates that satisfy the demand and minimize the total material cost.

T: Thickness W: Width (in mm)	**1524 ≤W< 2000**	**2000 ≤W< 2500**	**2500 ≤W< 2750**	**2750 ≤W< 3000**	**3000 ≤W< 3300**
28.0<T≤35.0	11.85	11.90	11.95	12.00	12.20
35.0<T≤50.8	11.95	12.00	12.05	12.10	12.30
50.8<T≤65.0	12.05	12.10	12.15	12.20	12.40

Table 9. Material unit prices for SS400 increasing by thickness and width. All prices are in NT$ per Kg.

3.2 Mathematical Model and Example

The CPG problem can be formulated mathematically. Adopted from the integer programming model discussed in section 2.2.5, a matrix representation is used to denote the group membership of each element. The mathematical model is presented and explained below.

Minimize

$$\sum_{m=1}^{M} U_m T_m \tag{1}$$

Subject to

$$T_m = W_m L_m HD \tag{2}$$

$$L_m = Max(x_{m1} l_1, x_{m2} l_2, ..., x_{mN} l_N) + XL \tag{3}$$

$$W_m = \sum_{n=1}^{N} x_{mn} w_n + \mathrm{M}ax\left(0, 3\left(\sum_{n=1}^{N} x_{mn} - 1\right)\right) \tag{4}$$

$$U_m = \begin{cases} Q_k & \text{if } S_m > G \\ Q_k + \mathrm{P} & \text{if } S_m \le G \end{cases} \quad \text{where } Q_k \text{ is the corresponding unit cost} \tag{5}$$

$$A_{\min} \le W_m < A_{\max} \tag{6}$$

$$S_m = \sum_{r=1}^{M} T_m y_{rm} \tag{7}$$

$$y_{rs} = \begin{cases} 1 & \text{if } L_r = L_s \text{ and } W_r = W_s \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

$$x_{mn} = \begin{cases} 1 & \text{if element } n \text{ is a member of plate } m \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

$$\sum_{m=1}^{M} x_{mn} = 1 \text{ for all } n = (1, N) \tag{10}$$

$$M = \begin{cases} \left\lfloor \dfrac{\left(\sum_{n=1}^{N} w_n\right) + 3N}{A_{\min}} \right\rfloor & \text{if } \left(\sum_{n=1}^{N} w_n - A_{\min} \left\lfloor \dfrac{\left(\sum_{n=1}^{N} w_n\right) + 3N}{A_{\min}} \right\rfloor \right) \le A_2 - A_1 \\ \left\lceil \dfrac{\left(\sum_{n=1}^{N} w_n\right) + 3N}{A_{\min}} \right\rceil & \text{otherwise} \end{cases} \tag{11}$$

Variables:

C = Total material cost

B_m = Material cost of plate m

U_m = Unit cost of plate m

T_m = Weight of plate m

W_m = Width of plate m

L_m = Length of plate m

x_{mn} = 1 if element n is assigned to plate m; 0 otherwise

w_n = Width of element n

l_n = Length of element n

S_m = Total weight of plates that have the same size of plate m

y_{rs} = 1 if plate r and plate s have the same size; 0 otherwise

Parameters:

H = Thickness of plate

D = Density of steel

P = Unit penalty cost

XL = Extra margin for cutting along dimension of the length

A_k = Widths of unit cost table

A_{max} = Maximum width of plate allowed

A_{min} = Minimum width of plate allowed

Q_k = Unit cost corresponding to plate with width between A_k and A_{k+1}

G = Minimum weight to avoid penalty cost

M = Maximum number of plates possible

N = Number of required elements.

Indices:

m = Plate index

n = Element index

r, s = Index for the matrix of the same dimension

k = Index for the unit cost table

Following is an explanation of the objective function and constraints.

(1) The objective is to minimize the material cost of all plates.

(2) The plate's weight is the product of its volume and density.

(3) The length of plate *m* is the maximum length of its elements plus an extra technological factor for production feasibility.

(4) The width of plate *m* is the total width of its elements, plus the trimming loss[6] between elements.

(5) When the total weight of all plates of a given size exceeds G, the unit cost is determined by its width according to the unit cost table. Otherwise, an extra penalty unit cost P is added.

(6) The width of any plate must be within vender specified (technology) limits.

(7) The total weight of same size plates is the sum of the individual plates of the size.

(8) The symmetric matrix ***y*** is used to denote whether two plates are of the same size. If plate *r* has the same size as plate *s*, then y_{rs} within matrix ***y*** is 1. Otherwise it is 0.

(9) Matrix ***x*** is used to denote the group setting that forms the plates. If element *n* belongs to group *m* (i.e. element *n* is part of plate *m*), then x_{mn} in matrix ***x*** equals to 1. Otherwise it is 0.

(10) Each element must be assigned to one and only one plate.

[6] Plate measurements always include the width of the elements assigned to the plate plus any additional width lost due to the cutting process. Depending up different thickness (usually thinner than 28 mm), the cutting process loss is 10 mm on each side of the plate plus 3 mm between elements. Plates with this extra width are called "cut edge" plates. Plates thicker than 28 mm are also subject to 3 mm cutting loss between elements but do not require the additional 10-mm edges and are referred to as "mill edge" plates. These material losses are required by the technology and will not be explicitly re-stated in this section.

(11) The maximum number of plates is calculated by equation (11). This number will influence the size of the problem being formulated. It is desirable to obtain the value of this parameter as small as possible; so the problem size will be as small as possible.

To calculate the value of *M*, an expected number of plates is calculated first. This number, *SP*, is calculated first by dividing the total width plus possible cutting margins by the minimum width allowed. The round-down integer of this division is the expected number of plates. Without considering the residual of the division, ordering SP plates, even with the minimum width, will satisfy the demand.

If the residual after forming those suppositional plates is larger than (A_2 - A_1), the range of the first increment in the cost table, then another plate may be needed to reduce the width and avoid higher unit cost. In this case, the round-up integer of the division is used.

To further illustrate the mathematical formulation of the problem, the previous example is formulated below. The maximum number of plates possible is calculated first to determine the value of *M*. In this example, M = 3. Other parameter values include

H = 40 mm

D = 0.00000785 kg/mm^3

P = 1 NT$/kg

A_{max} = A_5 = 3000 mm

A_{min} = A_1 = 1524 mm

A_2 = 2000 mm, A_3 = 2500 mm, A_4 = 2750 mm

Q_1 = 11.95 NT$/kg, Q_2 = 12.00 NT$/kg, Q_3 = 12.05 NT$/kg,

Q_4 = 12.10 NT$/kg, Q_5 = 12.30 NT$/kg

G = 2600 kg

N = 9

Minimize

$$\sum_{m=1}^{3} U_m T_m$$

Subject to

$$T_1 = W_1 L_1 \times 40 \times 0.00000785$$
$$T_2 = W_2 L_2 \times 40 \times 0.00000785$$
$$T_3 = W_3 L_3 \times 40 \times 0.00000785$$

$$L_1 = \mathrm{M}ax(8000x_{11}, 8000x_{12}, 8000x_{13}, 8000x_{14}, 8000x_{15}, 8000x_{16}, 10000x_{17}, 10000x_{18}, 10000x_{19}) + 20$$
$$L_2 = \mathrm{M}ax(8000x_{21}, 8000x_{22}, 8000x_{23}, 8000x_{24}, 8000x_{25}, 8000x_{26}, 10000x_{27}, 10000x_{18}, 10000x_{19}) + 20$$
$$L_3 = \mathrm{M}ax(8000x_{31}, 8000x_{32}, 8000x_{33}, 8000x_{34}, 8000x_{35}, 8000x_{36}, 10000x_{37}, 10000x_{38}, 10000x_{39}) + 20$$

$$W_1 = 400x_{11} + 400x_{12} + 450x_{13} + 500x_{14} + 500x_{15} + 700x_{16} + 500x_{17} + 500x_{18} + 700x_{19}$$
$$+ Max(0, 3(x_{11} + x_{12} + x_{13} + x_{14} + x_{15} + x_{16} + x_{17} + x_{18} + x_{19} - 1))$$
$$W_2 = 400x_{21} + 400x_{22} + 450x_{23} + 500x_{24} + 500x_{25} + 700x_{26} + 500x_{27} + 500x_{28} + 700x_{29}$$
$$+ Max(0, 3(x_{21} + x_{22} + x_{23} + x_{24} + x_{25} + x_{26} + x_{27} + x_{28} + x_{29} - 1))$$
$$W_3 = 400x_{31} + 400x_{32} + 450x_{33} + 500x_{34} + 500x_{35} + 700x_{36} + 500x_{37} + 500x_{38} + 700x_{39}$$
$$+ Max(0, 3(x_{31} + x_{32} + x_{33} + x_{34} + x_{35} + x_{36} + x_{37} + x_{38} + x_{39} - 1))$$

$$U_1 = 11.95 \text{ if } 1500 \le W_1 < 2000 \text{ and } S_1 > 2600$$
$$U_1 = 12.00 \text{ if } 2000 \le W_1 < 2500 \text{ and } S_1 > 2600$$
$$U_1 = 12.05 \text{ if } 2500 \le W_1 < 2750 \text{ and } S_1 > 2600$$
$$U_1 = 12.10 \text{ if } 2750 \le W_1 < 3000 \text{ and } S_1 > 2600$$
$$U_1 = 12.30 \text{ if } 3000 \le W_1 < 3300 \text{ and } S_1 > 2600$$

$$U_2 = 11.95 \text{ if } 1500 \le W_2 < 2000 \text{ and } S_2 > 2600$$
$$U_2 = 12.00 \text{ if } 2000 \le W_2 < 2500 \text{ and } S_2 > 2600$$
$$U_2 = 12.05 \text{ if } 2500 \le W_2 < 2750 \text{ and } S_2 > 2600$$
$$U_2 = 12.10 \text{ if } 2750 \le W_2 < 3000 \text{ and } S_2 > 2600$$
$$U_2 = 12.30 \text{ if } 3000 \le W_2 < 3300 \text{ and } S_2 > 2600$$

$U_3 = 11.95$ if $1500 \le W_3 < 2000$ and $S_3 > 2600$
$U_3 = 12.00$ if $2000 \le W_3 < 2500$ and $S_3 > 2600$
$U_3 = 12.05$ if $2500 \le W_3 < 2750$ and $S_3 > 2600$
$U_3 = 12.10$ if $2750 \le W_3 < 3000$ and $S_3 > 2600$
$U_3 = 12.30$ if $3000 \le W_3 < 3300$ and $S_3 > 2600$

$U_1 = 12.95$ if $1500 \le W_1 < 2000$ and $S_1 \le 2600$
$U_1 = 13.00$ if $2000 \le W_1 < 2500$ and $S_1 \le 2600$
$U_1 = 13.05$ if $2500 \le W_1 < 2750$ and $S_1 \le 2600$
$U_1 = 13.10$ if $2750 \le W_1 < 3000$ and $S_1 \le 2600$
$U_1 = 13.30$ if $3000 \le W_1 < 3300$ and $S_1 \le 2600$

$U_2 = 12.95$ if $1500 \le W_2 < 2000$ and $S_2 \le 2600$
$U_2 = 13.00$ if $2000 \le W_2 < 2500$ and $S_2 \le 2600$
$U_2 = 13.05$ if $2500 \le W_2 < 2750$ and $S_2 \le 2600$
$U_2 = 13.10$ if $2750 \le W_2 < 3000$ and $S_2 \le 2600$
$U_2 = 13.30$ if $3000 \le W_2 < 3300$ and $S_2 \le 2600$

$U_3 = 12.95$ if $1500 \le W_3 < 2000$ and $S_3 \le 2600$
$U_3 = 13.00$ if $2000 \le W_3 < 2500$ and $S_3 \le 2600$
$U_3 = 13.05$ if $2500 \le W_3 < 2750$ and $S_3 \le 2600$
$U_3 = 13.10$ if $2750 \le W_3 < 3000$ and $S_3 \le 2600$
$U_3 = 13.30$ if $3000 \le W_3 < 3300$ and $S_3 \le 2600$

$1500 \le W_1 < 3300$
$1500 \le W_2 < 3300$
$1500 \le W_3 < 3300$

$S_1 = T_1(y_{11} + y_{21} + y_{31} + y_{41})$
$S_2 = T_2(y_{12} + y_{22} + y_{32} + y_{42})$
$S_3 = T_3(y_{13} + y_{23} + y_{33} + y_{43})$

$y_{12} = y_{21} = 1$ if $L_1 = L_2$ and $W_1 = W_2$
$y_{13} = y_{31} = 1$ if $L_1 = L_3$ and $W_1 = W_3$
$y_{23} = y_{32} = 1$ if $L_2 = L_3$ and $W_2 = W_3$
$y_{11} = y_{22} = y_{33} = 1$
$y_{rs} = 0$ otherwise

$$x_{mn} = \begin{cases} 1 \text{ if element } n \text{ is a member of plate } m \\ 0 \text{ otherwise} \end{cases}$$

$$\sum_{m=1}^{3} x_{mn} = 1 \text{ for all } n = (1, 2, 3, 4, ..., 8, 9)$$

3.3 Solution Approaches

The mathematical formulation in section 3.2 shows a complex, mixed integer and non-linear model. It is clearly not a good candidate for conventional optimization tools such as linear or integer programming. This section develops approaches better suited to the characteristics of the CPG problem.

3.3.1 Multi-expert System

The first solution approach adopted in this dissertation is a new type of multi-expert system. Unlike ordinary rule-based expert systems which use rules in the simple format of "if X then Y" (see Section 2.3.3), the multi-expert system in this dissertation uses procedural rules to represent experts' knowledge. This approach provides a method to capture procedural knowledge that is not easy to grasp using traditional rule-based knowledge representation (Hung and Sumichrast 2000).

This multi-expert system approach relies on the expertise of design engineers who routinely develop cutting plans. It is believed that these cutting plan experts can usually develop very good solutions and further, they are generally aware of the steps involved in creating these solutions. However, these experts are hindered by the time consuming nature of the task, by inconsistent application of their own rules and by other errors. The multi-expert heuristic described below is based on formalizing and repeating process rules that such experts use for developing cutting plans.

3.3.1.1 Process Rules

Some general rules can be drawn from observation and interviews with the experts. The cutting plan experts all agree on how to begin forming steel plates. All use a general approach of separating the material requirements into sets with common steel type and thickness. All begin by forming as many groups as possible within one of these sets such that the plate width is within the lowest unit cost region. However, after most of the elements are assigned to these low-cost plates, the experts disagree on how to complete problems. Even a single expert might apply slightly different rules at different times. The best approach cannot be concluded since "the best expert" and "the best approach" are determined by the specific problem. Most of the steps could be easily explained as rules of thumb applied as part of a step-by step process. However, some decisions (including which approach to follow) are based on intuition.

Each "rule" described in the next section is a process which forms elements into plates or adds elements to existing plates. The rules are based on the expertise of several of the engineers. Not every engineer uses every rule and the order of application may vary. The XS adapted this concept by combining rules to form complete solutions and selecting among the solutions based on cost. The precise way the rules are combined into the XS is described later in this section.

One special case is not covered by the rules or heuristic described below. Occasionally, a project requires a very small number of elements of a particular steel type and thickness. If the total width of all required elements for some set is less than the heuristic’s lower bound, then they define a plate. There is no need to search for a better arrangement of the elements from such a set.

The heuristic combines needed elements into plates with the goal of minimizing material costs. Table 10 is an overview of each of the rules of thumb used by the engineers and embedded in the XS. Different types of material and different thicknesses cannot be mixed so these steps are applied separately to sets containing all requirements for elements of a particular steel type and thickness.

Rule	Name	Notes
1a	Stack to lower bound	Apply to same-length elements sorted by width
1b	Stack to lower bound	Apply alternating to widest and narrowest
2	Augment to upper bound	Always used and follow rule 1. Applies to elements of one length
3a	Stack unequal lengths	Apply from longest to shortest
3b	Stack unequal lengths	Apply from shortest to longest
4	Augment with unequal lengths	Optional completion rule
5	Exceed upper bound	Optional completion rule
6	Marginal cost	Optional completion rule

Table 10. Rules of thumb embedded in the heuristic solution.

Rule 1 - Stack to lower bound. Notice that two versions of this rule are in Table 10. First consider rule 1a. The process begins by sorting all elements of the current set with the same length by width. Next, elements of the same length are stacked (or combined into one plate) until the total plate width reaches the lower bound. More specifically, this procedure begins by assigning element 1 to a new plate. If this element is not wide enough to reach the lower bound, then element 2 is added to the plate. While the total width of the plate is less than the lower bound, remaining elements from the list are considered for addition to the plate. If the plate width becomes at least as great as the lower bound, then the plate is formed and a new plate is started using the first, unassigned element of the set. The procedure ends when there are no longer any unassigned elements to test for addition to the plate being formed.

The stack to lower bound rule is always used. However, it can be applied using the elements directly from the sorted list as described above (widest to narrowest) or it can be applied by alternating the selection of elements from the beginning and end of the list. Selecting elements alternating from the beginning and ending of the list is identified as rule 1b.

Rule 2 – Augment to Upper Bound. This rule augments plates already formed with unassigned elements of the same length. It is always used and follows rule 1. The narrowest plate is selected and the first unassigned element of the plate

length is considered. If adding this element does not exceed the upper bound, then it is added to the plate. If adding the element would exceed the upper bound, then each next unassigned element of the same length is tested until either an element is assigned to the plate or there are no more elements to test. The augment to upper bound rule continues selecting the narrowest plate and testing equal length, unassigned elements until no further assignments can be made.

Rule 3 - Stack unequal lengths. This rule is used to create new plates rather than to augment existing plates. The unassigned elements are ordered by length before an attempt is made to form a new plate. One-by-one, unassigned elements are added to the potential plate regardless of their lengths. If the total width becomes at least as great as the lower bound, then the plate is formed and the process is repeated. Like rule 1, two variations of this rule can be used. The elements can be ordered from longest to shortest (rule 3a) or from shortest to longest (rule 3b).

Rule 4 - Augment with unequal lengths. Select the narrowest existing plate. Add element 1 from the unassigned list if this does not violate the upper bound. If element 1 cannot be added, test each next element until either one is added or none can be. If an element is added, select the new narrowest plate and repeat the process until no plate can be augmented. Note that this rule ignores plate lengths.

Rule 5 - Exceed upper bound. Select the narrowest existing plate. Add the first element from the unassigned list that has the same length as the plate, even if the upper bound is violated. Repeat this process until no unassigned element has the same length as an existing plate. If unassigned elements remain, each is assigned to the plate with the smallest difference in length.

Rule 6 - Marginal Cost. This is the only completion rule that applies equally to augmenting existing plates or beginning new ones. The first unassigned element is selected. The cost of each existing plate, augmented by the element is determined. In addition, the expected cost of creating a new plate with the

unassigned element is determined. The option that causes the smallest increase in the cost of the set of plates is selected. This process is repeated for each unassigned element.

The six rules are combined in eighteen ways to form complete solutions. Figure 9 shows nine ways that the rules are combined if the first rule is 1a. The number in each node represents one of the rules described in the previous section. For example, rule 1a can be followed by rule 2, then 3a, then 4, and 5. If rule 1b is applied first, then a figure identical to Figure 9 (except for node 1a) could represent the remaining nine ways to form complete solutions. These eighteen rule combinations always begin with the steps common to all cutting plan experts. This assigns most elements to plates at a low cost.

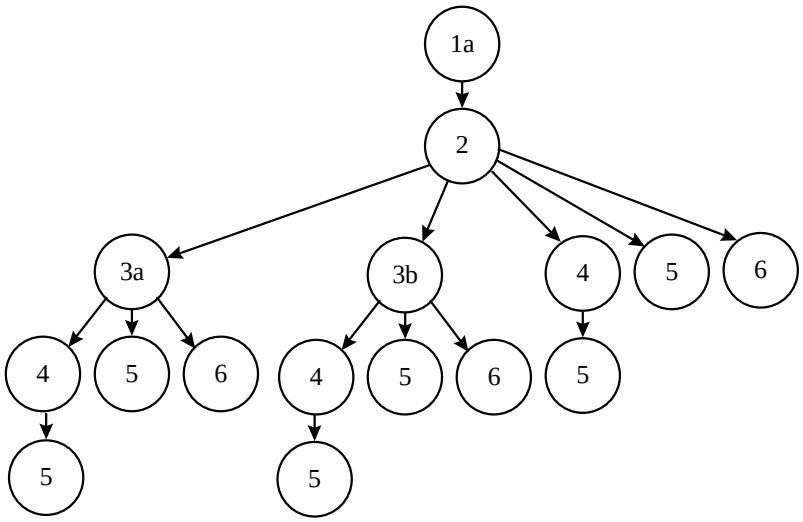

Figure 9. Each path from node 1a to a bottom node represents a set of rules applied to achieve a complete problem solution. Notice that if rule 1a is applied first, then there are nine possible ways to complete the solution.

The best way to combine the rules is not known in advance. Instead, all eighteen combinations of rules are embedded in the XS and the lowest cost solution selected. Figure 10 is a flowchart of the heuristic. Notice that in addition to selecting from among the eighteen rule combinations, various sets of bounds are tested. Recall that the unit price is defined over various width ranges for a given thickness and steel type. Initially, the lower bound and upper bound refer to the limits of width associated with the lowest cost per unit. Experimentation has

shown that better solutions may be produced when bounds other than those associated with the lowest unit price are tested.

The resulting cost from each of these completed solutions is compared and the lowest cost solution selected. The speed at which the calculations are completed makes it unnecessary to develop a heuristic for selecting among the completion procedures. While it was not possible to capture the occasional decision made through intuition, the experimentation shows that the overall quality of the XS solutions is good.

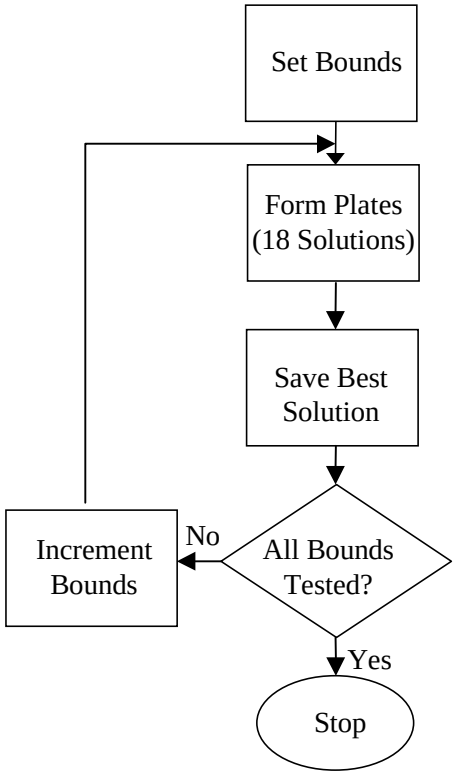

Figure 10. The heuristic requires the steps shown in the flow chart above.

3.3.1.2 Expert System Implementation

The heuristic is coded as an expert system using Visual Basic. Data are stored in a series of tables in Microsoft Access format. The following describes the system by presenting the dialog that a user would encounter in solving a material cutting plan problem.

After the title screen shown in Figure 11, the user is allowed to open an existing construction project or begin a new project using the controls shown in Figure 12.

If a new project is started, then a list of the individual elements required to complete the project must be entered. Figure 13 is an example of a screen used to edit the element specification. The software also makes provisions for editing other input parameters such as material costs. When possible, data created during previous projects are reused. Figure 14 is an example of some of the final output after material plates have been designed.

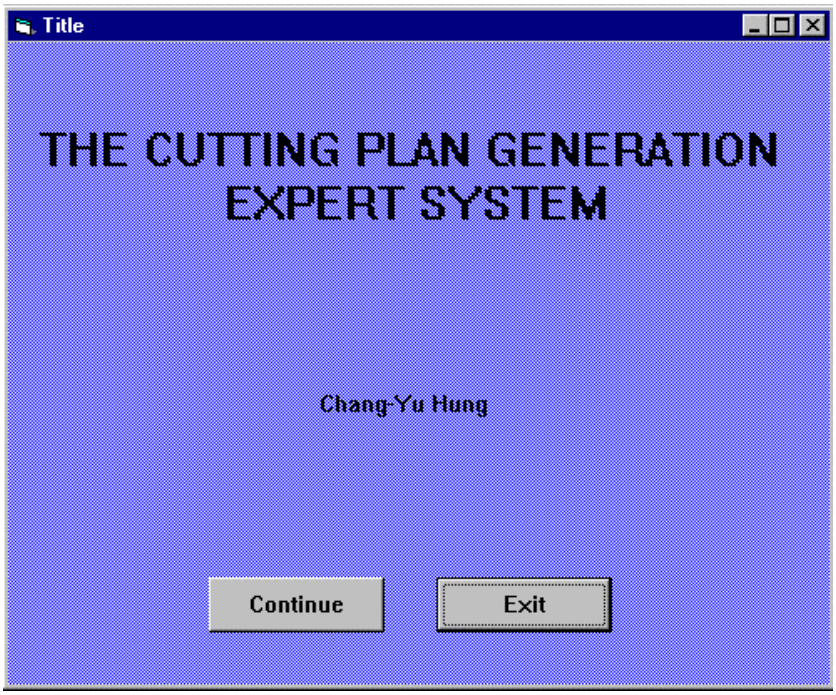

Figure 11. Title screen of the DSS in which the heuristic is embedded.

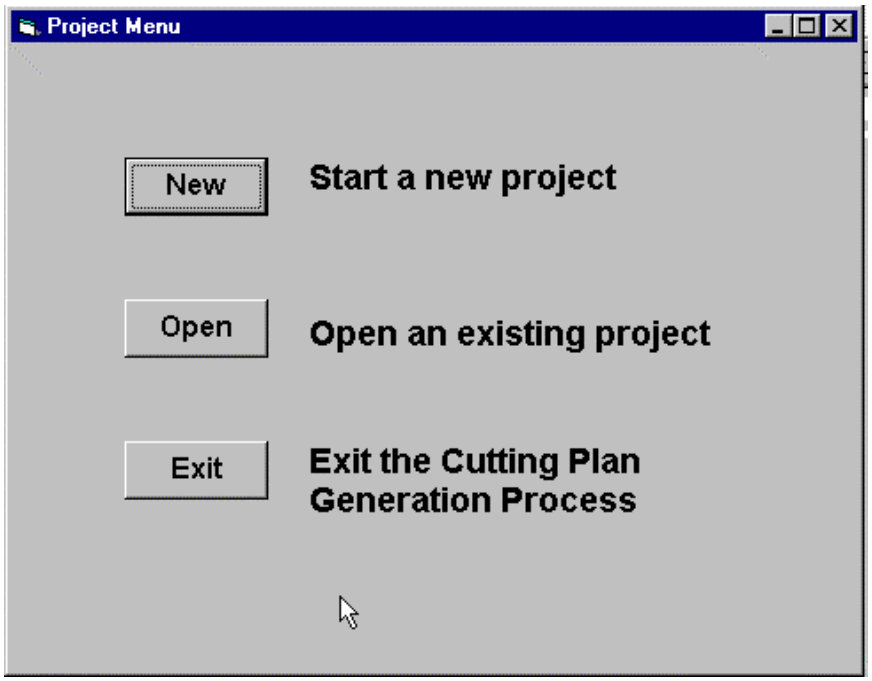

Figure 12. This screen allows a user of the DSS to select or begin a project.

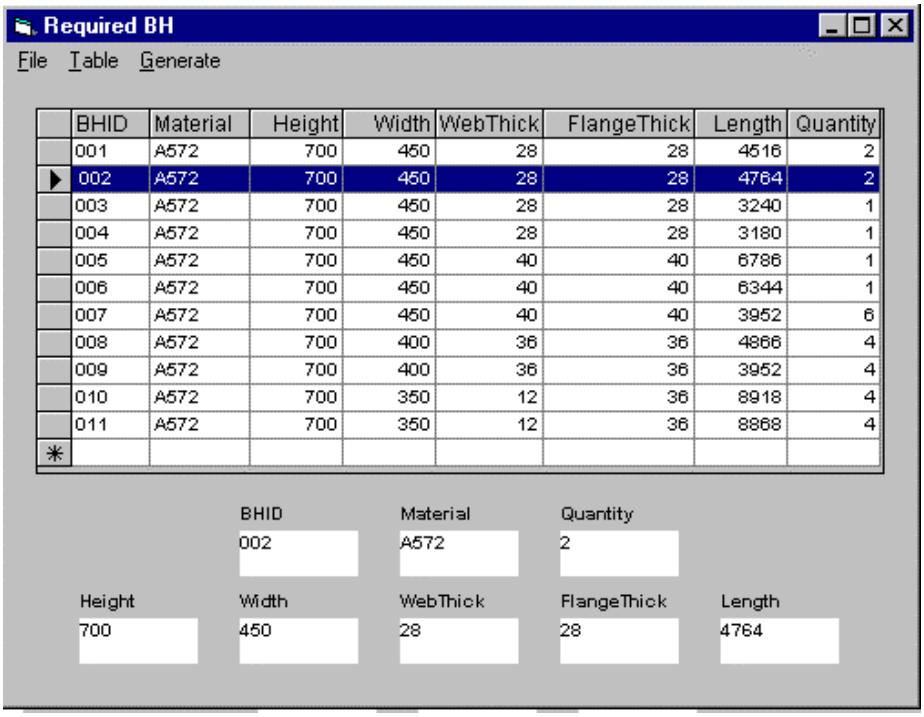

Figure 13. The table shown allows easy editing of the construction elements.

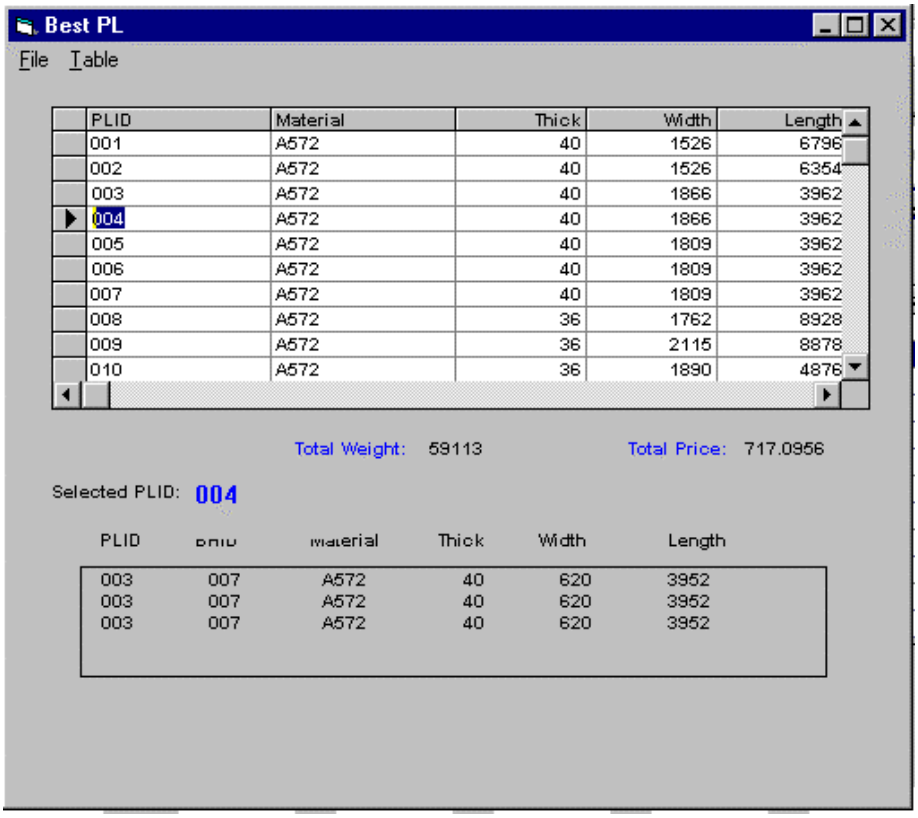

Figure 14. The solution in terms of what plates to order is shown in a display such as the one above.

3.3.1.3 Expert System Limitations

The initial test using a multi-expert system approach shows superior results compared to human experts. However, the author found it does not always produce optimal solutions. An example has been constructed to highlight the limitations of the XS.

Assume a project requires ten elements with the dimensions shown below. Assume the thickness is 40 *mm* and the material costs in Table 7 apply.

Element ID	1	2	3	4	5	6	7	8	9	10
Width (mm)	300	300	660	670	670	670	800	800	900	900
Length (mm)	9000	9000	9000	9000	9000	9000	9000	9000	9000	9000

Table 11. Required Elements in the Example Illustrating the Limitations of the Expert System.

One of the best arrangements is obtained by human intuition and is illustrated in Table 12. This solution includes four plates and all fall in the lowest unit cost range without cutting waste. This arrangement cannot be achieved using the XS since the heuristic always begins with rules (Rule 1a and 1b) that stack required elements either from widest to narrowest or alternating width.

Plate	1			2			3		4	
Plate Width (mm)	1776			1776			1563		1573	
Element ID	1	5	7	2	6	8	3	10	4	9
Width (mm)	300	670	800	300	670	800	660	900	670	900
Length (mm)	9000	9000	9000	9000	9000	9000	9000	9000	9000	9000

Table 12. Optimal arrangement that cannot be achieved using current rules in the expert system.

It is possible to add procedural rules to the expert system to improve its performance as cases in which it performs poorly are identified. However, this ad hoc approach would increase the complexity of the system. It would never be clear when new rules are needed. A more robust and systematic approach is desirable.

In the next section, the trend of adopting EA in solving grouping problems is reviewed first. Then the EA approach is explained in detail, followed by an example. The EA used in this dissertation is modified from Falkenauer's Grouping Genetic Algorithm (GGA) (1998). The reason for adopting this approach is also explained.

3.3.2 Evolutionary Algorithms

As explained in Chapter 2, many forms of evolutionary algorithm have been developed and applied to practical problems. One of the more recent applications for EAs is grouping problems. For example, Venugopal and Narendran (1992) use a GA to group machines into cells; Jiang et al. (1997) use a GA to cluster entities with similar characteristics; Billo et al. (1996) use a GA to solve a machine-component grouping problem; Hinterding and Khan (1995) and Lai and Chan (1997) solve cutting stock problems in one and more dimensions using an EA.

This dissertation adapts the grouping genetic algorithm (GGA) that has been proposed by Falkenauer (1996; 1998). There are several reasons for this choice:

1. The GGA has demonstrated superior results compared to the procedure developed by Martello and Toth (1990), which is regarded by many to be one of the best methods in solving bin packing problems.
2. The GGA provides a framework of constructing the EAs in solving grouping problems, such as the encoding, crossover and mutation. Other EA approaches may only provide a specific algorithm tailored to a specific problem.
3. Some EAs applied to grouping problems use the permutation representation. This approach requires a decoding scheme and can only be adopted if the plate sizes are fixed (e.g. cutting stock problems). The CPG problem, as stated before, does not rely on fixed size plates.

3.3.2.1 Grouping Genetic Algorithm

The EA developed for the CPG problem is adapted from Falkenauer's grouping genetic algorithm (GGA) (1991; 1994; 1996; 1998). GGA provides a "high level paradigm" or "meta-heuristic" (in Falkenauer's term) that must be tailored for different problems. However, all implementations include several key features.

This section explains these features including the representation scheme, crossover, mutation and inversion.

(1) Representation

The GGA uses a solution with two parts, a component string and a group string. For the CPG problem, a group is a plate.

$$\underbrace{g_1\ g_2\ g_3\ \ldots\ g_n}_{\textbf{component string}}\ :\ \underbrace{M_1\ M_2\ M_3\ \ldots\ M_m}_{\textbf{group string}}$$

The component string has one value for each construction element i. Each of these N values (g_i) represents the group into which the element has been assigned. For example, the solution corresponding to Table 12 has element 1 assigned to group 1 ($g_1 = 1$), element 2 assigned to group 2 ($g_2 = 2$), etc. The group string is simply a list of all group numbers recorded sequentially from g_1 to g_n. The complete representation for the solution of Table 12 is

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

The primary reason for using this scheme is to preserve the context-sensitive information presented in groups rather than within individual objects. The encoding enables the genetic operators to operate primarily on the group-bases instead of on individual elements.

(2) Crossover

The crossover operator in GGA distinguishes it from the other EAs found in the literature. Crossover is performed on the group string instead of the component string. Falkenauer (1998) argues that "the schemata defined on the genes of the simple chromosome (component string) do not convey useful information that could be exploited by the implicit sampling process carried out by the (standard) GA". Because of this "context insensitivity", the standard crossover (e.g. two point crossover) performed on the component string may not effectively produce good schemata.

The crossover follows the five-stage pattern in the GGA:

1. Select two crossing sites at random on each parent. Since the crossover operates on the group string, the selected sites refer to the group string, not the component string.
2. Insert the crossing section of the second parent into the first crossing site of the first parent.
3. Eliminate all duplicate elements in the first parent.
4. If necessary, elements in the affected groups (where duplicate elements are eliminated) are re-assigned according to a heuristic.
5. Repeat steps 2 to 4 on the two parents with reversed roles to generate the second progeny.

The procedure can be demonstrated with the following example with 7 components and 3 groups:

1. Suppose two parents are chosen for crossover (Notice that for clarity, unique group numbers are assigned across all parents.):

Parent 1:

Group Number	1	2	3
Components	1 3	2 4 5	6 7

Parent 2:

Group Number	4	5	6	7
Components	1 2	3	4 6	5 7

Randomly select two crossing sites for each parent using discrete uniform distribution. Suppose the crossing sites for parent 1 are 2 and 3; and the crossing sites for parent 2 are 6 and 6 (i.e. only one group in parent 2 is selected for crossover.)

2. Insert group 6 of parent 2 into parent 1.

Group Number	1	2	6	3
Components	1 3	2 4 5	4 6	6 7

3. Duplicated elements in parent 1 are eliminated. (Groups 2* and 3* denote changes from groups 2 and 3.)

Group Number	1	2*	6	3*
Components	13	25	46	7

4. If necessary, elements within the affected groups are re-assigned. In this example, suppose elements 2, 5, 7 are re-assigned as a new group. The resulting new solution is

Group Number	1	6	8
Components	13	46	257

5. Repeat steps 2 to 4 with two parents reversing roles.

(3) Mutation

Similarly, the mutation operator in GGA works with group (i.e. group string) rather than individual elements. In general, GGA provides no specific mechanism for the mutation operator; rather, different strategies have been suggested and provide guidelines on how mutation can be implemented. These strategies include

- create a new group
- eliminate an existing group
- shuffle a small number of randomly selected components (Falkenauer 1998).

For example, suppose group 6 in the above example is eliminated from parent 2. The result is

Group Number	4	5	7
Components	1 2	3	5 7

Elements 4 and 6 must be re-assigned. Suppose element 4 is re-assigned to group 4 and element 6 is re-assigned to group 5 according to some heuristic. The new solution generated from mutation is

Group Number	4*	5*	7
Components	1 2 4	3 6	5 7

(4) Inversion

Falkenauer also uses an operator called inversion in the GGA. Inversion consists of exchanging two genes on a chromosome. In fact, the idea of inversion had been proposed by Holland (1975) to alter the linkage of chromosomes. However, "neither early nor later work has managed to demonstrate a clear benefit from using inversion" (Radcliffe 1997, p. B2.5:5). Based on the above reason, the inversion operator is not used in the EA developed for the CPG problem.

The above highlights the major characteristics of GGA. The implementation details differ from application to application. An instance of the overall procedure for applying GGA to a bin packing problem is as follows (Falkenauer 1994):"

1. Generate at random an initial population of POPSIZE individuals.
2. Evaluate each individual in the current population according to the objective function to optimize. If the stop condition is attained, terminate.

3. Perform a noisy sort[7] of the individuals according to the values obtained in step 2 (tournament of size 2 is used).
4. Take the first (best[8]) N_C individuals, $N_C \leq$ POPSIZE/2, and cross them over, replacing the N_C last (worst) individuals with the progeny.
5. Mutate N_M individuals selected at random from the current population.
6. Apply inversion to N_I individuals selected at random from the current population. At this point, one generation of the GA has been completed, i.e., a new population has been created.
7. Go to step 2."

The initial parameters Falkenauer used were POPSIZE = 49 and 47, N_C = 12, N_M = 4, N_I = 4. In other cases (Falkenauer 1996), he uses POPSIZE = 100, N_C = 50, N_M = 33, N_I = 25.

3.3.2.2 Evolutionary Algorithm for CPG Problem

In this section, the EA solution approach for the CPG problem (CPG_{EA}) is explained. CPG_{EA} uses the GGA as a framework. Some details of GGA are changed to improve performance in the problem domain of interest.

(1) Representation

Like GGA, CPG_{EA} uses two chromosomes to represent a solution. Recall equations (9) and (10) in the mathematical model,

$$x_{mn} = \begin{cases} 1 \text{ if element } n \text{ is a member of plate } m \\ 0 \text{ otherwise} \end{cases}$$

$$\sum_{m=1}^{M} x_{mn} = 1 \text{ for all } n = (1, N)$$

[7] A noisy sort is simply taking two individuals from the list and putting one with higher fitness value at the beginning of the list and the other at the end of the list.

[8] Given the noisy sort used in step 3. the first N_C individuals have a high probability of being best ones.

		$X_{plate,\ element}$			
		Element			
Plate	1	x_{11}	x_{12}		
	2	x_{21}			
	..				
	M				x_{MN}

The matrix ***x*** represents the grouping solution. Since each column of ***x*** sums to 1, each gene in the component string can use its position to represent the element and the row number *k* where x_{mk} = 1 to indicate the group (i.e. steel plate) number of the element. For example, a component string "2112" represents elements 1 and 4 belong to group 2; elements 2 and 3 form group 1. Using ***x*** matrix representation, this is written

		Element			
		1	2	3	4
Group	1	0	1	1	0
	2	1	0	0	1

The second string of the chromosome is the group string - the collection of all group numbers. This information can be derived from the component string. In this example above the entire solution with component string : group string representation is

2 1 1 2 : 2 1

(2) Initialization

The initialization step populates the first generation. It is possible to form this population using a heuristic so that its solutions are guaranteed to be feasible and generally of good quality. The literature suggests that there is a net disadvantage to such an initialization approach because it tends to limit population diversity and increases initialization time. In this dissertation, a randomly generated population is used. A random initial population has the

additional advantage of providing a better test of the new EA operators in this dissertation.

The following steps outline CPG_{EA} initialization process:

1. Sort the elements according to length, width.
2. Number elements from 1 to N
3. Calculate the maximum number of possible plates, M. (See Equation (11).)
4. For each individual n, randomly assign a number, g_n, from the discrete uniform distribution with range from 1 to M. The value of g_n denotes the group number to which element n belongs. The collection of g_n, n = (1, N) forms the component string.
5. Repeat steps 2 to 4 until all individuals in the population are generated.

The initial solutions may not be feasible. The feasibility problem is addressed in the evaluation step by applying a penalty cost and will be discussed at more length in that context.

(3) Crossover

CPG_{EA} crossover operator is the same as in GGA. A heuristic is always used in CPG_{EA} to re-assign elements in the affected groups after eliminating duplicate elements. The GGA framework provided by Falkenauer does not specify the use of heuristics within the crossover operation. However, in solving a bin packing problem, the use of heuristics has a crucial impact on the performance (Falkenauer 1996).

(4) Mutation

Similar to crossover, mutation is performed on the group string. In CPG_{EA}, mutation eliminates an existing group and re-arranges its components using a special purpose heuristic.

(5) Evaluation

The objective function in the mathematical model (see Section 3.2) is a natural choice for evaluation. However, the grouping arrangement may not be feasible. The problem of infeasibility is addressed by imposing a high cost to the plates that are too narrow or too wide. If a plate is too narrow, the minimum width is imposed to the plate and the price is calculated accordingly since a plate with the minimum width can be ordered to satisfy the demand. The extra cost, in this case, is the cutting waste. On the other hand, if a plate is too wide, the highest unit cost is imposed to the plate.

(6) Selection

The GGA framework does not specify the choice of selection schemes. In solving a bin packing problem, Falkenauer (1996) uses the tournament selection with overlapped population between generations. Falkenauer (1998) describes that the reason for this preference over more "orthodox" generational EA models is to reduce memory requirements of the algorithm. In this dissertation, CPG_{EA} uses generational EA with elitist strategy as the selection scheme for the next generation. Without further analysis, it is felt that the selection pressure in the GGA using overlapped populations is greater than that in CPG_{EA} since approximately half of the best parents will survive in the next generation in the GGA. As indicated in the literature review, high selection pressure must be compensated for by other operators that diversify the population to avoid premature convergence. In this dissertation, the approach with lower selection pressure is used in the interest of finding the optimal solution within a reasonable time. The selection scheme of CPG_{EA} is the major difference between CPG_{EA} and Falkenauer's GGA in solving the bin packing problem. Table 13 summarizes the differences between the GGA and CPG_{EA}. The comparison of effects using different selection schemes is beyond the scope of this dissertation.

	Falkenauer's GGA for a bin packing problem	CPG_{EA}
Selection Scheme	Overlapping Population	Non-overlapping population with elitist strategy
Selection Pressure	Higher	Lower
Selection Methods	Tournament	Linear Ranked
Embedded Heuristics	First fit heuristic; Local search modified from Martello and Toth's heuristic	Specific designed for the CPG problem
Inversion	Yes	No inversion operation

Table 13. Differences between GGA and CPG_{EA}.

On the selection for crossover and mutation, the process ranks solution values by objective function value. The selection probability is a linear function of solution rank. This selection scheme is designed to avoid the premature convergence effect often observed in populations containing a "super-solution". CPG_{EA} adopts the elitist strategy to assure that the best solution is preserved in the next generation. The selection process includes the following steps:

1. Rank individuals according to the evaluation procedure
2. The best individual within a generation is automatically preserved (or copied) into the next generation. Other individuals in the parent generation will not survive in the next generation; this is the elitist strategy used in generational EA. (See Section 2.4.3.)
3. Calculate the probability of selection for each individual according to its rank using linear ranking (Grefenstette 1997):

$Rank_{Worst}$ = 0; $Rank_{Best}$ = Population Size - 1

$$Probability = \frac{\alpha + \frac{Rank}{Population\ Size - 1} \times (\beta - \alpha)}{Population\ Size}$$

where β is the expected number of offspring to be allocated to the best individual during each generation and $\alpha = 2 - \beta$, $1 \leq \beta \leq 2$. Usually, $\beta = 1.5$.

In CPG_{EA}, the selection process builds a list of all parents for use later in the crossover and mutation operations. In forming this list, the original population is repeatedly sampled so the same solution may appear more than once as a candidate for crossover and mutation.

(7) Termination

CPG_{EA} is terminated while it satisfies either one of the following conditions:

1. a total of *MaxGen* generations is processed.
2. there is no improvement in the best objective function value in *MaxSameGen* generations.

The overall procedure of the EA is depicted in Figure 15.

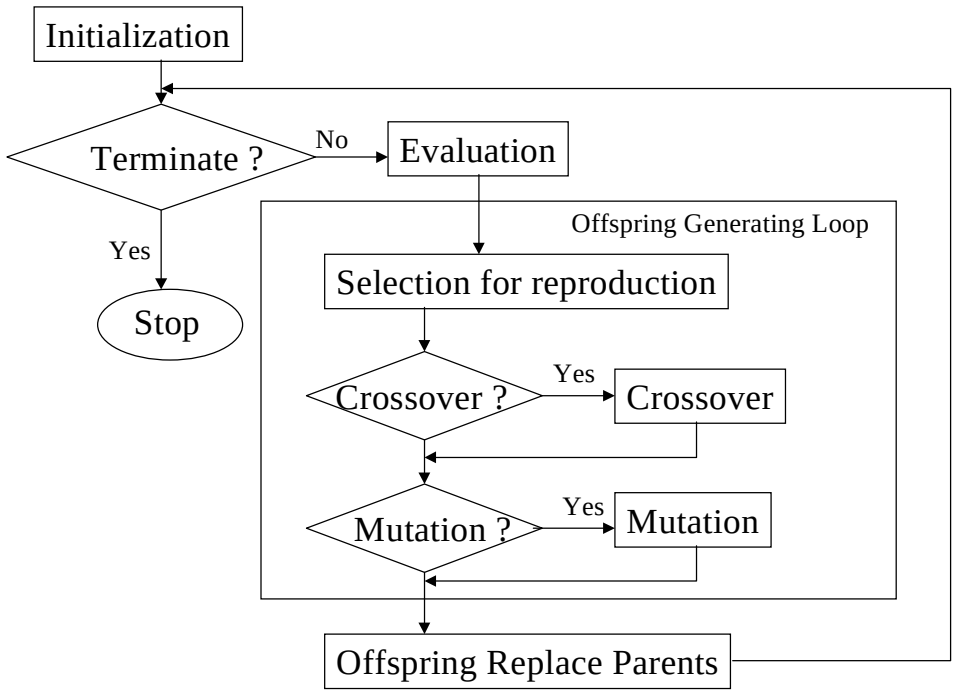

Figure 15. Flowchart of the EA

The overall procedure is similar to a "traditional" generational GA except the representation, crossover and mutation follow GGA's scheme. A probability of crossover is chosen. Parents chosen to reproduce offspring will actually go

through crossover with this probability of crossover; so the number is also called the crossover rate. Similarly, a mutation rate has to be chosen for the mutation operator. The chosen parents that do not go through crossover are carried over to the next generation, i.e. they produce offspring identical to themselves.

No rate for crossover or mutation is universally accepted as the most appropriate but successful implementations by prior researchers provide some guidance. In Falkenauer's GGA implementations, 12% - 50% of a generation is replaced by crossover; mutation affects 8% to 33% of a generation. However, as explained above, the selection scheme Falkenauer used is overlapping population. The crossover rate used in generational EA is usually much higher (e.g. 95%) and the mutation rate is much lower (e.g. 1%). The effects of the selecting of these parameter values will be investigated in this dissertation.

3.3.2.1 Heuristic within CPG_{EA}

While implementing the EA, crossover is likely to disrupt many groups if the grouping arrangements are very different between both parents since the affected groups are discarded and elements in those groups are reassigned. This makes the heuristic for reassigning the elements critical in searching for good solutions. A good heuristic can produce good grouping arrangements from elements that are disrupted by crossover or mutation. A good heuristic accelerates the building of good schemata and thus provides additional optimization pressure to the EA beyond the power of genetic operators. One study concluded that the optimization power of GGA comes primarily from the heuristic used within it (Zulawinski 1995).

It is desired to study the effects of different heuristics within the EA, and to investigate if the disruption is affected by the stage of the evolution. It is possible that, toward the end of the EA, the affected groups and the disrupted elements will be reduced. In this dissertation, different heuristics are investigated to determine their effectiveness.

3.3.2.2 Example of CPG_{EA}

To illustrate CPG_{EA}, a simple example is provided. This example has the same cost structure and required elements as the example illustrated in section 3.1. Refer to Table 7 for unit prices and to Table 8 for required elements.

(1) Initialization

1) Sort the elements according to length and width. Number elements from 1 to N (N = 9, in this example).

2) Calculate the maximum number of possible plates. In this example, the number is 3.

$$M = \begin{cases} \left\lfloor \frac{\left(\sum_{n=1}^{N} w_n\right) + 3N}{A_{\min}} \right\rfloor & \text{if } \left(\sum_{n=1}^{N} w_n - A_{\min} \left\lfloor \frac{\left(\sum_{n=1}^{N} w_n\right) + 3N}{A_{\min}} \right\rfloor \right) \le A_2 - A_1 \\ \left\lceil \frac{\left(\sum_{n=1}^{N} w_n\right) + 3N}{A_{\min}} \right\rceil & \text{otherwise} \end{cases}$$

$$= \begin{cases} \left\lfloor \frac{4650 + 3\times 9}{1524} \right\rfloor & \text{if } \left(4650 - 1524 \left\lfloor \frac{4650 + 3\times 9}{1524} \right\rfloor \right) \le 2000 - 1524 \\ \left\lceil \frac{4650 + 3\times 9}{1524} \right\rceil & \text{otherwise} \end{cases}$$

3) Select 3 as the population size.

4) For each component string element, assign a random number generated according to discrete uniform distribution U(1-3). Repeat until all three component strings in the population are generated. Derive the associated group strings.

Suppose the initial population consists the following:

Population	Component	Group
1	233112222	123
2	122111122	12
3	333211122	123

(2) Evaluation:

Plates are feasible if their width is between the minimum and maximum allowed plate sizes. The material cost of feasible plates is used directly as a fitness value. For plates that are too narrow, the cost (and thus the fitness value) is calculated after increasing the width to the minimum allowed. The wider plates can be ordered and used although material will be wasted. If the width is beyond the feasible range, the highest unit cost is imposed as the penalty.

Consider group 1 of the first solution in the population:

Element	4	5
Width	500	500
Length	8000	8000

The plate’s width = 500+500+3 = 1003

Since the minimum required plate width is 1524 mm, the plate's width is increased to this value.

Length = Max{8000, 8000} = 8000 mm

Thickness = 40 mm

Weight = $40 \times 1524 \times 8000 \times 0.00000785 = 3828.288$ Kg

Cost = $3828.288 \times 11.95 = 45748$ NT$

The second group (or plate) of the first solution includes elements 1, 6, 7, 8 and 9.

Width = 400 + 700 + 500 + 500 + 700 + 4×3 = 2812 mm

Length = Max {8000, 8000, .., 10000} = 10000 mm

Thickness = 40 mm

Weight = $40 \times 2812 \times 10000 \times 0.00000785$ = 8829.68 Kg

Cost = 8829.68×12.10 = 106839 NT$

Group 3 includes elements 2 and 3.

Width = 400 + 400 + 3 = 803 mm. The width must be increased to 1524 mm, the minimum allowed.

Lenght = Max {8000, 8000} = 8000 mm

Weight = $40 \times 8000 \times 1524 \times 0.00000785$ = 3828.288 Kg

Cost = 3828.288×11.95 = 45748 NT$

Total cost of solution 1: 45748 + 106839 + 45748 = 198335 NT$

The fitness values of other solutions are also calculated using this procedure. The probability of being selected as a parent is calculated by the linear ranking with α = 0.5 and β = 1.5. For solution 1, the probability is

$$Probability = \frac{0.5 + \frac{0}{3-1} \times (1.5 - 0.5)}{3} = 0.1667$$

The result is listed below. Notice that the best solution is assigned the greatest rank.

Solution Index	Solution		Fitness	Rank	Probability
	Component	Group			
1	233112222	231	198335	0	16.67 %
2	122111122	12	176413	1	33.33 %
3	333211122	321	173776	2	50.00 %

(3) Crossover

1) Suppose solutions 1 and 3 are selected as parents for crossover

Parent #1: 233112222 : 231

While the above notation is efficient for coding a computer implementation of CPG_{EA}, it is inconvenient for humans to use. New notation is introduced in this section to help clarify CPG_{EA} operations. The new notation specifies a solution by listing each group number and element set. For example, in parent #1, group 2 contains elements 1, 6, 7, 8 and 9; group 3 contains elements 2 and 3; and group 1 contains elements 4 and 5. Using the new notation, this is written as:

Group(Elements): 2(1, 6, 7, 8, 9) 3(2, 3) 1(4, 5).

Similarly, Parent #2: 333211122 : 321 is written as

3(1, 2, 3) 2(4, 8, 9) 1(5, 6, 7)

2) Performed at each iteration, the probability of crossover is determined by the crossover rate. Assume crossover will be performed on the selected parents.

3) Select two crossing sites at random on each parent. Suppose the two crossing sites for parent 1 are 2 and 2 (only the second group is selected for crossing); for parent 2 are 1 and 2 (the first and the second groups are selected for crossing).

4) Insert the crossing section of the second parent into the first crossing site of the first parent. In this example, insert groups 3 and 2 of parent #2 into parent #1; after renumbering the groups from parent #2, Parent #1 becomes

   2(1, 6, 7, 8, 9) 4(1, 2, 3) 5(4, 8, 9) 3(2, 3) 1(4, 5)

5) After this operation, some elements are present in more than one group. Element 1 is present in groups 2 and 4. Similarly, elements 2, 3, 4, 8, 9 are also in multiple groups. The operation continues by eliminating all duplicate elements in the first parent from groups originally in this parent. The inserted groups are never changed.

   2(6, 7) 4(1, 2, 3) 5(4, 8, 9) 1(5)

6) Elements in groups changed by the above step are freed from any group as illustrated below.

   4(1, 2, 3) 5(4, 8, 9) 5, 6, 7

   These free elements must now be assigned to a group through a heuristic. CPG_{EA} use one of the heuristics from the XS for this assignment. In this example, rules 1a, 2, 3a, 4 and 5 are chosen (i.e. the leftmost path in Figure 9). The result is

   4(1, 2, 3, 5) 5(4, 8, 9) 6(6, 7)

7) Re-index (rename) groups. (This is for computational convenience and allows values to be stored within a restricted range.)

   1(1, 2, 3, 5) 2(4, 8, 9) 3(6, 7)

   Using the original chromosome notation, this solution is 111213322 : 123

8) The same procedure is applied to Parent 2:

   4(2, 3) 3(1, 2, 3) 2(4, 8, 9) 1(5, 6, 7)

   4(2, 3) 2(4, 8, 9) 1(5, 6, 7) 1

   4(1, 2, 3) 2(4, 8, 9) 1(5, 6, 7)

   After renumbering, the solution becomes

   3(1, 2, 3) 2(4, 8, 9) 1(5, 6, 7)

Using the original notation, this solution is

333211122 : 321

(4) Mutation

1) In CPG_{EA}, mutation is carried out on the offspring generated after crossover. Unlike Falkenauer's implementation, this is similar to generational EA. Similar to the crossover, a mutation rate determines if mutation is actually performed. Suppose offspring #2 is selected for mutation:

   333211122 : 321

2) Select a group at random. Assume group 3 is chosen.

   Eliminate the group, the solution becomes

   2(4, 8, 9) 1(5, 6, 7) 1, 2, 3

3) Reinsert the missing elements according to a heuristic. In CPG_{EA}, the missing elements 1, 2, 3 are assigned according to one set of the rules in the XS.

   2(1, 3, 4, 8, 9) 1(2, 5, 6, 7)

   Using the original notation, the new offspring is

   212211122 : 21

(5) The next population

Solution Index	Solution	
	Component	Group
1 (best from last generation)	333211122	321
2	111213322	123
3	212211122	21

(6) Termination

Perform steps 2 through 5 until either the maximum number of generations has been formed or no improvement in the best solution is seen for a specified number of generations.

CHAPTER 4 RESULTS AND ANALYSIS

A series of experiments was designed to investigate the effectiveness and efficiency of the expert system and CPG_{EA}. In this chapter, different aspects of the experiments including the data sets used, the performance measurements, the results, and the analysis are explained in detail.

4.1 Data Sets

Experimentation is performed using two classes of data – historical data and simulated data. The historical data are provided by Lien Kang Heavy Industrial Co. (LK). The data consist of seven different steel construction projects that the company encountered. Each project includes components with different dimensions. The data are summarized in Table 14. This table summarizes the projects by grouping similar sized elements in the same row rather than listing each element in a separate row. This helps simplify the table. Manual solutions generated by human experts and implemented by the company are available. So the historical data provide a base line for comparison with the proposed solution technique.

	Element Dimensions in mm			
Historical Case	**Thickness**	**Width**	**Length**	**Quantity**
H1	40	450 - 620	3952 - 6786	24
	36	350 - 628	3952 - 8918	40
	28	450 - 644	3180 - 4764	18
	12	628	8868 - 8918	8
H2	50	400 - 984	9375 - 14040	58
	40	400	8040 - 15540	38
	35	500 - 1870	9540 - 14040	30
	32	599	9090 - 13090	4
	30	400 - 1100	9340 - 15540	53
	28	1470	9090 - 13090	4
	20	890 - 1430	8040 - 15540	43
	15	940	9340	1
H3	32	400	12040	6
	28	350	10080 - 11660	18
	25	350	9480 - 13320	94
	22	300	10610 - 13320	16
	19	661	12040	3
	16	381 - 675	9480 - 13320	64
H4	40	600 - 650	11610 - 16910	16
	35	600 - 650	9010 - 15185	58
	22	1520 - 1720	11610 - 16910	8
	20	1330 - 1530	9010 - 15185	29
H5	36	400	9040	4
	32	350	11660	6
	28	350	9780	8
	25	350	9780 - 13320	58
	22	300	12040	6
	19	450 - 653	9040 - 13320	19
	16	661 - 681	9780 - 13320	22
H6	40	400	9950	8
	36	400	6210	6
	32	350	11370	4
	28	350 - 400	9040 - 13420	30
	25	350	9940 - 12980	40
	19	645 - 653	6210 - 9950	7
	16	600 - 675	9040 - 13420	37
H7	50	400 - 984	9375 - 14040	58
	40	400	8040 - 15540	38
	35	500 - 1870	9540 - 14040	30
	32	599	9090 - 13090	4
	30	400 - 1100	9340 - 15540	53
	28	1470	9090 - 13090	4
	20	890 - 1120	8040 - 15540	43
	15	940	9340	1

Table 14. Summary of dimensional ranges of historical cases.

The amount of historical data is limited and does not necessarily represent all cases important to test. Therefore, simulated test projects have been created. The range of sizes, weights and other characteristics are selected to conform with the purpose of each experiment. The range may be consistent with problems that might be encountered in the industry or restricted for the purpose of testing the effect of the problem characteristics. The following process is used to create problems so that the optimal solution is known.

Note that

PL_1	=	Lower bound of plate length
PL_2	=	Upper bound of plate length
H	=	The weight threshold to avoid penalty unit cost
EW_1	=	Lower bound of element width
EW_2	=	Upper bound of element width
Min_Comp	=	Minimum number of elements generated

1. Randomly select the plate width from a uniform distribution within the range of minimum unit cost and select a length from a uniform distribution between PL_1 and PL_2.

2. Calculate the weight of the plate generated in step 1. If the weight is under the threshold, H, to avoid the penalty cost, another plate with identical dimensions is added to the project order. This step is repeated until the total weight of all identical plates is greater than H.

3. Divide the generated plate(s) by following two sub-steps:

   A Select a number from a uniform distribution between EW_1 and EW_2 as the element width. This element width is subtracted from the plate width.

   B Repeat step 3A to generate other elements. If the generated number is greater that the remaining width or the remaining width after subtracting the generated component is less than EW_1, the remaining width is used as the element width to enforce the correct plate being formed. To illustrate in mathematical form, the element width is:

$$w = \begin{cases} RW & \text{if } Rnd > RW \text{ or } RW - Rnd < EW_1 \\ Rnd & \text{otherwise} \end{cases}$$

where

w = element width

RW = remaining width

Rnd = random number generated

4. Repeat steps 1 to 3 until the number of elements generated is over Min_Comp.

A problem with a range similar to a typical medium-size problem in industry has the following parameter values:

PL_1 = 3000 mm

PL_2 = 9000 mm

H = 2600 kg

EW_1 = 100 mm

EW_2 = 900 mm

Min_Comp = 100

In this chapter, all simulated cases are generated using these values except changes in the plate length and thickness in experiments in Sections 4.4.1, 4.4.3, and 4.5.1 that explore the effect of problem characteristics.

4.2 Performance Measurement

The cutting plan must satisfy demand and has a major impact on raw material cost. Minimizing cost is the objective in both XS and CPG_{EA}. The specific material requirements and cost details of a project determine its optimal cutting plan and minimum cost. Therefore, project cost can be a misleading performance measure if it is not compared among solutions to a specific problem. A relative index derived from the generated cost and the optimal solution serves as a better performance measurement. In this dissertation, the performance measurement of both solution approaches is measured by the percentage of the generated solution over the optimal solution as in Equation (12).

$$\Delta = \frac{C_{gen} - C_{opt}}{C_{opt}} 100\% \quad (12)$$

Here Δ is the performance measurement,
C_{gen} is the solution value generated by either XS or CPG_{EA}, and
C_{opt} is the optimal solution.

In the cases where optimal solutions are not available, such as historical cases, a lower bound is used in place of the optimal solution. The bound uses the fact that each set of steel element requirements, defined by the steel type and thickness, can be treated separately. It considers the unavoidable cutting loss associated with the process of cutting the plates into construction elements. For most sets, the lower bound assumes that an arrangement can be found such that the lowest unit cost can be used without any additional material waste. In practice, such an arrangement may not exist and thus the bound may not represent the cost of the best feasible solution. A few of these sets are small. If all widths in a set can be added and the total width (plus the technological waste factors) is less than the lower bound for the lowest unit cost material, either a higher price per unit must be paid or additional material must be added to the plate to achieve the lowest unit cost. The cost of each of these two options is computed and the smaller is used as the cost of the set.

The processing time to generate a solution is also measured. Since we are most interested in the solution cost, the processing time is a secondary consideration. However, the solution time must not be greater than is practical for solutions to this type of problem in industry.

4.3 XS Tests

The multi-expert, expert system described in Section 3.3.1 has been created as a computer program. A series of experiments was designed to investigate the effectiveness and efficiency of the XS, and to investigate how different problem characteristics affect its performance. The purpose of the XS tests is to

determine if procedural knowledge from multiple experts can be adequately captured and applied through such a system.

4.3.1 Design of Experiments

The first experiment uses the seven historical projects as the testing subjects. The results of the comparison of manual solution, XS solution, and the lower bound for each of the seven historical projects will demonstrate how well the XS performs compared to human experts and will provide some information about the absolute quality of the solutions.

The XS will also be used to solve 40 simulated projects in a 2^2 factorial experiment. The projects were generated using the procedure previously described varying the plate cost structure and thickness as shown in Table 15. These two factors are chosen to investigate the influence of the cost structure and the characteristic of the required elements on the effectiveness of the XS. Recall that the cost structure labeled monotonic increase is one where the unit cost of material increases as the plate width increases for the entire range of plate width available from the vendor. The cost structure labeled convex is one where the unit cost is high for plates that are either narrow or wide. For either cost function, a penalty cost is added if all plates of a given set of dimensions weigh less than a threshold. In the test, thick plates have 40 mm thickness while thin plates have 16 mm thickness.

	Number of simulated problems	
Cost function of plate width	Thick plates	Thin plates
Monotonic	10	10
Convex	10	10

Table 15. Number of projects for each combination of factors

4.3.2 Results and Analysis of XS Tests

The results of XS tests on the historical projects are shown in Table 16. The expert system solution is less expensive than the manual solution implemented by LK in all of the historical projects except H6. The savings for one project ranges from NT$-3,000 to NT$39,000 with an average saving of NT$8,143. This test indicates that the expert system can solve the cutting plan problem faced by LK with less costly solutions than are generated manually by its engineers. However, the solution quality is only slightly better than that produced by LK's engineers and it is not guaranteed to be better than a manual solution for all projects that might be acquired by LK.

Comparison of LK's solutions and the expert system solutions to the lower bound confirm that these techniques produce high quality solutions. See columns Δ_{LK} and Δ_{XS} in Table 16. The XS solution is 2.28% higher than the lower bound for project H1. In all other cases, this gap between the feasible solutions of the XS and the potentially infeasible lower bound solution is less than 2%.

Project	Lower Bound	Expert System			Manual		Difference between XS and Manual
		Cost in thousands of NT$	Δ_{XS}	Time (sec.)	Cost in thousands of NT$	Δ_{LK}	
H1	701	717	2.28%	67	725	3.42%	8
H2	5676	5735	1.04%	200	5774	1.73%	39
H3	1979	1988	0.45%	60	1990	0.56%	2
H4	3205	3248	1.34%	94	3254	1.53%	6
H5	1213	1224	0.91%	42	1227	1.15%	3
H6	1424	1448	1.69%	49	1445	1.47%	-3
H7	5674	5733	1.04%	91	5735	1.08%	2
Average			1.25%	86		1.56%	

Table 16. Comparison of material costs for historical projects

The XS is also very fast. It takes only 200 seconds to generate the solution for project H2 on a PC with Celeron 500MHz CPU and 64M RAM. In the other cases, the calculation takes less than two minutes. This short processing time makes the XS potentially useful in more situations than simply material ordering.

For example, it can be used for calculating many different what-if scenarios in the bidding of projects and for exploring design alternatives.

On the performance of different combinations of the procedure rules, close examination reveals that many of the heuristics produce similar results. For example, the first nine heuristics generate the same final cost for the 40 mm elements in historical case 1 (see Table 17). Other heuristics also generate similar results with final costs of NT$214,000 and NT$216,000. It is unclear whether one of the heuristic is superior to the others in all circumstances.

Heuristic	Combination of Rules	Cost in 1,000NT$
1	1a - 2 - 3a - 4 - 5	212
2	1a - 2 - 3a - 5	212
3	1a - 2 - 3a - 6	212
4	1a - 2 - 3b - 4 - 5	212
5	1a - 2 - 3b - 5	212
6	1a - 2 - 3b - 6	212
7	1a - 2 - 4 - 5	212
8	1a - 2 - 5	212
9	1a - 2 - 6	212
10	1b - 2 - 3a - 4 - 5	216
11	1b - 2 - 3a - 5	216
12	1b - 2 - 3a - 6	214
13	1b - 2 - 3b - 4 - 5	216
14	1b - 2 - 3b - 5	216
15	1b - 2 - 3b - 6	214
16	1b - 2 - 4 - 5	216
17	1b - 2 - 5	216
18	1b - 2 - 6	214

Table 17. Results of 40 mm thickness element in historical case 1

In the course of testing the XS, it was observed that heuristics which do not include Rule 3 can generate infeasible solutions. For example, given the lower bound 1524 mm and upper bound 1575 mm, heuristic 7 (rules 1a - 2 - 4 -5) produces an infeasible solution on the 28 mm elements in historical case 1. The arrangement produces plates with widths that exceed the upper limit available from the vendor. Recall that only procedure rules 1, 2 and 3 can produce new

plates; also, rules 1 and 2 can only produce a limited number of plates, depending upon the makeup of required elements, since they only build new plates from elements with the same length. Without rule 3, it is possible to generate only a limited variety of plates, and the completion rules 4, 5 and 6 will add all the remaining elements upon these plates, potentially resulting in plates with widths exceeding the upper limit.

The results of developing material plans for the simulated projects are displayed in Table 18 and Table 19. Table 18 shows that the average cost of the XS solution for problems based on thick material is less than 2% greater than the optimal solutions. However, Table 19 shows that the average cost of XS solutions for problems using thin material is more than 13% higher than the cost of using the optimal solutions. Careful examination of the detailed solutions shows that this decrease in performance as compared to the optimal solutions is caused when the XS solutions incur a penalty cost due to arranging the requirements into individual plates that are lighter than the weight threshold and not duplicating those plates. Recall that the penalty unit cost only occurs when plates with a specific dimension weigh less than a threshold. A single plate with 40 mm thickness can easily weigh more than the threshold so the penalty cost is unlikely to be applied no matter how the elements are arranged. On the other hand, the elements in the thin cases have to be grouped in a way that all the elements with the same length form plates with identical dimension. In addition, the plates must have a width within the lowest unit cost range to achieve the minimum cost. This arrangement may be unique and this is much more difficult to generate. The average percentage increase (about 12%) in the price reflects the penalty cost applied in the thin material cases.

Analysis of variance on the effect of the cost function type and the plate thickness on the difference in cost between the optimal and XS solutions indicates that only the plate thickness factor is statistically significant at the 5% level. (See Appendix A.1.) The cost structure does not have a significant impact

on the performance of the XS. This finding suggests that the thickness of the required elements is crucial in determining the performance of the XS, and the CPG problem is complicated by the penalty cost constraint. Based on this finding, further investigation of the effect of component thickness is performed in the testing of CPG_{EA}, but the effect of different cost functions is assumed to be insignificant and is not tested further.

	Unit Cost Structure							
	Monotonic				Convex			
Thickness	Optimal Cost in thousands of NT$	XS Cost in thousands of NT$	Δ	Time in Seconds	Optimal Cost in thousands of NT$	XS Cost in thousands of NT$	Δ	Time in Seconds
40 mm	844	864	2.35%	90	859	871	1.47%	91
	859	863	0.50%	85	954	962	0.86%	95
	909	928	2.02%	84	929	938	0.95%	87
	872	897	2.79%	93	976	985	0.89%	91
	763	783	2.60%	94	1039	1040	0.07%	89
	976	993	1.68%	90	1122	1130	0.73%	93
	864	877	1.44%	85	1011	1019	0.79%	89
	965	977	1.30%	91	1018	1025	0.73%	90
	1087	1098	1.04%	95	1010	1019	0.87%	94
	1182	1187	0.40%	93	936	950	1.68%	93
		Average	**1.61%**	**90**		**Average.**	**0.90%**	**91**

Table 18. Comparison of material costs for simulated projects (40 mm)

	Unit Cost Structure							
	Monotonic				Convex			
Thickness	Optimal Cost in thousands of NT$	XS Cost in thousands of NT$	Δ	Time in Seconds	Optimal Cost in thousands of NT$	XS Cost in thousands of NT$	Δ	Time in Seconds
16mm	384	425	10.83%	88	398	453	13.82%	89
	394	457	15.97%	95	386	436	12.93%	87
	385	435	13.14%	87	364	420	15.31%	89
	333	391	17.47%	98	401	448	11.79%	91
	425	470	11.95%	89	371	421	13.46%	93
	479	537	12.08%	100	418	466	11.74%	94
	478	528	10.80%	93	388	443	14.28%	94
	481	544	12.95%	99	438	501	14.33%	99
	370	429	16.04%	94	417	476	14.36%	94
	399	445	11.70%	87	460	503	9.41%	94
		Average.	**13.29%**	**93**		**Average.**	**13.14%**	**92**

Table 19. Comparison of material costs for simulated projects (16 mm)

4.4 CPG_{EA} Tests

The second solution approach proposed in this dissertation is the evolutionary algorithm called CPG_{EA}. The implementation of CPG_{EA} involves the selection of more parameters than the XS. In addition, the heuristics embedded in the algorithm must be designed. Since there are many factors involved, it is impractical to design an experiment that investigates all of them simultaneously. Instead, parameters believed to be most important are investigated through a sequential series of experiments. These experiments cannot determine a global optimal set of parameter choices; however, they can suggest which factors are key in achieving good performance and suggest good values. Subsequently, a more systematic approach including fewer variables is conducted based on this exploratory work. Its results are reported in Section 4.5. As explained in the previous section, the characteristic of the problem may also have a great impact on the performance of the solution approach. This uncontrollable factor (problem characteristics) is also investigated with an experiment.

4.4.1 Exploratory CPG_{EA} Tests

While implementing CPG_{EA}, there are several parameters to specify such as population size, crossover rate and mutation rate. An initial pre-test of CPG_{EA} is performed using some of the historical projects and several simulated cases before a more extensive series of experiments is designed. The pre-testing is used to obtain information that will estimate the appropriate range of the parameter values in CPG_{EA}, and the running time required for the experiments.

4.4.1.1 Exploratory CPG_{EA} Results and Analysis

Table 20 shows the exploratory results. Several small tests exploring the relationships between the performance of CPG_{EA} and the factors discussed

above are summarized. The following is a description of the testing procedure and the results.

1. As discussed in Section 3.3.2.1, the heuristic within CPG_{EA} may play an important role in determining its performance. One of the purposes of this test is to determine which heuristic should be used in subsequent experiments examining other parameters. The heuristics from the XS are natural choices. Since there are 18 possible combinations in the XS procedure rules, testing all of them in CPG_{EA} is impractical. Based on the observation made in testing the XS in previous section, different combinations of rules produce similar results. It is reasonable to assume that using all the combinations of rules in CPG_{EA} would not significantly improve its performance. In the first test, the combination of the procedural rules 1a, 2, 4 and 5 is randomly chosen as the embedded heuristic in CPG_{EA}. This corresponds to heuristic 7 in Table 17.

   The column marked CPG_{EA}-H0 in Table 20 shows the results of this test using all 7 historical cases. The following parameter values are used:

   a. Population size: 51
   b. Crossover rate: 1.0
   c. Mutation rate: 0.10
   d. Stopping criteria: at 60 generations, or if there is no improvement for 30 generations.
   e. Linear ranking parameter: a = 0.5

   These values were selected based on the guideline discussed in Section 2.4.7 and on a test with a small population.

   The results showed that CPG_{EA}-H0 has performance somewhat worse than the XS. It produced better results in projects H1 and H3. On the projects H4, H5, H6 and H7, the results are more costly than the manual solution. Careful examination of the solutions reveals that CPG_{EA}-H0 tends to aggregate many elements into a few groups resulting in greater width and higher unit cost.

Recall that only procedural rules 1, 2 and 3 will produce new plates in the XS; the embedded heuristic in CPG_{EA} is used only on elements affected by the crossover operation. The number of elements available for the arrangement by the heuristic is much smaller in CPG_{EA}. Similar to the observation made in the XS test that heuristics without rule 3 may produce infeasible solutions, it is possible that no new plates or very few plates would be produced with the heuristic since rules 1 and 2 only apply to elements with the same width. Without producing new plates, the stacking process in rule 3, 4 and 5 can only combine elements onto existing plates and would eventually produce plates with widths exceeding the lowest unit cost range. To compensate for this shortcoming, another heuristic containing rule 3 is tested.

	Material Costs in Thousands of NT$		
Project	**CPG_{EA}-H0**	**CPG_{EA}-H1**	**lower bound**
1	710	712	701
2	5741	5739	5676
3	1988	1996	1979
4	3258	3251	3205
5	1228	1228	1213
6	1448	1444	1424
7	5741	5749	5674

Table 20. Results of exploratory CPG_{EA} tests using historical cases

	Material Costs in Thousands of NT$	
Project	**CPG_{EA}-H1**	**lower bound (Known Optimal Solution)**
Simulated 1	1081	1046
Simulated 2	451	389
Simulated 3	1277	1252 (1258)
Simulated 4	446	399 (403)

Table 21. Results of exploratory CPGEA tests using simulated cases

2. Based on the observation made in the first test, the heuristic embedded in CPG_{EA} is changed to heuristic 1 (rules 1a - 2 - 3a - 4 - 5) to encourage the formation of new plates. This corresponds to heuristic 1 in Table 17. The Column marked CPG_{EA}-H1 in Table 20 shows the small test using the modified heuristic. The results improved in project H2, H4, H5 and H6. Although the values of the Δ suggest good performance by CPG_{EA}-H1, none of these solutions is superior to those produced by the XS. This result suggests that there may be room for improvement. However, there is no evidence to suggest that significant improvement is likely to be achieved by applying other heuristics within CPG_{EA}. The next tests which consider the population size and termination criteria will be tested without varying the heuristic.

   Four simulated cases are also generated to test CPG_{EA}-H1. Table 22 shows the thickness and the ranges of width and length used in the four simulated cases. Cases 1 and 2 are generated randomly with the given ranges without knowing the optimal arrangement and solution while Cases 3 and 4 are generated by the algorithm described in Section 4.1. Cases 1 and 2 represent more general situations likely to happen in industry since many real projects cannot be arranged in an ideal way, i.e. no cutting loss, lowest unit cost for every plate and no penalty unit cost applied. On the other hand, Cases 3 and 4 have known optimal solutions so the performance can be measured more accurately.

   As expected, there is a gap in the performance between thin and thick cases suggesting the strong influence of problem characteristic and the penalty cost constraint. Notice the relatively small differences in the lower bound and the real optimal solution values. (See Cases 3 and 4 in Table 21.) This suggests that the lower bound is tight and can serve as a good substitute for the real optimal value when it is not available.

Simulated Cases	Thickness (mm)	Known Optimal Solution?	Width (mm)	Length (mm)
1	40	No	100 - 900	3000 - 9000
2	15			
3	40	Yes		
4	15			

Table 22. Dimensional ranges of the simulated cases

3. A mutation only CPG_{EA} is tested using simulated Cases 1 and 2. This tests the power of using only the embedded heuristic with randomly selected elements. The first two simulated cases are tested using this approach. The generated costs are NT$ 1,252,000 and NT$ 539,000 for simulated case 1 and 2 respectively. They suggest that without the crossover operation to accumulate good schemata or building blocks, the unsystematic use of the heuristic cannot achieve desirable results.

4.4.2 Parameters in CPG_{EA}

As discussed above, many factors are associated with the performance of CPG_{EA}. Among these, three factors - population size, crossover rate and mutation rate - are directly related to the convergence of the EA. Careful selection of these parameters is needed to "optimize" the running efficiency and effectiveness. As reviewed in Section 2.4.7, the selection of the EA parameter values is an optimization problem itself. Although obtaining the "best" setting of parameters for CPG_{EA} is of interest, a "universal" best combination of parameters may not exist since the optimal parameter values are dependent on the problem size and other characteristics. For example, a small population size is effective and efficient for small problems, but it may fail to produce the desired results in problems with a large number of elements. On the other hand, a large

population size would inevitably require a long running time and thus be inefficient for small problems.

A 3^3 factorial experiment is performed using a medium-size simulated case to investigate the effect of these parameters and to estimate the "best" values. One of the subjects in the thin case from the XS testing is selected for the test. Table 23 shows its profile of optimal arrangement. Since the results of thick cases are very close to optimal solutions, any improvement due to the change of a parameter setting may not be obvious. The thin case poses a greater challenge to CPG_{EA} and the results may be more sensitive to parameter settings. Since the testing subject is the same in each trial, there is no need to use the relative index; the absolute cost generated by the algorithm is used as the performance measurement for the statistical testing. The termination criteria are set to stop CPG_{EA} at 1000 generations or if there is no improvement in 100 generations. The values are chosen with reference to the result of the pre-test, the general guidelines in Section 2.4.7, and Falkenauer's experiments.

The experiment is carried out on a PC with Pentium 450 MHz processor and 64 MB RAM. Table 24 shows the results of this experiment. Statistics (see Appendix 4A.3) indicate that only population size has a significant influence on the performance - the largest population size has the best performance. Also, from the breakdown of means, the highest crossover rate produces the best results. However, this difference is not statistically significant at the 0.05 level.

Thick	Width	Length	Weight	Quantity
16	1570	8956	1767	2
16	1765	8917	1977	2
16	1610	8787	1777	2
16	1913	8767	2107	2
16	1941	5588	1363	2
16	1588	4825	963	3
16	1703	4450	952	3
16	1683	4449	941	3
16	1665	4156	870	3
16	1524	3094	593	5

Table 23. Optimal grouping of the thin case chosen for the 3^3 experiment.

Optimal Solution = 384 (1000NT$)		Crossover Rate = 1.0		Crossover Rate = 0.8		Crossover Rate = 0.5	
		Price	Time (min)	Price	Time (min)	Price	Time (min)
PopSize = 251	Mutation Rate = 0.3	415.09	90.0	416.09	87.9	413.72	84.7
		417.00	91.1	410.92	100.3	419.50	93.2
		416.86	97.4	417.17	109.8	419.07	87.9
	Avg.	**416.32**	**92.8**	**414.73**	**99.3**	**417.43**	**88.6**
	Mutation Rate = 0.15	414.56	101.7	415.06	93.9	418.67	88.2
		411.35	107.7	418.91	94.3	417.79	86.5
		417.25	120.8	414.77	100.3	420.78	98.1
	Avg.	**414.38**	**110.1**	**416.25**	**96.1**	**419.08**	**90.9**
	Mutation Rate = 0.01	416.23	98.9	415.38	106.3	417.65	94.3
		413.31	129.7	414.58	98.9	420.38	99.2
		408.34	112.3	415.81	105.2	414.19	85.7
	Avg.	**412.63**	**113.6**	**415.26**	**103.5**	**417.41**	**93.1**
PopSize = 101	Mutation Rate = 0.3	420.50	33.1	416.53	24.0	431.68	29.1
		421.71	27.5	420.04	26.7	421.28	28.6
		417.58	36.2	419.28	29.4	423.96	26.9
	Avg.	**419.93**	**32.3**	**418.62**	**26.7**	**425.64**	**28.2**
	Mutation Rate = 0.15	422.07	35.4	423.20	28.3	414.94	34.2
		422.25	30.2	419.51	29.0	425.38	31.5
		419.74	31.9	421.67	27.4	420.09	29.4
	Avg.	**421.36**	**32.5**	**421.46**	**28.2**	**420.14**	**31.7**
	Mutation Rate = 0.01	423.17	32.4	423.97	34.6	423.19	32.3
		417.55	39.4	416.17	29.9	425.02	36.1
		421.69	34.1	425.85	27.1	421.00	35.9
	Avg.	**420.80**	**35.3**	**421.99**	**30.5**	**423.07**	**34.8**
PopSize = 31	Mutation Rate = 0.3	429.31	7.0	433.32	9.1	445.62	10.5
		428.34	7.2	443.40	9.8	426.94	10.9
		427.97	8.3	427.64	8.8	431.38	8.9
	Avg.	**428.54**	**7.5**	**434.78**	**9.2**	**434.65**	**10.1**
	Mutation Rate = 0.15	440.63	7.4	430.77	10.3	424.07	10.2
		423.77	7.4	424.75	9.0	429.00	11.2
		435.10	8.6	426.43	8.3	432.56	10.0
	Avg.	**433.17**	**7.8**	**427.32**	**9.2**	**428.54**	**10.5**
	Mutation Rate = 0.01	430.36	11.0	434.63	9.2	431.15	19.2
		417.61	9.9	424.91	15.9	412.21	19.9
		444.25	11.3	428.49	12.7	459.27	10.1
	Avg.	**430.74**	**10.7**	**429.34**	**12.6**	**434.21**	**16.4**

Table 24. Results of 3^3 factorial design

As a grouping problem, the number of possible arrangements in the CPG problem expands exponentially as the number of elements increase. Given a medium-size problem in industry, this represents a vast search space. A larger population size encompasses more locations or points; thus making it more effective in finding a good solution. However, as the population size grows so

does the processing time. Table 24 shows that the processing time grows significantly as the population size increases.

Based on the experimental result, the settings of the parameters are chosen as:

Population size: 251

Crossover Rate: 1.0

Mutation Rate: 0.15

These parameter values are used in further experiments that test other factors in the next section. Appendix 4A.4 shows the breakdown of means for all three factors. Although the mutation rate does not have a statistically significant impact on the performance, the mutation rate of 0.15 is picked according to the best mean value among three tested levels.

4.4.3 The Impact of Heuristics, Subject Types and Termination Criteria

Based on the analysis and observation made on the above experiments, it is clear that the population size is more critical than the genetic operator parameter values (i.e. crossover rate and mutation rate). The embedded heuristic is also important since some heuristics are logically inferior if they do not have Rule 3 and are thus limited in creating new plates. Observations made in the pre-test also suggest that the plate thickness and cost penalty threshold characteristics will affect the performance of CPG_{EA}. Another factor that may affect the performance in CPG_{EA} is the termination criteria. More generations may evolve better results. With the parameters selected in the previous experiment, a 2^3 factorial experiment is performed to investigate the impact of different heuristics, problem characteristics and termination criteria.

Table 25 shows two levels of the three factors in the experimental design. The first factor is the heuristic embedded within CPG_{EA}. Heuristic 1 is the same as the one used in the previous experiment, i.e. CPG_{EA}-H1. From the previous experiment, it is clear that degraded performance is caused by the penalty cost

constraint. The second heuristic, CPG_{EA}-H2, is developed to avoid this penalty cost. Inspired by Zulawinski's swapping heuristic (Zulawinski 1995), CPG_{EA}-H2 substitutes the mutation operation with a local search algorithm that swaps elements between two plates, attempting to make their widths equal. Recall that the penalty cost will apply to plates of a given size that weigh less than the threshold. Making the widths even on the same-length plates will produce plates with identical dimension and increase the total weight of the plate set, thus avoiding the penalty cost. The following are the details of the swapping heuristic:

1. After the crossover operation, randomly pick one plate, P1, from the first offspring.
2. Find a plate, P2, with the same length as P1. If there is no plate with the same length, terminate the algorithm.
3. Swap the elements between the two plates if the swapping will make the difference between those two plate widths smaller. Terminate the operation when the two plates have the same width, or swapping any two elements cannot close the gap between the widths of the two plates. The following is the pseudo-code explaining the swapping procedure:

```
Let counter = 0
Do while counter = 0
    Let counter = 1
    For m = 1 to maximum_number_of_elements_in_P1
        For n = 1 to maximum_number_of_elements_in_P2
            If exchanging element m in P1 with element n in P2 will close the
            gap between the widths, then
                exchange element m and element n, and let counter = 0
            If the widths of P1 and P2 are equal, then
                stop the procedure.
        Next n
    Next m
Loop back until counter is not equal to 0.
```

This local search algorithm is always performed after the crossover operation, i.e. the mutation rate is set to 1 in this case.

Factors	Heuristic	Termination	Problem Type
Level 1	Heuristic 1 - CPG_{EA}-H1	100 Generations	Mix
Level 2	Heuristic 2 - CPG_{EA}-H2	200 Generations	3 identical plates

Table 25. Different levels of three factors in the 2^3 factorial design

The second factor in the experiment is the termination criteria. In the previous experiment, the number of generations before CPG_{EA} terminates is between 163 and 464 with an average of 251. It is possible to improve the result by increasing the number of generations before CPG_{EA} terminates. The two levels used in this experiment are 100 and 200 generations. This is the number of generations without improvement before CPG_{EA} terminates. The maximum number of generations is always 1000 in these tests.

Two types of problems are tested in the experiment. The first problem type, marked as "mix" in Table 25, uses the same subject as the previous experiment. As shown in Table 23, its optimal arrangement contains 5 different plate dimensions with each dimension containing two plates, 4 different plate dimensions having three identical plates in each dimension and one dimension containing 5 plates. The second problem type is generated using the algorithm described in Section 4.1 with the thickness of 16 mm and length ranging from 4600 mm to 5100 mm. This algorithm, with controlled length range, produces a problem that has 9 different plate dimensions and three plates in each dimension. Both problem types are thin cases that are likely to invoke the penalty cost constraint.

The results of the experiment are shown in Table 26 and Table 27. The analysis of variance shows that only heuristic has significant impact on the performance on the 0.05 level (see Appendix A.5). The swapping heuristic embedded in CPG_{EA} improves the performance in solving the thin cases, though the processing time is longer.

CPG_{EA}-H1							
		100 Gen.			200 Gen.		
	Optimal Solution	Price	Δ	Time (Min.)	Price	Δ	Time (Min.)
Subject 1	383.89	414.56	7.99%	131.1	413.73	7.77%	253.3
T=16		411.35	7.15%	138.8	411.66	7.23%	234.8
L=3000-9000		417.25	8.69%	155.7	413.43	7.69%	270.5
W=1524-2000		**Avg.**	**7.94%**	**141.9**	**Avg.**	**7.57%**	**252.9**
Subject 2	356.10	388.73	9.16%	149.3	376.57	5.75%	218.2
T=16		381.02	7.00%	218.4	383.80	7.78%	251.7
L=4600-5100		385.99	8.39%	179.6	387.20	8.74%	227.6
W=1524-2000		**Avg.**	**8.19%**	**182.4**	**Avg.**	**7.42%**	**232.5**

Table 26. Results of CPG_{EA}-H1 in the 2^3 factorial design

CPG_{EA}-H2							
		100 Gen.			200 Gen.		
	Optimal Solution	Price	Δ	Time (Min.)	Price	Δ	Time (Min.)
Subject 1	383.89	412.20	7.37%	164.8	409.44	6.65%	362.3
T=16		414.97	8.09%	187.4	413.46	7.70%	289.8
L=3000-9000		406.49	5.89%	197.5	406.58	5.91%	238.9
W=1524-2000		**Avg.**	**6.97%**	**180.6**	**Avg.**	**6.75%**	**297**
Subject 2	356.10	370.77	4.12%	392.4	376.39	5.70%	438.0
T=16		382.87	7.52%	223.8	377.37	5.97%	243.7
L=4600-5100		385.36	8.22%	184.0	373.86	4.99%	601.7
W=1524-2000		**Avg.**	**6.62%**	**266.7**	**Avg.**	**5.55%**	**427.8**

Table 27. Results of CPG_{EA}-H2 in the 2^3 factorial design

The experiment shows that more generations produce better results on average, although the improvement is not statistically significant. The 1000 upper limit of generations is only reached once in the experiment. This suggests that running CPG_{EA} as long as tolerable in practical situations may produce the best result possible. In practice, the termination criteria can also be changed to end the program in a certain amount of time instead of generations.

The different subjects do not influence the results significantly in this experiment. To test the effect of the number of identical plates needed in avoiding the

penalty, another small experiment is performed using CPG_{EA}-H1. One testing subject was generated with the thickness of 28 mm, and the length ranging from 3900 to 6000. The controlled length and thickness ensure that exactly two identical plates are needed for plates of a given length to avoid the penalty cost. Table 28 shows the results of this test. The average Δ of two replicates is 0.64%, which is significantly lower than the average thin cases that need more than two identical plates to avoid penalty. This phenomenon shows the difficulty of the CPG problem increases dramatically as the number of required identical plates increases from two to three. Further tests of the effect on the problem characteristics are performed in the next section when both XS and CPG_{EA} are compared.

	Optimal Solution	Price	Δ	Time (Min.)
Subject 3	597.31	600.82	0.59%	255.9
T=28		601.44	0.69%	271.7
L=3900-6000		**Avg.**	**0.64%**	
W=1524-2000				

Table 28. Test results of the subject that needs two identical plates to avoid penalty cost

4.4.4 Summary of CPG_{EA} Analysis

After the above experiments on CPG_{EA}, several observations can be summarized:

1. The processing time of CPG_{EA} is substantially longer than that of XS.
2. Different heuristics within CPG_{EA} have a significant effect on the performance, while the crossover rate and the mutation rate do not have any significant influence. It is possible to improve CPG_{EA} with different heuristics, especially those that target specific problem types. With local search, CPG_{EA}-H2 has better performance than CPG_{EA}-H1. However, the processing time also increases.

3. The difficulty of the CPG problem increases dramatically as the number of identical plates required to avoid the penalty cost increases from two to three.

4.5 Comparison Between the XS and "Optimized" CPG_{EA}

Both XS and CPG_{EA} are investigated independently in previous sections. In this section, the two approaches are compared for their efficiency and effectiveness. First, new simulated cases are used to compare the performance as well as to investigate the influence of problem characteristics on both approaches. Then, the XS and "optimized" CPG_{EA} are tested with the historical cases to observe their performance under the realistic situations.

4.5.1 Experiment Using Simulated Cases

In previous sections, several varieties of CPG_{EA} are tested with changes in the embedded heuristic, genetic operator values, and number of generations. To compare the performance of CPG_{EA} to the XS, the best CPG_{EA}, i.e. CPG_{EA}-H2 with 200 generations, is chosen for the comparison. Since CPG_{EA} is stochastic in nature, several subjects are needed in the experiment to take the variation into account.

To test the influence of different problem characteristics, three different types of problems are generated. They are all generated using the procedure described in Section 4.1 with changes in thickness and length. Table 29 shows the parameter values used in the generating procedure. The thick and thin-1 types are similar to medium-sized problem in industry while thin-2 type has the length range limited so that the optimal solution will always require three identical plates.

Problem types	Thick	Thin-1	Thin-2
Thickness (mm)	40	16	16
PL_1	3000	3000	4600
PL_2	9000	9000	5100
H	2600		
EW_1	100		
EW_2	900		
Min_Comp	100		
Width	1524 - 2000		

Table 29. Parameter values for generating different problem types

A single-factor, paired comparison experiment is performed using the simulated data. Since the subjects are generated randomly, a paired comparison design is necessary to eliminate the additional source of variability, i.e., the difference between simulated cases (Montgomery 1991, p. 41). Each subject is tested using both the XS and CPG_{EA}. Since running CPG_{EA} is time consuming, only one replication is tested for each subject. For each problem types, five different cases are generated and tested.

The results are shown in Table 30. It is clear that CPG_{EA} is superior in every case. The improvement over XS is especially pronounced for both types of thin problems. The statistical analysis confirms that overall CPG_{EA} has better performance (see Appendix A.7 and Appendix A.8). The Duncan's Multiple Range Test also shows that the performance difference is larger in thin cases compared to thick cases. However, the improvement between the thin-1 and thin-2 cases is not significant.

Problem Type	Cases	Optimal Solutions	XS			CPG_{EA}		
			Price	Δ	Time (min.)	Price	Δ	Time (min.)
Thick	11	1036.03	1050.90	1.44%	1.6	1041.39	0.52%	262.9
	12	939.04	955.08	1.71%	1.5	946.87	0.83%	238.2
	13	921.05	934.61	1.47%	1.5	929.78	0.95%	293.1
	14	943.94	957.63	1.45%	1.5	948.76	0.51%	201.8
	15	866.75	879.73	1.50%	1.4	870.33	0.41%	251.9
			Avg.	**1.51%**	**1.5**	**Avg.**	**0.64%**	**249.6**
Thin-1	21	453.37	498.90	10.04%	1.6	471.95	4.10%	200.3
	22	353.53	405.13	14.59%	1.5	368.51	4.24%	294.5
	23	331.53	394.77	19.08%	1.5	343.20	3.52%	233.3
	24	420.05	479.44	14.14%	1.6	443.68	5.62%	229.4
	25	354.43	402.18	13.47%	1.5	370.69	4.59%	301.8
			Avg.	**14.26%**	**1.5**	**Avg.**	**4.41%**	**251.9**
Thin-2	31	353.45	411.60	16.45%	1.6	367.25	3.90%	541.6
	32	312.61	365.17	16.81%	1.5	328.13	4.97%	386.5
	33	331.18	374.49	13.08%	1.4	357.40	7.92%	331.2
	34	370.65	432.14	16.59%	1.7	388.37	4.78%	367.7
	35	336.70	386.17	14.69%	1.6	355.82	5.68%	234.9
			Avg.	**15.52%**	**1.5**	**Avg.**	**5.45%**	**372.4**

Table 30. Results of the single-factor, paired comparison experiment

Several observations can be made from the above results:

1. Overall, CPG_{EA} is more effective, especially in the thin cases where the penalty unit cost constraint is applied. Using CPG_{EA} in the thin cases can significantly reduce the material costs. CPG_{EA} solutions are approximately 10% lower in performance index Δ than the XS solutions for these cases.

2. The XS is fast in all cases and also effective in thick cases. It would be a good solution technique in situations that require fast reaction time such as estimation and testing for what-if scenarios. However, the XS does not perform well when all elements are thin.

3. The variation of Δ generated by the XS is large in the thin cases, ranging from 10% to 19%. Also, different problem types have greater impact on the XS than on CPG_{EA}. This shows that the XS is more sensitive to problem characteristics.

4. CPG_{EA} requires substantially more solution time than does the XS. In one case (Case 31), it takes more than 9 hours for CPG_{EA} to generate a solution.

4.5.2 Experiment using Historical Cases

Finally, both XS and CPG_{EA} are tested using the historical cases from LK. The tests compare both approaches in the real situations. Due to the time required to generate solutions, only one trial is made for each case using CPG_{EA}. Also, the theoretical lower bound is used instead of the optimal solution value since the real optimal value is unknown.

Table 31 shows the results of the XS and CPG_{EA} tests with historical cases. Several inferences can be made from this simple comparison:

1. Both XS and CPG_{EA} solve the problems effectively. None of them exceed 3% of the theoretical lower bound. Solutions generated by CPG_{EA} are all less than 1% above the theoretical lower bound.
2. Although CPG_{EA} is a stochastic method and the results may vary each time, its performance is more consistent than the deterministic XS. However, CPG_{EA} does not always perform better than the XS (see Cases H3 and H5).
3. In Case H1, the solution from CPG_{EA} is very close to the lower bound, which is not always feasible. This suggests that it might be possible to obtain the optimal solution in cases that contain relatively few elements.

Case	Theoretical Lower Bound	CPG_{EA}			XS		
		Cost	Time (min.)	Δ	Cost	Δ	Time (min.)
H1	701	702.49	88	0.21%	717.10	2.28%	1.1
H2	5676	5729.87	436	0.95%	5735.50	1.04%	3.3
H3	1979	1993.93	219	0.75%	1988.38	0.45%	1.0
H4	3205	3229.38	188	0.76%	3247.91	1.34%	1.6
H5	1213	1224.45	127	0.94%	1223.92	0.91%	0.7
H6	1424	1432.09	165	0.57%	1448.75	1.69%	0.8
H7	5674	5704.57	263	0.54%	5732.69	1.04%	1.5
Average			**212.3**	**0.67%**		**1.25%**	**1.4**

Table 31. Comparison between XS and CPGEA using historical cases

CHAPTER 5 SUMMARY AND CONCLUSION

This dissertation researches a cutting plan generation problem that is important to the steel construction industry. The problem is described, mathematically modeled and compared to existing problems in the literature. In this problem, the complex cost structure and virtually infinite number of combinations of material ordering make conventional discrete optimization techniques inefficient. Two solution approaches, a multi-expert system and an evolutionary algorithm, are proposed and investigated. Both solution techniques are extended in ways that have significance beyond the CPG problem. In this chapter, the significance of the research to the industry and to more general heuristic optimization methodologies is described; the performance and practical implication of both approaches are explained; and finally, future research is discussed.

5.1 Contribution to the Steel Construction Industry

Cutting plan generation is a critical design operation in the steel construction industry. It is the pivotal point among architectural design, material preparation, manufacturing and final assembly. A timely and economical solution to the CPG problem will not only reduce the total project cost but also provide advantages in different operations such as lead time reduction, accuracy in cost estimation, and creation of design alternatives.

In this dissertation, several related problems in classic operations research literature are studied and compared to the CPG problem. These classic problems include cutting and packing problems, and cluster problems. Although the CPG problem is distinct from those problems due to its lack of fixed stock sizes and complex cost structure, they provided useful guidelines for formulating the mathematical model of the CPG problem. Since the CPG problem is atypical, two new solution approaches were developed, a multi-expert system and an evolutionary algorithm called CPG_{EA}. The unique features and effectiveness of both solution approaches are discussed in the next sections.

Assuming a perfect optimal condition can be achieved, a theoretical lower bound is developed. It assumes that there is no cutting loss, no penalty cost and the lowest unit cost always applies. As shown in Section 4.4.1.1 and 4.5.2, this lower bound is very close to the real optimal value in the simulated cases. This suggests that it can serve as an excellent substitution for the optimal value in problems without known optimal solutions. Any solution approach to the CPG problem can use this lower bound as a reference in testing its performance.

A procedure is developed to create the simulated problems with known optimal solutions. This procedure creates plates that satisfy the optimal conditions and then divides the plates into individual elements. Parameters used in this procedure can be altered to simulate different scenarios for experiments investigating the influence of various problem characteristics. This problem generating procedure was used to test the solution methodologies specific to this dissertation, but it is independent of the solution technique and may be useful to gauge future solution approaches including company expert (or intuitive) solutions.

The mathematical model developed for the CPG problem uses a matrix representation to denote the group membership of each element. This representation forms the basic representation scheme in both solution approaches. In this model, the maximum number of plates is calculated. The number is critical in practice for limiting the problem size. The model clarifies some of the complex characteristics of the CPG problem including its need for integer variables and non-linear functions.

The mathematical model and solution approaches of this dissertation have practical significance. Lien-Kang Heavy Industrial Company has adopted the XS approach presented in this dissertation. Experiments show excellent results in solving real cases provided by company LK. The solutions generated are very close to the theoretical lower bounds, which may not be attainable in those cases.

A series of experiments was performed using simulated cases to investigate the performance of the XS and CPG_{EA}, as well as to study the influence of problem characteristics. The results show that if three or more identical plates are needed to avoid the penalty cost, the performance of both solution approaches decreases. The impact of this complicating factor is more severe to the XS than to the CPG_{EA}. The average percentage above the optimal value increases more than 12% using the XS, and about 4% using CPG_{EA}. This indicates that CPG_{EA} is more robust and resistant to changes in problems. On the other hand, the XS has shorter processing time, making it a better choice in situations that require prompt response. The discoveries and practical implication of both XS and CPG_{EA} are explained in more detail in the next sections.

5.2 Contribution to Expert System Development

A multi-expert system solution to the CPG problem is developed. This expert system is atypical. Multiple experts were used to develop the knowledge base and they contributed on an equal basis. The knowledge gathered from these experts is synthesized into different procedural rules. These rules can be combined in different ways to produce different solution processes representing different experts' solution techniques. The knowledge base consisted of the set of these processes rather than a set of production rules. Rather than determining the best set of processes in advance, the XS applies all of them to a problem and the final solution is selected based on its quality. In this way, problem solution was accomplished using a custom control mechanism rather than using inference and a generic technique (e.g., backward chaining).

A series of experiments was performed to investigate the effectiveness and efficiency of the expert system. The XS shows excellent results in solving the historical cases provided by company LK. With an average of 1.25% above the theoretical lower bound, these solutions generated by the XS are better than the manual solutions generated by the experts in most cases. Similar performance is also found in the simulated problems that are not likely to require multiple

identical plates to avoid the penalty cost. However, the XS does not perform well in cases requiring multiple identical plates. It generates solutions that are, on average, between 13% to 16% above the optimal solutions in those cases.

The XS is very fast. Most of the problems tested are solved within 2 minutes. This short processing time makes the XS very useful in practice despite its deficiency in solving certain types of problem. It is possible to expand the role of the XS in other operations such as bidding and preliminary design. The XS can also be used for preparing the real cutting plan in cases that are not likely to require multiple identical plates to avoid penalty cost; these cases usually have a thick thickness.

Since this solution approach was adopted by LK, its evaluation also included feedback from its users. Referring to Figure 1, the XS is used in the detailed design phase to help LK's engineers track project components, determine steel elements, and prepare the material plan. This plan is also used to facilitate material ordering and production activities.

The practical benefits of this expert system are not limited to controlling material costs. LK's engineers report that they spend much less time working on each project. This time reduction is due in part to a reduction in the amount of time spent recording data as compared with the manual solution. Obviously, there has also been a great reduction in the time required to create a new material plan. Excluding simple project data entry, LK's engineers report that they typically spent an entire day creating a solution. The expert system creates solutions of higher quality almost instantly. Nearly as significant is the time saved when changes to the project are necessary. Such changes may require re-computing the material requirements for at least part of the material plan. Use of the system also makes it easier to document any change in cost that might result from a change in the project. Based on the experiences of LK, these other benefits are at least as important as controlling material cost.

Although the XS is developed by extracting knowledge from experts in LK, this material cutting problem faced by LK is faced by other competing firms in the

steel construction industry. Engineers in LK also suggested that CPG problem can be found in factories producing heavy-lifting cranes. The approach of developing a XS for the problem and for embedding an expert system solution based on expertise existing at the company could be applied at other firms in the steel construction industry.

5.3 Contribution to Evolutionary Algorithm Development

The other solution approach proposed in this dissertation is an evolutionary algorithm, CPG_{EA}. Adopted from Falkenauer's Grouping Genetic Algorithm (GGA), CPG_{EA} has a two-chromosome representation scheme and a special crossover operation. However, the selection scheme follows the prevailing rank-based method. The embedded heuristic within crossover is selected from one set of the procedural rules from the XS.

The CPG problem is effectively solved by CPG_{EA}[9]. The solutions it generates are less than 1% above the optimal solutions in the simulated cases with thick plates. In the more difficult cases involving the penalty cost constraint, it generates solutions an average of 5% above the optimal solution values compared to 15% for solutions generated by the XS. CPG_{EA} shows a great robustness in solving the difficult cases and its performance is very consistent despite its stochastic nature. Judging from the experiment with the historical cases, a single run is sufficient in practice, making it an exceptional choice for generating material cutting plans.

CPG_{EA} consistently outperforms the XS except in two of the historical cases. Careful examination reveals that the local search heuristic tends to arrange elements into plates such that all plates have approximately the same width. This strategy is designed to create identical plates so that the penalty cost can be avoided. However, the strategy may group elements with different lengths together resulting in cutting loss. With thick plates where the penalty cost is not

[9] Without specific indication, CPG_{EA} in this section represents CPG_{EA} in its final form, i.e. CPG_{EA}-H2.

likely to apply, the arrangement may produce plates with unnecessary cutting loss. In the two exceptional cases, CPG_{EA} produced plates with cutting loss, resulting in a slightly more expensive arrangement than those from the XS. Being a critical part in CPG_{EA}, the embedded heuristic can be modified to improve the algorithm's performance. One modification includes bypassing the local search procedure for plates exceeding the threshold of the penalty cost. The improvement on the heuristic can be made whenever new information about the problem is discovered.

There is one drawback of CPG_{EA} -- long processing time. Depending upon the termination criteria, the processing time of CPG_{EA} ranges from 200 minutes to 540 minutes in the simulated cases and 88 minutes to 436 minutes in the historical cases. These are substantially longer than the processing time of the XS, which is generally less than 2 minutes. Although the processing time of CPG_{EA} is still practical in the business use, shortening the time without compromising the performance will make it even more powerful. Several tactics might be adopted to shorten or control the processing time. These tactics are described in the next section.

The development and testing of CPG_{EA} also serves as an investigation of the more general GGA. Several insights in CPG_{EA} may be applied to GGA as well:

1. As discovered from the experiments, the population size and the embedded heuristic are critical to the performance of CPG_{EA}. Larger population size will produce better results. And if the embedded heuristic can produce better solutions, the overall performance improves. The embedded heuristic can be designed and improved using problem specific knowledge.
2. As Falkenauer indicated (Falkenauer 1998), the power of GGA is from the representation of groups as an additional chromosome and the crossover operation conducted on this chromosome. The experiments show the GGA framework can be adapted to the CPG problem effectively. However, given the criticality of the embedded heuristic, the importance of the representation scheme and the crossover operation remains to be studied.

3. Given the complexity of the problem, its characteristics have great impact on the performance. The performance decreases significantly when the problems become more difficult and involve the penalty cost constraint.

The above findings are not only useful in understanding the CPG problem and improving CPG_{EA}, but also beneficial in solving other grouping problems. The general framework used in CPG_{EA} can be adopted to solve other similar grouping problems.

5.4 Future Research

The two solution approaches proposed in this dissertation solve the CPG problem effectively. Both solution approaches developed in this research add new knowledge for solving similar grouping problems. The XS is unique in its deployment of procedure rules synthesized from multiple experts. It shows a practical example in designing an expert system using multiple experts as peers. Adopted from the grouping genetic algorithm, CPG_{EA} is unique in its deployment of two-chromosome representation, crossover operation and the embedded heuristic. The discoveries made from the investigation of CPG_{EA} explore the strengths and weaknesses of the GGA, which has been shown to have great potential for solving different kinds of grouping problems.

Nevertheless, the research can be expanded to deepen the understanding of general grouping problems and explore new methods in solving the CPG problem as well as some related grouping problems. There are four areas that will be explored in the future.

1. Improvement of the XS and development of general framework for procedural rule-based multi-expert system:

   One of the XS's advantages is that new procedural rules can be developed and added to the system. Based on the findings in CPG_{EA}, the local search algorithm improves the solutions and can be easily incorporated into the XS. Also, rules can be developed and added to create multiple identical plates

when the weight of a single plate is likely to be less than the penalty threshold. More procedural rules can be developed and added once there is any new discovery of problem specific knowledge or a different expert's knowledge is consulted. To make the manipulation of procedural rules more manageable, a general framework for the procedural rule-based multi-expert system may be developed. This will require a new control structure in the use of procedural rules that is fast and flexible.

With the short processing time, the XS could be extended to support the bidding and preliminary design phase. The XS could help companies prepare better project cost estimates quickly. This will improve the bidding decisions and thus improve the chances of winning profitable projects while avoiding those that would be marginal or even unprofitable.

2. Improvement of CPG_{EA}:

   There are several simple tactics which can be employed to shorten the processing time. The tradeoff between shorter and possibly more predictable processing time must be compared to its effect on solution quality. The following tactics will be investigated:

   a. Use a time limit as the termination criterion alone or in combination with a "no improvement" criterion. Depending upon individual company's tolerance in generating the cutting plan, it can set the maximum running time in the termination criteria to control the processing time.

   b. Set a target value and the maximum running time, terminate the program once CPG_{EA} generates a solution value equal to or lower than the target value. The target value can be set by referencing the theoretical lower bound developed in this dissertation.

   c. Adjust the population size according to the problem size. The experiments developed in this dissertation assume a medium problem size in the industry. The population size of 251 is set according to the experiment.

(See Section 4.4.2 for detail.) In practice, a project may consist of many small sub-cases with different plate thickness. (See Table 14 for the dimensional ranges of historical cases.) A smaller population size may be sufficient to produce similar results. However, further research is needed to determine the interaction effect of problem size and population size.

Also, new heuristic can be developed to improve the effectiveness and efficiency of CPG_{EA}. For an immediate improvement, a filter can be added to the heuristic to bypass the local search heuristic on the plates with weights over the penalty threshold. This should improve the solution as well as shorten the processing time.

3. Expanding knowledge of CPG_{EA} and the related GGA

   Other than the tactics to improve the performance of CPG_{EA}, some future research will expand the understanding of CPG_{EA}. The first is to know the impact of problem difficulty or complexity. In this dissertation, the author found that the requirement of generating three identical plates to avoid the penalty cost decreases the performance of CPG_{EA}. The impact of increasing the number of identical plates required is unknown. The exploration can be easily carried out through experiments utilizing the procedure that generates different simulated cases.

   More study can be implemented on CPG_{EA} to expand the understanding of CPG_{EA} and GGA in general. First, a different heuristic or search strategy can be implemented at different stages of execution. For example, through the control of mutation rate, the portion of offspring that undergo local search can be reduced in the beginning and gradually increased through execution. This strategy may accelerate the execution without sacrificing the solution quality.

   The other study that can be conducted is comparing CPG_{EA} to a GA of single chromosome with the same embedded heuristic to investigate Falkenauer's claim of GGA's supremacy, e.g., using two-point crossover on the component

string (see Section 3.3.2.1) and the heuristic on the elements of affected groups.

4. Applying approaches to related problems:

   The CPG problem belongs to the family of grouping problems. The solution approaches proposed in this dissertation can be useful in applying to other similar grouping problems. These include the machine-component grouping problem in cellular manufacturing system (Billo et al. 1996; Hwang and Sun 1996; Venugopal and Narendran 1992), the line balancing problem, the equal piles problem, etc. (Falkenauer 1998).

Other than the four areas stated above, more research can be conducted to validate the results, and the solution approaches can be adopted in further development of a decision support system that expands beyond the process of cutting plan generation.

BIBLIOGRAPHY

Adamowicz, M., and Albano, A. (1976). “A Solution of the Rectangular Cutting-Stock Problem.” *IEEE Transaction of Systems, Man and Cybernetics*, SMC-6, 302-310.

Altenberg, L. (1994). “Emergent Phenomena in Genetic Programming.” *Proceedings of 3rd Annual Conference on Evolutionary Programming*, San Diego, CA, 233-241.

Anderberg, M. R. (1973). *Cluster Analysis for Applications*, Academic Press, New York.

Angeline, P. J. (1997). “Parse Trees.” Handbook of Evolutionary Computation, T. Bäck, D. B. Fogel, and Z. Michalewicz, eds., Institute of Physics Publishing and Oxford University Press, New York, C1.6:1-3.

Askin, R. G., Dror, M., and Vakharia, A. J. (1994). “Printed Circuit Board Family Grouping and Component Allocation for a Multimachine, Open-Shop Assembly Cell.” *Naval Research Logistics*, 41, 587-608.

Bäck, T. (1996). *Evolutionary Algorithms in Theory and Practice*, Oxford University Press, New York.

Bäck, T. (1997). “Introduction.” Handbook of Evolutionary Computation, T. Bäck, D. B. Fogel, and Z. Michalewicz, eds., Institute of Physics Publishing and Oxford University Press, New York, B1.1:1-4.

Badiru, A. B. (1992). *Expert Systems Applications in Engineering and Manufacturing*, Prentice Hall, Englewood Cliffs, New Jersey.

Beasley, D. (1997). “Possible Applications of Evolutionary Computation.” Handbook of Evolutionary Computation, T. Bäck, D. B. Fogel, and Z. Michalewicz, eds., Institute of Physics Publishing and Oxford University Press, New York, A1.2:1-10.

Beasley, J. E. (1985a). “An Algorithm for the Two-Dimensional Assortment Problem.” *European Journal of Operational Research*, 19, 253-261.

Beasley, J. E. (1985b). “Algorithms for Unconstrained Two-Dimensional Guilloline Cutting.” *Journal of the Operational Research Society*, 36, 297-306.

Beasley, J. E. (1985c). “An Exact Two-Dimensional Non-Guillotine Cutting Tree Search Procedure.” *Operations Research*, 33(1), 49-64.

Bellack, P. M. (1984). “Dofasco's Microcomputer Decision Support System.” *Planning Review*, 12(4), 21-23.

Bhaskar, G., and Narendran, T. T. (1996). “Grouping PCBs for Set-Up Reduction: a Maximum Spanning Tree Approach.” *International Journal of Production Research*, 34(3), 621-632.

Billo, R. E., Bidanda, B., and Tate, D. (1996). “A Genetic Cluster Algorithm for the Machine-Component Grouping Problem.” *Journal of Intelligent Manufacturing*, 7, 229-241.

Blanchard, B. S., and Fabrycky, W. J. (1998). *Systems Engineering and Analysis*, Prentice Hall, Upper Saddle River, New Jersey.

Booker, L. B., Fogel, D. B., Whitley, D., and Angeline, P. J. (1997). “Recombination.” Handbook of Evolutionary Computation, T. Bäck, D. B. Fogel, and Z. Michalewicz, eds., Institute of Physics Publishing and Oxford University Press, New York, C3.3:1-27.

Chambers, M. L., and Dyson, R. G. (1976). “The Cutting Stock Problem in the Flat Glass Industry - Selection of Stock Sizes.” *Operational Research Quarterly*, 27(4), 949-957.

Chen, H., and Flann, N. S. (1994). “Parallel Simulated Annealing and Genetic Algorithms: a Space of Hybrid Methods.” Parallel Problem Solving from Nature-PPSN III (Proceedings of International Conference on Evolutionary Computation and 3rd Conference on Parallel Problem Solving from Nature.), Y. Davidor, H.-P. Schwefel, and R. Männer, eds., Springer, Berlin, 428-438.

Christofides, N., and Whitlock, C. (1977). “An Algorithm for Two Dimensional Cutting Problem.” *Operations Research*, 25(1), 30-44.

Cowgill, M. (1993). “Monte Carlo Validation of Two Genetic Clustering Algorithms,” Ph. D. Dissertation, Virginia Polytechnic Institute and State University, Blacksburg.

Dasgupta, D., and Michalewicz, Z. (1997). “Evolutionary Algorithms - An Overview.” Evolutionary Algorithms in Engineering Applications, D. Dasgupta and Z. Michalewicz, eds., Springer, Berlin, 3-28.

Davis, L. (1991). *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, New York.

De Jong, K., Fogel, D. B., and Schwefel, H.-P. (1997). “A History of Evolutionary Computation.” Handbook of Evolutionary Computation, T. Bäck, D. B. Fogel, and Z. Michalewicz, eds., Institute of Physics Publishing and Oxford University Press, New York, A2.3:1-12.

de la Maza, M., and Tidor, B. (1993). “An Analysis of Selection Procedures with Particular Attention Paid to Proportional and Boltzmann Selection.” *Proceedings of 5th International Conference on Genetic Algorithms*, Urbana-Champaign, IL, 124-131.

Deb, K. (1997). “Introduction (Selection).” Handbook of Evolutionary Computation, T. Bäck, D. B. Fogel, and Z. Michalewicz, eds., Institute of Physics Publishing and Oxford University Press, New York, C2.1:1-4.

Duran, B. S., and Odell, P. L. (1974). *Cluster Analysis: A Survey*, Springer-Verlag, New York.

Dyckhoff, H. (1990). “A Typology of Cutting and Packing Problem.” *European Journal of Operational Research*, 44, 145-159.

Dyckhoff, H., and Finke, U. (1992). *Cutting and Packing in Production and Distribution: A Typology and Bibliography*, Physica-Verlag, Heidelberg.

DynaSys Inc. (1998). “AccuFAB.” , DynaSys Inc.

Falkenauer, E. (1991). “A Genetic Algorithm for Grouping.” *Proceedings of the Fifth International Symposium on Applied Stochastic Models and Data Analysis*, Granada, Spain, 198-206.

Falkenauer, E. (1994). “A New Representation and Operators for Genetic Algorithms Applied to Grouping Problems.” *Evolutionary Computation*, 2(2), 123-144.

Falkenauer, E. (1996). “A hybrid grouping genetic algorithm for bin packing.” *Journal of Heuristics*, 2(1), 5-30.

Falkenauer, E. (1998). *Genetic Algorithms and Grouping Problems*, John Wiley & Sons, New York.

Feigenbaum, E. A. (1979). “Themes and Case Studies of Knowledge Engineering.” Expert Systems in the Micro-Electronic Age, D. Michie, ed., Edinburg University Press, 3-25.

Fogel, D. B. (1997a). “Finite-State Representations.” Handbook of Evolutionary Computation, T. Bäck, D. B. Fogel, and Z. Michalewicz, eds., Institute of Physics Publishing and Oxford University Press, New York, C1.5:1-3.

Fogel, D. B. (1997b). “Real-Valued Vectors.” Handbook of Evolutionary Computation, T. Bäck, D. B. Fogel, and Z. Michalewicz, eds., Institute of Physics Publishing and Oxford University Press, New York, C1.3:1-2.

Fogel, D. B., and Angeline, P. J. (1997). “Guidelines for a Suitable Encoding.” Handbook of Evolutionary Computation, T. Bäck, D. B. Fogel, and Z. Michalewicz, eds., Institute of Physics Publishing and Oxford University Press, New York, C1.7:1-2.

Fogel, D. B., and Stayton, L. C. (1994). “On the Effectiveness of Crossover in simulated Evolutionary Optimization.” *BioSystems*, 32, 171-182.

Freisleben, B. (1997). “Metaevolutionary Approaches.” Handbook of Evolutionary Computation, T. Bäck, D. B. Fogel, and Z. Michalewicz, eds., Institute of Physics Publishing and Oxford University Press, New York, C7.2:1.

Garey, M. R., and Johnson, D. S. (1979). *Computers and Intractability: A guide to the Theory of NP-Completeness*, W. H. Freeman and Company, New York.

Gasser, L. (1991). “Social conceptions of knowledge and action: DAI foundations and open systems semantics.” *Artificial Intelligence*, 47, 107-138.

Gates, G. H., Jr., Merkle, L. D., and Lamont, G. B. (1995). “Simple Genetic Algorithm Parameter Selection for Protein Structure Prediction.” *1995 IEEE*

International Conference on Evolutionary Computation (ICEC '95), Perth, Western Australia, 620-624.

Gemmill, D. D., and Sanders, J. L. (1990). “Approximate Solutions for the Cutting Stock 'Portfolio' Problem.” *European Journal of Operational Research*, 44, 167-174.

Gilmore, P. C., and Gomory, R. E. (1961). “A Linear Programming Approach to the Cutting Stock Problem.” *Operations Research*, 9, 849-859.

Gilmore, P. C., and Gomory, R. E. (1963). “A Linear Programming Approach to the Cutting Stock Problem - Part II.” *Operations Research*, 11, 863-888.

Gilmore, P. C., and Gomory, R. E. (1965). “Multistage Cutting Stock Problems of Two and More Dimensions.” *Operations Research*, 13, 94-120.

Gilmore, P. C., and Gomory, R. E. (1966). “The Theory and Computation of Knapsack Functions.” *Operations Research*, 14, 1045-1074.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley Publishing Company, Inc., Reading, Massachusetts.

Goldberg, D. E. (1994). “Genetic and Evolutionary Algorithms Come of Age.” *Communications of the ACM*, 37(3), 113-119.

Grefenstette, J. (1986). “Optimization of Control Parameters for Genetic Algorithms.” *IEEE transactions on systems, man, and cybernetics*, SMC-16, 122-128.

Grefenstette, J. (1997). “Rank-Based Selection.” Handbook of Evolutionary Computation, T. Bäck, D. B. Fogel, and Z. Michalewicz, eds., Institute of Physics Publishing and Oxford University Press, New York, C2.4:1-6.

Haessler, R. W., and Sweeney, P. E. (1991). “Cutting Stock Problems and Solution Procedures.” *European Journal of Operational Research*, 54, 141-150.

Hanachi, C. (1996). “A Cooperative Information System to Support Multi-Expert Systems Development.” *Expert Systems With Applications*, 11(4), 561-569.

Hansen, P., Jaumard, B., and Mladenovic, N. (1995). “How to Choose K Entities Among N.” *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 19, 105-116.

Hart, W. E., and Belew, R. (1991). “Optimizing an Arbitrary Function is Hard for the Genetic Algorithm.” *Proceedings of 4th International Conference on Genetic Algorithms*, San Diego, CA, 190-195.

Hesser, J., and Männer, R. (1990). “Towards an Optimal Mutation Probability for Genetic Algorithms.” Parellel Problem Solving from Nature, H.-P. Schwefel and R. Männer, eds., Springer-Verlag, Berlin, 23-32.

Hinterding, R., and Khan, L. (1995). “Genetic Algorithms for Cutting Stock Problems: with and without Contiguity.” Lecture Note in Artificial Intelligence, X. Yao, ed., Springer, New York, 166-186.

Hinxman, A. I. (1976). “Problem Reduction and the Two-Dimensional Trim-Loss Problem.” *AISB Summer Conference*, University of Edinburgh, 158-165.

Hinxman, A. I. (1980). “The Trim-Loss and Assortment Problems: A Survey.” *European Journal of Operational Research*, 5, 8-18.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor.

Huang, K.-C. (1998). “Personal Email Communication.” , Lien-Kang Heavy Industrial Company.

Hung, C.-Y., and Sumichrast, R. T. (2000). “*A Multi-Expert System for Material Cutting Plan Generation*.” *Expert Systems with Applications*, 19, (forthcoming).

Hwang, H., and Sun, J.-U. (1996). “Genetic-algorithm-based heuristic for the GT cell formation problem.” *Computers & Industrial Engineering*, 30(4), 941-955.

Ibaraki, T. (1997). “Combination with Local Search.” Handbook of Evolutionary Computation, T. Bäck, D. B. Fogel, and Z. Michalewicz, eds., Institute of Physics Publishing and Oxford University Press, New York, D3.2:1-5.

INS Engineering Corporation. (1998). “FRED-EX.” , INS Engineering Corporation.

Jármai, K. (1990). “Decision Support System on IBM PC for Design of Economic Steel Structures Applied to Crane Girders.” *Thin-Walled Structures*, 10, 143-159.

Jensen, R. E. (1969). “A Dynamic Programming Algorithm for Cluster Analysis.” *Operations Research*, 12, 1034-1057.

Jiang, J. H., Wang, J. H., Chu, X., and Yu, R. Q. (1997). “Clustering Data Using a Modified Integer Genetic Algorithm (IGA).” *Analytica Chimica Acta*, 354, 263-274.

Kaufman, L., and Rousseeuw, P. J. (1990). *Finding Groups in Data: An Introduction to Cluster Analysis*, John Wiley & Sons, Inc., New York.

Lai, K. K., and Chan, W. M. (1997). “An Evolutionary Algorithm for the Rectangular Cutting Stock Problem.” *International Journal of Industrial Engineering*, 4(2), 130-139.

Li, J., Li, T., and Guo, L. (1995). “The Fuzzy Reasoning of Multi-Expert System.” *The Third International Symposium on Uncertainty Modeling and Analysis and Annual Conference of the North American fuzzy Information Processing Society*, College Park, Maryland, 749-751.

Marconi, R. (1970). “Generalization of a mathematical procedure for the solution of 'two-dimensional cutting problems'.” *Centro Studi IBM of Pisa, Technical Report no. 7*.

Martello, S., and Toth, P. (1990). “Lower Bounds and Reduction Procedures for the Bin Packing Problem.” *Discrete Applied mathematics*, 22, 59-70.

McCart, C. D. (1991). “Expert Systems for Financial Analysis of University Auxiliary Enterprises,” Ph. D. Dissertation, Virginia Polytechnic Institute and State University, Blacksburg.

Michalewicz, Z. (1992). *Genetic Algorithms + Data Structures = Evolution Programs*, Springer, Berlin.

Michalewicz, Z., and Nazhiyath, G. (1995). “Genocop III: A Co-Evolutionary Algorithm for Numerical Optimization Problems with Nonlinear Constraints.” *Proceedings of the 2nd IEEE International Conference on Evolutionary Computation*, Perth, 647-651.

Milligan, G. W., and Cooper, M. C. (1987). “Methodology Review: Clustering Methods.” *Applied Psychological Measurement*, 11, 329-354.

Montgomery, D. C. (1991). *Design and Analysis of Experiments*, John Wiley & Sons, New York.

Ng, K.-C. (1990). “Consensus in a Multi-Expert System.” *Cooperation 1990 ACM Eighteenth Annual Computer Science Conference*, Washington, DC, 351-357.

Nikolopoulos, C. (1997). *Expert Systems: Introduction to First and Second Generation and Hybrid Knowledge Based Systems*, Marcel Dekker, Inc., New York.

O'Leary, D. E. (1998). “Knowledge Acquisition from Multiple Experts: An Empirical Study.” *Management Science*, 44(8), 1049-1058.

Page, E. (1975). “A Note on a Two-Dimensional Dynamic Programming Problem.” *Operational Research Quarlerly*, 26(2), 321-324.

Palmer, C. C., and Kershenbaum, A. (1994). “Representing Trees in Genetic Algorithms.” *Proceedings of the IEEE International Conference on Evolutionary Computation*, 379-384.

Pasley, G. P., and Roddis, W. M. K. (1996). “Decision Support Environment for Structural Steel.” *The 1996 12th Conference on Analysis and Computation*, Chicago, IL, 371-382.

Porto, V. W. (1997). “Evolutionary Programming.” Handbook of Evolutionary Computation, T. Bäck, D. B. Fogel, and Z. Michalewicz, eds., Institute of Physics Publishing and Oxford University Press, New York, B1.4:1-10.

Radcliffe, N. J. (1997). “Schema Processing.” Handbook of Evolutionary Computation, T. Bäck, D. B. Fogel, and Z. Michalewicz, eds., Institute of Physics Publishing and Oxford University Press, New York, B2.5:1-10.

Rahman, A. F. R., and Fairhurst, M. C. (1996). “Recognition of Handwritten Characters with a Multi-Expert System.” *Workshop on Handwriting Analysis and Recognition - a European Perspective*, 6/1 - 6/4.

Richardson, J. T., Palmer, M. R., Liepins, G., and Hilliard, M. (1989). “Some Guidelines for Genetic Algorithms with Penalty Functions.” *Proceedings of the Third International Conference on Genetic Algorithms*, Los Altos, CA, 191-197.

Romesburg, H. C. (1984). *Cluster Analysis for Researchers*, Lifetime Learning Publications, Belmont, California.

Ronald, S. (1997). “Robust Encodings in Genetic Algorithms.” Evolutionary Algorithms in Engineering Applications, D. Dasgupta and Z. Michalewicz, eds., Springer, Berlin, 29-44.

Rudolph, G. (1997). “Evolution Strategies.” Handbook of Evolutionary Computation, T. Bäck, D. B. Fogel, and Z. Michalewicz, eds., Institute of Physics Publishing and Oxford University Press, New York, B1.3:1-6.

Sarin, S. C. (1983). “Two-Dimensional Stock Cutting Problems and Solution Methodologies.” *Journal of Engineering for Industry*, 105, 155-160.

Schwefel, H.-P. (1997). “Advantages (and Disadvantages) of Evolutionary Computation Over Other Approaches.” Handbook of Evolutionary Computation, T. Bäck, D. B. Fogel, and Z. Michalewicz, eds., Institute of Physics Publishing and Oxford University Press, New York, A1.3:1-2.

Schwefel, H. P. (1995). *Evolution and Optimum Seeking*, John Wiley, Chichester, UK.

Seber, G. A. F. (1984). *Multivariate Observations*, Wiley, New York.

Siedlecki, W., and Sklanski, J. (1989). “Constrained Genetic Optimization via Dynamic Reward-Penalty Balancing and Its Use in Pattern Recognition.” *Proceedings of the Third International Conference on Genetic Algorithms*, Los Altos, CA, 141-150.

Sneath, P. H. A., and Sokal, R. R. (1973). *Numerical Taxonomy. The Principles and Practice of Numerical Classification.*, W. H. Freeman, San Francisco.

Soremekun, G., Gürdal, Z., Haftka, R. T., and Watson, L. T. (1996). “Improving Genetic Algorithm Efficiency and Reliability in the Design and Optimization of Composite Structures.” , Technical Report, Virginia Polytechnic Institute and State University, Blacksburg, VA.

Spears, W. M., and De Jong, K. A. (1991). “On the Virtues of Parameterized Uniform Crossover.” *Proceedings of 4th International Conference on Genetic Algorithms*, San Diego, CA, 230-236.

Stansfield, W. (1991). *Theory and Problems of Genetics*, McGraw Hill.

Stecke, K. E. (1983). “Formulation and Solution of Nonlinear Integer Production Planning Problems for Flexible Manufacturing Systems.” *Management Science*, 29(3), 273-288.

Sumichrast, R. T. (1995). “Class Notes of MSCI 5474.” , Class Note, Virginia Polytechnic Institute and State University, Blacksburg, VA.

Sweeney, P. E., and Paternoster, E. R. (1992). “Cutting and Packing Problems: A Categorized, Application-Orientated Research Bibliography.” *Journal of the Operational Research Society*, 43(7), 691-706.

Tsutsumida, T., Matsui, T., Noumi, T., and Wakahara, T. (1996). “Results of IPTP Character Recognition Competitions and Studies on Multi-Expert System for Hanprinted Numeral Recognition.” *IEICE Transactions on information and Systems*, E79-D(5), 429-435.

Tzafestas, S., and Adrianopoulos, A. (1993). “Knowledge Acquisition for Expert System Design.” Expert Systems in Engineering Applications, S. Tzafestas, ed., Springer-Verlag, Berlin, 3-24.

Tzafestas, S. G., Kokkinaki, A. I., and Valavanis, K. P. (1993). “An Overview of Expert Systems.” Expert Systems in Engineering Applications, S. Tzafestas, ed., Springer-Verlag, Berlin, 3-24.

Vasko, F. J. (1989). “A Computational Improvement to Wang's Two-Dimensional Cutting Stock Algorithm.” *Computers and Industrial Engineering*, 16, 109-115.

Venugopal, V., and Narendran, T. T. (1992). “A Genetic Algorithm Approach to the Machine-Component Grouping Problem With Multiple Objectives.” *Computers & Industrial Engineering*, 22(4), 469-480.

Vinod, H. D. (1969). “Integer Programming and Theory of Grouping.” *JASA*, June, 506-519.

Viswanathan, K. V., and Bagchi, A. (1988). “An Exact Best-First Search Procedure for the Constrained Rectangular Guillotine Knapsack Problem.” *Proceedings of the American Association for Artificial Intelligence*, 145-149.

Wang, P. Y. (1983). “Two Algorithms for Constrained Two-Dimensional Cutting Stock Problems.” *Operations Research*, 31, 573-586.

Whitley, D. (1997). “Permutations.” Handbook of Evolutionary Computation, T. Bäck, D. B. Fogel, and Z. Michalewicz, eds., Institute of Physics Publishing and Oxford University Press, New York, C3.3:14-20.

Wood III, W. H., and Agogino, A. M. (1996). “Case-based Conceptual Design Information Server for Concurrent Engineering.” *Computer-Aided Design*, 28(5), 361-369.

Yusuf, K. O., McCarthy, T. J., Smith, N. J., and Tizani, W. T. (1997). “A Decision Support System for Construction-Led Design.” *Computing in Civil Engineering: Proceedings of the Fourth Congress Held in Conjunction with AEC Systems '97*, 183-189.

Zulawinski, B. (1995). “The Swapping Heuristic,” Masters Thesis, Michigan State University.

Appendix Statistics Generated by SAS Program

A.1 Statistics (ANOVA) of 2^2 factorial experiment in testing the XS

```
1                          ANOVA of XS 2^2 Factorial Design                          1
                                                20:59 Wednesday, February 16, 2000

                                The GLM Procedure

                             Class Level Information

                           Class              Levels    Values

                           CostStructure           2    0 1

                           Thickness               2    0 1


                              Number of observations    40
1                          ANOVA of XS 2^2 Factorial Design                          2
                                                20:59 Wednesday, February 16, 2000

                                The GLM Procedure

 Dependent Variable: Price   Price

                                              Sum of
      Source                       DF         Squares     Mean Square    F Value    Pr > F

      Model                         3      0.14328805      0.04776268     201.11    <.0001

      Error                        36      0.00855001      0.00023750

      Corrected Total              39      0.15183806


                        R-Square     Coeff Var      Root MSE    Price Mean
```

0.943690	21.29102	0.015411	0.072383

Source	DF	Type III SS	Mean Square	F Value	Pr > F
CostStructure	1	0.00018396	0.00018396	0.77	0.3846
Thickness	1	0.14302554	0.14302554	602.21	<.0001
CostStruct*Thickness	1	0.00007855	0.00007855	0.33	0.5688

A.2 Statistics (breakdown of means) of 2^2 factorial experiment in testing the XS

1 ANOVA of XS 2^2 Factorial Design 1

20:59 Wednesday, February 16, 2000

Breakdown of Means and Other Descriptive Statistics

---------- Effect=Overall ----------

Cost Structure	Thickness	Mean of PRICE	Std. Dev. of PRICE
.	.	0.072383	0.062396

---------- Effect=THICKNESS ----------

Cost Structure	Thickness	Mean of PRICE	Std. Dev. of PRICE
.	0	0.13218	0.020216
.	1	0.01259	0.007426

---------- Effect=COSTSTRUCTURE ----------

Cost Structure	Thickness	Mean of PRICE	Std. Dev. of PRICE
0	.	0.070238	0.063987
1	.	0.074527	0.062349

A.3 Statistics (ANOVA) of 3^3 factorial experiment in testing CPG_{EA}

1 ANOVA of EA 3^3 Factorial Design 1

20:59 Wednesday, February 16, 2000

The GLM Procedure

Class Level Information

Class	Levels	Values
PopSize	3	31 101 251
CrossOver	3	1 0.5 0.8
Mutation	3	0.3 0.01 0.15

Number of observations 81

1 ANOVA of EA 3^3 Factorial Design 2

20:59 Wednesday, February 16, 2000

The GLM Procedure

Dependent Variable: Prices Prices

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	18	3471.407245	192.855958	4.65	<.0001
Error	62	2571.923403	41.482636		
Corrected Total	80	6043.330648			

R-Square	Coeff Var	Root MSE	Prices Mean
0.574420	1.523057	6.440701	422.8800

Source	DF	Type III SS	Mean Square	F Value	Pr > F
CrossOver	2	101.982168	50.991084	1.23	0.2996
Mutation	2	13.430855	6.715428	0.16	0.8509
PopSize	2	3248.766076	1624.383038	39.16	<.0001
CrossOver*Mutation	4	56.793051	14.198263	0.34	0.8484
PopSize*CrossOver	4	8.763862	2.190965	0.05	0.9947
PopSize*Mutation	4	41.671234	10.417808	0.25	0.9079

A.4 Statistics (breakdown of means) of 3^3 factorial experiment in testing CPG_{EA}

1 ANOVA of EA 3^3 Factorial Design 1
20:59 Wednesday, February 16, 2000

Breakdown of Means and Other Descriptive Statistics

---------- Effect=CROSSOVER ----------

PopSize	Cross Over	Mutation	Mean of PRICES	Std. Dev. of PRICES	Variance of PRICES
.	0.5	.	424.462	10.0212	100.424
.	0.8	.	422.194	7.4951	56.177
.	1	.	421.984	8.4801	71.912

---------- Effect=MUTATION ----------

PopSize	Cross Over	Mutation	Mean of PRICES	Std. Dev. of PRICES	Variance of PRICES
.	.	0.01	422.827	10.6436	113.287
.	.	0.15	422.410	6.7821	45.998
.	.	0.3	423.403	8.5226	72.635

---------- Effect=Overall ----------

PopSize	Cross Over	Mutation	Mean of PRICES	Std. Dev. of PRICES	Variance of PRICES
.	.	.	422.880	8.69147	75.5416

---------- Effect=POPSIZE ----------

PopSize	Cross Over	Mutation	Mean of PRICES	Std. Dev. of PRICES	Variance of PRICES
31	.	.	431.254	9.32103	86.8816
101	.	.	421.445	3.48774	12.1643
251	.	.	415.941	2.90470	8.4373

---------- Effect=CROSSOVER*MUTATION ----------

PopSize	Cross Over	Mutation	Mean of PRICES	Std. Dev. of PRICES	Variance of PRICES
.	0.5	0.01	424.895	14.0872	198.449
.	0.5	0.15	422.585	5.6683	32.130
.	0.5	0.3	425.906	9.4488	89.281
.	0.8	0.01	422.197	7.0677	49.952
.	0.8	0.15	421.675	5.2551	27.616

Breakdown of Means and Other Descriptive Statistics

---------- Effect=CROSSOVER*MUTATION (continued) ----------

PopSize	Cross Over	Mutation	Mean of PRICES	Std. Dev. of PRICES	Variance of PRICES
.	0.8	0.3	422.708	10.2180	104.408
.	1	0.01	421.389	10.5924	112.200
.	1	0.15	422.969	9.4209	88.754

.	1	0.3	421.595	5.5769	31.102

---------- Effect=POPSIZE*CROSSOVER ----------

PopSize	Cross Over	Mutation	Mean of PRICES	Std. Dev. of PRICES	Variance of PRICES
31	0.5	.	432.465	13.3138	177.258
31	0.8	.	430.481	5.9629	35.556
31	1	.	430.815	8.1860	67.010
101	0.5	.	422.950	4.5628	20.819
101	0.8	.	420.690	3.2819	10.771
101	1	.	420.695	2.0309	4.124
251	0.5	.	417.972	2.5048	6.274
251	0.8	.	415.410	2.1599	4.665
251	1	.	414.442	2.9998	8.999

---------- Effect=POPSIZE*MUTATION ----------

PopSize	Cross Over	Mutation	Mean of PRICES	Std. Dev. of PRICES	Variance of PRICES
31	.	0.01	431.430	13.9772	195.362
31	.	0.15	429.676	5.7073	32.573
31	.	0.3	432.656	7.0272	49.382
101	.	0.01	421.956	3.2683	10.682
101	.	0.15	420.984	2.9291	8.580
101	.	0.3	421.395	4.4425	19.736
251	.	0.01	415.095	3.2833	10.780
251	.	0.15	416.570	2.8799	8.294
251	.	0.3	416.158	2.6541	7.044

A.5 Statistics (ANOVA) of 2^3 factorial experiment in testing CPG_{EA}

1 ANOVA of EA 2^3 Factorial Design 1
20:59 Wednesday, February 16, 2000

The GLM Procedure

Class Level Information

Class	Levels	Values
Heuristic	2	1 2
Num_OfGen_	2	100 200
Subjects	2	1 3

Number of observations 24

1 ANOVA of EA 2^3 Factorial Design 2
20:59 Wednesday, February 16, 2000

The GLM Procedure

Dependent Variable: Price_Diff_ Price_Diff_

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	6	0.00147744	0.00024624	1.84	0.1502
Error	17	0.00227199	0.00013365		
Corrected Total	23	0.00374943			

R-Square	Coeff Var	Root MSE	Price_Diff_ Mean
0.394043	16.18009	0.011561	0.071449

Source	DF	Type III SS	Mean Square	F Value	Pr > F
Heuristic	1	0.00096460	0.00096460	7.22	0.0156
Num_OfGen_	1	0.00024740	0.00024740	1.85	0.1914
Subjects	1	0.00009645	0.00009645	0.72	0.4074
Heuristic*Num_OfGen_	1	0.00000312	0.00000312	0.02	0.8804
Heuristic*Subjects	1	0.00012129	0.00012129	0.91	0.3541
Num_OfGen_*Subjects	1	0.00004457	0.00004457	0.33	0.5712

A.6 Statistics (breakdown of means) of 2^3 factorial experiment in testing CPG_{EA}

1 ANOVA of EA 2^3 Factorial Design 1
20:59 Wednesday, February 16, 2000

Breakdown of Means and Other Descriptive Statistics

---------- Effect=HEURISTIC ----------

Heuristic	Num_Of Gen_	Subjects	Mean of PRICE_ DIFF_	Std. Dev. of PRICE_DIFF_	Variance of PRICE_ DIFF_
1	.	.	0.077789	0.009294	.000086376
2	.	.	0.065110	0.012915	.000166790

---------- Effect=NUM_OFGEN_ ----------

Heuristic	Num_Of Gen_	Subjects	Mean of PRICE_ DIFF_	Std. Dev. of PRICE_DIFF_	Variance of PRICE_ DIFF_
.	100	.	0.074660	0.013644	.000186172
.	200	.	0.068239	0.011498	.000132194

---------- Effect=Overall ----------

Heuristic	Num_Of Gen_	Subjects	Mean of PRICE_ DIFF_	Std. Dev. of PRICE_DIFF_	Variance of PRICE_ DIFF_
.	.	.	0.071449	0.012768	.000163019

---------- Effect=SUBJECTS ----------

Heuristic	Num_Of Gen_	Subjects	Mean of PRICE_ DIFF_	Std. Dev. of PRICE_DIFF_	Variance of PRICE_ DIFF_
.	.	1	0.073454	0.008489	.000072056
.	.	3	0.069445	0.016126	.000260032

---------- Effect=HEURISTIC*NUM_OFGEN_ ----------

Heuristic	Num_Of Gen_	Subjects	Mean of PRICE_ DIFF_	Std. Dev. of PRICE_DIFF_	Variance of PRICE_ DIFF_
1	100	.	0.080639	0.008576	.000073551
1	200	.	0.074939	0.009848	.000096978
2	100	.	0.068681	0.015818	.000250223

Breakdown of Means and Other Descriptive Statistics

---------- Effect=HEURISTIC*NUM_OFGEN_ ----------
(continued)

Heuristic	Num_Of Gen_	Subjects	Mean of PRICE_ DIFF_	Std. Dev. of PRICE_DIFF_	Variance of PRICE_ DIFF_
2	200	.	0.061539	.009279522	.000086110

---------- Effect=HEURISTIC*SUBJECTS ----------

Heuristic	Num_Of Gen_	Subjects	Mean of PRICE_ DIFF_	Std. Dev. of PRICE_DIFF_	Variance of PRICE_ DIFF_
1	.	1	0.077546	0.005593	.000031282
1	.	3	0.078032	0.012594	.000158603
2	.	1	0.069362	0.009331	.000087062
2	.	3	0.060857	0.015378	.000236470

---------- Effect=NUM_OFGEN_*SUBJECTS ----------

Heuristic	Num_Of Gen_	Subjects	Mean of PRICE_ DIFF_	Std. Dev. of PRICE_DIFF_	Variance of PRICE_ DIFF_
.	100	1	0.075302	0.009732	.000094717
.	100	3	0.074018	0.017716	.000313872

.	200	1	0.071606	0.007457	.000055611
.	200	3	0.064871	0.014422	.000208001

A.7 Statistics (ANOVA) of the experiment in comparing the XS and CPG_{EA}

1 Comparison of EA and XS 1

20:59 Wednesday, February 16, 2000

The GLM Procedure

Class Level Information

Class	Levels	Values
Subject	3	1 2 3

Number of observations 15

1 Comparison of EA and XS 2

20:59 Wednesday, February 16, 2000

The GLM Procedure

Dependent Variable: DifferenceOfXSandEA DifferenceOfXSandEA

Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	2	0.02758789	0.01379395	18.70	0.0002
Error	12	0.00885012	0.00073751		
Corrected Total	14	0.03643801			

R-Square	Coeff Var	Root MSE	DifferenceOfXSandEA Mean
0.757118	39.17742	0.027157	0.069318

Source	DF	Type III SS	Mean Square	F Value	Pr > F
Subject	2	0.02758789	0.01379395	18.70	0.0002

A.8 Duncan's Multiple Range Test and other statistics in comparing the XS and CPG_{EA}

1 Comparison of EA and XS 3
20:59 Wednesday, February 16, 2000

The GLM Procedure

Duncan's Multiple Range Test for DifferenceOfXSandEA

NOTE: This test controls the type I comparisonwise error rate, not the experimentwise error rate.

Alpha	0.05
Error Degrees of Freedom	12
Error Mean Square	0.000738

Number of Means	2	3
Critical Range	.03742	.03917

Means with the same letter are not significantly different.

Duncan Grouping	Mean	N	Subject
A	0.10077	5	3
A			
A	0.09851	5	2
B	0.00868	5	1

Means and Descriptive Statistics

Subject	Mean of DIFFERENCEOFXSANDEA	Std. Dev. of DIFFERENCEOFXSANDEA	Variance of DIFFERENCEOFXSANDEA
.	0.06932	0.051017	.002602715
1	0.00868	0.002077	.000004314
2	0.09851	0.035628	.001269365
3	0.10077	0.030641	.000938852

VITA

Chang-Yu Hung was born in Kaohsiung, Taiwan. He received his Ph. D. in Industrial and Systems Engineering in 2000 from Virginia Polytechnic Institute and State University. He received his Masters of Science in Industrial Engineering from Bradley University, Peoria Illinois, and Bachelors of Science in Civil Engineering from Tamkang University, Taiwan.

During his graduate study in Virginia Tech, Chang-Yu Hung worked as a graduate teaching assistant and graduate research assistant in the Industrial and Systems Engineering Department, the Engineering Dean's Office, and Public Service Program. Before his graduate study in Virginia Tech, he worked as a management information system engineer in Lien Kang Heavy Industrial Co. He also worked as a part-time Erho (a Chinese stringed instrument) performer in Kaohsiung City Chinese Music Orchestra, and was the president of the Chinese Music Club during his junior year in Tamkang University.

Chang-Yu Hung is a member of Alpha Pi Mu (Honor Society of Industrial and Systems Engineering). His paper had won the First Place Award in the Production and Operations Management track at the 1996 Southeastern Chapter of the Institute for Operations Research and the Management Sciences. He was presented with an Outstanding Graduate Student award upon graduation from Bradley University.