APPROACHES TO CUTTING STOCK AND STRIP PACKING PROBLEMS

A THESIS

submitted in partial fulfillment of the requirements for the award of the degree

of

DOCTOR OF PHILOSOPHY

by

JAYA THOMAS

DISCIPLINE OF COMPUTER SCIENCE & ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY INDORE

MAY 2014

INDIAN INSTITUTE OF TECHNOLOGY INDORE

CANDIDATE'S DECLARATION

I hereby certify that the work, which is being presented in the thesis, entitled **APPROACHES TO CUTTING STOCK AND STRIP PACKING PROBLEMS**, in fulfillment of the requirement for the award of the degree of **DOCTOR OF PHILOSOPHY** and submitted in the **DISCIPLINE OF COMPUTER SCIENCE & ENGINEERING, Indian Institute of Technology Indore**, is an authentic record of my own work carried out during the period January, 2011 to May, 2014 under the supervision of Dr. Narendra S. Chaudhari, Professor, Discipline of Computer Science & Engineering, Indian Institute of Technology Indore.

The matter presented in this thesis has not been submitted by me for the award of any other degree of this or any other institute.

Dated: Signature of the Student with date

(JAYA THOMAS)

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Dated: Signature of the Thesis Supervisor

(NARENDRA S. CHAUDHARI)

JAYA THOMAS has successfully given his Ph.D. Oral Examination held on .....

Signature of Thesis Supervisor	Convener, DPGC
Date:	Date:

Signature of PSPC Member #1	Signature of PSPC Member #2	Signature of External Examiner
Date:	Date:	Date:

Acknowledgements

I would like to take this opportunity to express my heartfelt gratitude to a number of persons who in one or the other way contributed making this time as learn-able, enjoyable and bearable. At first, I would like to thank my supervisor Prof. Narendra S. Chaudhari, who was a constant source of inspiration during my work, without his constant guidance and research directions this research work could not be completed. His continuous support and encouragement has motivated me to remain streamline in my research work. I am grateful to Dr. Abhishek Srivastava for all his extended help and support. A highly motivating person always ready to listen and sort out things with a warm gesture. I am thankful to Dr. Ram Bilas Pachori and Dr. Anand Parey, my research committee members for taking out some valuable time to evaluate my progress all these years. Their good comments and suggestions helped me to improve my work at various stages. I wish to thank all my colleagues from the Discipline of Computer Science and Engineering Neetesh Saxena, Prakash Sharma, Rajkumar Jain, Pramod, Saumya, Neha, Vipul, Tanveer and Piyush for their suggestions and friendship. I also wish to thank the cooperative staff members Jaya Vakade, Shagufta Rahim, Shailendra Verma, Prahalad S. Panwar and library members for their extended help at all times. A special mention of Neetesh Saxena for some constructive opinions, efficiency and assistance on various matters, and Pooja Jain, Kratika Pathak for all the fun time we had during this tenure. I want to thank Pooja Jain (Asst. Professor, VITS Indore), surely one of my greatest supports, for her understanding and constant encouragement. I extend my sincere thanks to many government bodies like CSIR, CICS and IIT Indore to help me with financial support to attend international conferences, which gave me a right platform at the right time and helped a lot to groom my research work. I would like to thank my family, especially my mother and father for always believing in me, for their continuous source of inspiration and their support in my decisions. Without whom I could not have made it here.

Dated: (Jaya Thomas)

Dedicated to my parents

Abstract

Cutting and packing problems are progressively encountered in several manufacturing industries. The growing need for automation that may result in economic saving and better utilization of raw stock, are the major challenges faced by these industries. This thesis presents a number of methods for generating high quality solutions for these cutting and packing challenges. The thesis addresses two problems: (i) one dimensional cutting stock problem that deals with generating patterns for cutting the available raw stock that result in minimum trim loss and, (ii) strip packing problem that involves packing of small items into a large container (called strip) such that the resulting height of packing layout is minimized.

A constructive framework to solve large complex problems is presented with methods that lead to high material utilization. The framework is analyzed, and its effectiveness is illustrated on different datasets. We investigate different parameters for this framework, like low demand ratio, feasibility for other dimensions, etc. Metaheuristic and heuristic strategies are explored for effective column generation techniques in order to stabilize and accelerate the solution process, when applied to one dimensional cutting stock problem. Dynamism feature of a genetic algorithm is used to improve the solution convergence rate to a great extent, which controls the random behavior to an acceptable level.

A new placement strategy is proposed for the effective layout of small rectangles into a container. The obtained solution is improved by applying a metaheuristic technique that evolves a better placement sequence for items. Further, the data structure implementation improves the scalability to large problem instances. The main challenge is to develop systems of higher generality, which can intelligently select, evolve or combine search methods (heuristics) to operate upon a wider range of problems and their instances. Hence, we proposed

a new search technique that couples genetic algorithm with constructive hyper-heuristic approach. It investigates different low level heuristics, which are capable of producing good solutions for packing problems. Experimental evidence indicates that the hyper-heuristic can operate on a wide range of problems to produce some competitive results. We also demonstrate the capability of identifying the effectiveness of the low-level heuristics. This research follows the direction and contributes towards achieving the goal of exploring and automating the design of search systems. These facilitate the design and development of a similar automation system for the same or other domains.

List of Publications

International Journal

[1] Thomas J., Chaudhari N.S. (2014), A new metaheuristic genetic based placement algorithm for 2D-strip packing, Journal of Industrial Engineering International, Springer, 10(2), 1-16.

[2] Thomas J., Chaudhari N.S. (2014), Design of efficient packing system using genetic algorithm based on hyper-heuristic approach, Advances in Engineering Software, Elsevier, 73(1), 45-52.

[3] Thomas J., Chaudhari N.S. (2014), An integrated genetic algorithm approach to 1D-cutting stock problem, International Journal of Operational Research, Inderscience. (accepted)

[4] Thomas J., Chaudhari N.S., An integrated approach for trim loss minimization in 1D-cutting multiple stock optimization problem, Operation Research, Springer. (under review)

[5] Thomas J., Chaudhari N.S., Decomposition technique based on genetic algorithm for textile industries optimization problem, IEEE Transactions on Evolutionary Computing.(under review)

International Conferences

[1] Thomas J., Chaudhari N.S. (2013), Selection of efficient crossover operator for metaheuristic approach for 2D strip packing, IEEE International Conference on System, Man and Cybernetics (SMC'13), Manchester, U.K., pp. 415-420.

[2] Thomas J., Chaudhari N.S. (2013), Hybrid approach for 2D strip packing problem using genetic algorithm, International Work Conference on Artificial Neural Network (IWANN'13), Part I, LNCS 7902, Tenerife, Spain, pp. 566-574.

[3] Thomas J., Chaudhari N.S., Saxena N. (2013), Genetic based effective column generation for 1D-cutting stock problem, 5^{th} IEEE International Conference on Modeling, Simulation and Applied Optimization (ICMSAO'13), Hammamet, Tunisia, pp. 1-5.

[4] Thomas J., Chaudhari N.S. (2012), Genetic based bounded knapsack for column generation in 1D-cutting stock Problem, 3^{rd} IEEE International Conference on Emerging Applications of Information Technology (EAIT'12), ISI Kolkatta, India, pp. 268-274.

[5] Thomas J., Chaudhari N.S. (2012), An analytical approach for column generation for one dimensional cutting stock problem, ACM CUBE International Information Technology Conference (CUBE'12), Pune, India, pp. 90-94.

[6] Thomas J., Chaudhari N.S. (2012), Placement strategy for trim minimization in one dimensional cutting stock, 7^{th} IEEE Conference on Industrial Electronics and Applications (ICIEA'12), Singapore, pp. 1362-1365.

Contents

List of Figures xviii
List of Tables xx
List of Algorithms xxi
List of Abbreviations xxi

1 Introduction **1**
1.1 Motivation and Scope 2
1.1.1 Objectives 3
1.1.2 Contributions 4
1.2 Organization of the Thesis 5

2 Background and Related Work **9**
2.1 Classification of C&P Problem 9
2.1.1 Dyckhoff Typology 10
2.1.2 Wäscher's Improved Typology for C&P Problems 14
2.1.3 Lodi *et al.* Sub Typology 15
2.2 Cutting Problems Solution Methodology 15
2.2.1 Exact Approach 16
2.2.2 Heuristic Approach 18
2.2.3 Metaheuristic Approach 20
2.3 Packing Problems Solution Methodology 21
2.3.1 Exact Approach 22
2.3.2 Heuristic Approach 23
2.3.3 Metaheuristic Approach 25
2.3.4 Hyper-heuristic Approach 27
2.4 Chapter Summary 28

3 Column Generation and 1D-Cutting Stock Problem 29

3.1 Introduction 29

3.2 Terminology 30

3.2.1 Column Generation 30

3.2.2 Genetic Algorithm 31

3.3 The Mathematical Model 32

3.4 Algorithm Formulation 33

3.5 Minimize Number of Different Patterns 34

3.5.1 Encoding Scheme: Chromosome Representation 36

3.5.2 Fitness Function 38

3.5.3 Penalty Function 39

3.5.4 Selection Mechanism 40

3.5.5 Adaptive Crossover Mutation Operator 40

3.6 Computational Results 43

3.6.1 Dataset Description 43

3.6.2 A Comparison with Integrated Approach 45

3.6.3 Comparison with other Metaheuristic Approaches 46

3.7 Result Analysis 47

3.8 Chapter Summary 48

4 The Multiple Length 1D-Stock Problem 49

4.1 Introduction 49

4.2 Terminology 50

4.2.1 Branch-Cut-Price 50

4.2.2 Branching Strategy 51

4.2.3 Rounding Procedure 52

4.3 Problem and Mathematical Formulation 52

4.4 Proposed Methodology 54

4.4.1 Pattern Generation Approach to Accelerate Column Generation 54

4.4.2 Computational Selection and Updation 57

4.5 Computational Results 59

4.5.1 Data Instances 59

4.5.2 Assessing the Performance of Proposed Heuristic Against Other Existing Heuristics 60
4.6 Assessing the Performance of Proposed Heuristic Against Exact Approach 63
4.7 Chapter Summary 66

5 Metaheuristic Approach to 2D-Strip Packing 67
5.1 Introduction 67
5.2 Terminology 68
5.2.1 Two Dimensional Strip Packaging 68
5.2.2 Biased Random Key Based Genetic Algorithm 70
5.3 Two Dimensional Placement Heuristic 71
5.3.1 Placement Approach 71
5.3.2 Empty Block Creation 72
5.3.3 The Placement Position Search Strategy 75
5.3.4 Merge Rectangle Routine 76
5.3.5 Metaheuristic Enhancement to Strip Packing 77
5.3.6 Crossover Function 78
5.3.7 Fitness Evaluation 79
5.4 Improvement on Existing Genetic Algorithm Approach 80
5.5 Comparative Evaluation of the Packing Strategy 82
5.5.1 Dataset Description 82
5.5.2 Comparison with Other Metaheurisitic Approaches . . . 84
5.5.3 Statistical Analysis for Heuristic Approach 87
5.5.4 Comparison with Genetic Based Approaches 89
5.5.5 Statistical Analysis 90
5.5.6 Algorithm Complexity 90
5.6 Chapter Summary 91

6 Hyper-heuristic Approach to 2D-Strip Packing 93
6.1 Introduction 93
6.2 Theoretical Background 95
6.2.1 Hyper-heuristic 95
6.3 Mathematical Model 96
6.4 Proposed Technique 98

6.4.1 Problem Decomposition 98
6.5 Heuristic Decision 103
6.5.1 Hyper-heuristic Combined with Model 103
6.5.2 GA Approach to Proposed Model 106
6.6 Experimental Result 108
6.7 Chapter Summary 113

7 Conclusion **115**
7.1 Discussion 115
7.2 Future Work 116

Bibliography **117**

List of Figures

3.1 One dimensional cutting stock problem with standard stock type. 30
3.2 Column generation GA sub-routines. 36
3.3 Chromosome representation. 38
3.4 Trim loss produced by different approaches. 44
3.5 Statistical analysis of integrated LP-based and proposed approach. 44
3.6 Statistical analysis for adaptive crossover. 45

4.1 Graphical interpretation of execution time with exact approaches. 63

5.1 A view of 2D-strip packing. 68
5.2 Biased random key sequencing. 70
5.3 Creation of empty rectangle blocks. 72
5.4 Summary of proposed strategy. 74
5.5 Strip packing with gap. 75
5.6 Merging empty rectangular block (x-coordinate). 76
5.7 Merging empty rectangular block (y-coordinate). 77
5.8 General overview of proposed hybrid approach. 77
5.9 Maximal preservative crossover 79
5.10 Solution for dataset instances. 81
5.11 Study of (a) number of generation (b) frequency of occurrence of different values of best fitness in 50 runs of the algorithm. 85
5.12 Means plot and Tukey's confidence intervals (CI) for the evaluated algorithms. 89
5.13 Means plot and Tukey confidence intervals (CI) for the meta-heuristic algorithms. 90

6.1 Relationship between hyper-heuristic and problem instances. . . 95

6.2 Dynamic level creation at the initial stage (a) initial (b) dynamic levels. 98
6.3 Merging of dynamic level. 100
6.4 Subsequent placement of levels. 101
6.5 Low level heuristic operating on problem instance with 3 cases. . 104
6.6 Best and average solution against solution cutting (a) 300 pieces (b) 1000 pieces. 105
6.7 Violation of constraints for initial population. 106
6.8 A comparison of the (a) HSA, (b) BBFM and (c) Proposed using problem C7.1 from Hopper and Turton instances. 110
6.9 Trim loss (%-gap) analysis for Burke instances. 110

List of Tables

2.1 Different typology for C&P problems. 11
2.2 Improved typology by Wäsher *et al.* [3] for intermediate problem types for output maximization problem (small items to be packed to larger items). 12
2.3 Improved typology by Wäscher *et al.* [3] for intermediate problem types for output minimization problem (small items to be packed to larger items). 13

3.1 Trim loss and number of columns generated for LP-based and proposed approach. 43
3.2 Comparison of trim loss % for different metaheuristic approaches. 46
3.3 Computation results of metaheuristics for random instances generated by CUTGEN1 software. 47
3.4 Effect of adaptive crossover mutation on subproblem evaluation. 47

4.1 Notation and interpretation. 52
4.2 Random instances for CSP. 58
4.3 Real world instances from paper tube industry in Japan Matsumoto *et al.*(2011) [120]. 59
4.4 Real world instances from fiber industry in Japan Matsumoto *et al.*(2011) [120]. 59
4.5 Comparison amongst heuristic algorithm on trim loss and time (in sec.). 61
4.6 Exact approach performance using real world data (tube factory) from Matsumoto *et al.*(2011) [120]. 62
4.7 Performance of Exact approach using fiber industry instances from Matsumoto *et al.*(2011) [120]. 62

4.8 Heuristic algorithm performance on real world data (paper tube factory). 64

5.1 Computational results for Jakobs, Babu & Babu and C test instances. 83
5.2 Computational result for test instance by Burke. 84
5.3 Computational results for Nice and Path test instances. 86
5.4 Computational results for large test instances. 86
5.5 Computational results for large test instances. 88
5.6 Comparison of %-gap for genetic based approaches for Hopper & Turton test instances. 90

6.1 Computational results for Jakobs, Babu & Babu and C test instances. 109
6.2 Computational result for Burke test instances. 111
6.3 Computational result for Pisinger and Sigurd test instances. . . 111
6.4 Computational results for Nice & Path large test instances. . . . 111
6.5 Comparison between GPA and proposed hyper-heuristic large test instances. 113

List of Algorithms

1 Standard Genetic Algorithm 32
2 Proposed Algorithm 35
3 Integrated Heuristic with Integer Programming 56
4 Genetic Placement Approach (GPA) Algorithm 73
5 Placement Position Search Strategy 76
6 MPX_CROSSOVER (*parent1*, *parent2*) 77
7 MergeRoutine 102
8 PlacementRoutine 102

List of Abbreviations

1D-CSP	One Dimension Cutting Stock Problem
1D-MCSP	One Dimension Multiple Cutting Stock Problem
2D-SPP	Two Dimension Strip Packing Problem
ACO	Ant Colony Optimization
ANOVA	Analysis of Variance
BBF	Bidirectional Best Fit
BBFM	Bidirectional Best Fit Modified Heuristic
BP	Branch and Price
BCP	Branch Cut Price
BF	Best Fit
BKP	Bounded Knapsack Problem
C&P	Cutting and Packing
CG	Column Generation
FH	Fast Heuristic
GA	Genetic Algorithm
GPA	Genetic Based Placement Approach
GRASP	Greedy Randomized Adaptive Search Procedure
ILP	Integer Linear Programming
LLH	Low Level Heuristic
LP	Linear Programming
NP	Non Polynomial
RMP	Restricted Master Problem
SA	Simulated Annealing
SHP	Sequential Heuristic Procedure
TS	Tabu Search

Chapter 1

Introduction

Cutting and Packing (C&P) problems are common day-to-day problems that have existed for centuries, whether it is associated with household work like packing of utensils, animals, metal block or to pharmaceutical packing, shipping industries, newspaper article arrangement, etc. The tasks at those times were handled mechanically by some simple calculation or intuition with not much emphasis on scrap minimization. However, the problem was first reported by Kantorovich in 1939 [1], but published in 1960, for scrap minimization in One Dimensional Cutting Stock Problem (1D-CSP). The problem has an objective of minimizing the amount of raw stock used, in order to meet the customer demand of variable length stock, at the cost of minimum scrap produced.

The problem finds importance from the industrial perspective, where there is an ever demanding requirement for efficient material utilization and waste minimization. The commercial perspective shows that the main cause of the waste generation is often influenced by the higher production rate. This has a vital impact on the economy of any country. In the changing scenario with the improvement in living standard, the demand for textile and clothing is expected to grow. Use of traditional technology in cutting and packaging are a bottleneck to productivity. The major concern for these firms is the amount of waste produced. This necessitates the need for effective and efficient semi-automate cutting and packing systems in order to keep up with the increasing demand for the growth of these industries. Some fully automated systems with appropriate machineries were deployed by some factories. It has been observed that the performance of the automated procedure can be more efficient than the same

problem attempted by humans. Thus, it is evident that a small enhancement or up-gradation in these cutting and packing procedures can result in huge economical saving from the companies perspective. Moreover, it would result in change for better environmental conditions, which is undoubtedly a substantial research direction. Thus, the problem is studied by many researchers, with diverse approaches being taken for solving cutting and packing problems. The research work includes developing approaches that are a variant of existing, or are based on rightly calling of procedures and sometimes only to provide vague insight into how well each of the approaches may fare on any given problem.

Many automated approaches have been proposed based on different techniques like linear and dynamic programming, heuristic methods, local search methods and more recently a metaheuristic methods such as genetic and evolutionary algorithms, artificial neural networks, and Simulated Annealing (SA). However, their performance in terms of computation time depends on the type and size of the problem, which can be improved further. We add to the state of knowledge by the development and evaluation of novel metaheuristic hybridized cutting and packing techniques to obtain optimum results in less computation time. Our approaches achieve a good trade-off between the solution quality and the computational effort made to reduce the size of the search space.

The purpose of this chapter is to briefly introduce both the problems, report the objectives, highlight the contributions, and to present a detailed elaboration of the work reported in thesis to appear in the following chapters. The motivation behind studying the problems and scope are described in 1.1. The objectives are listed in 1.2 and the corresponding contributions made are reported in 1.3. The general preview of the organization of material in this thesis is presented in 1.4.

1.1 Motivation and Scope

These problems are NP-hard in nature, easy to understand, but difficult to find a solution, the CPU computation time increases as the problem size grows and may become intractable even on highly configured systems. Thus, processing of algorithms for these problems are not simple and require a great deal of efforts from the research community. Many problems from cutting and packing areas

are tackled using different techniques, but still there exists a scope for further improvement and few of these problems are still an open challenge. In this thesis, both these problems are studied because of their practical applicability in commercial and industrial real world. These are combinatorial problems in operations research common in business and industry domains. In cutting, a major challenge faced by the manufacturing industry is to cut down the production cost with high material utilization. Other industries like textile, leather, and ship building focus on economic criteria like reducing the pattern generation time. On the other hand, packing is a key component in transportation of goods from one place to another. It has an impact on various other factors like number of vehicles used, protection of goods, etc. It is obvious, therefore, that the improvements in automated cutting and packing procedures can lead to big savings for both, the companies and the environment, and are certainly a worthwhile research direction. All these encourage the need for an automated system that reduces manual generation requiring men and days.

1.1.1 Objectives

The objective of our research work is first to address 1D-CSP and develop new integrated heuristics or improve known heuristics in an attempt to find a solution as quickly as possible with minimal enumeration of cutting patterns. Next, is to develop heuristic as well as a metaheuristic algorithms to find an efficient packing mechanism for the given set of small rectangles. In order to realize these general aims, the following specific objectives are pursued in our research work:

1. Design a new, robust, and flexible to change system that can be used to model different cutting and packing problems occurring in major industries like shipbuilding, textile, wood, plastic, sheet metal, and leather manufacturing.

2. To analyze the well-known methods from literature competent in solving 1D-CSP and to strive towards improving some of these algorithms and propose new ones.

3. To analyze, design and implement a suitable

(a) hybrid integrated approach for the standard 1D-CSP, and

(b) new and improved heuristic algorithm for One Dimension Multiple Cutting Stock Problem (1D-MCSP).

4. To analyze the proposed algorithms in terms of their solution qualities and execution time for

   a) a general multiple length 1D-CSP stock problem, and

   b) the special case with a low demand ratio.

5. To establish a new

   a) rectangle placement approach for the Two Dimensional Strip Packing Problem (2D-SPP), and

   b) coupling of placement strategy with modified standard Genetic Algorithm (GA) for better exploitation and exploration of the search space.

6. To design and implement a suitable and efficient new search methodology using hyper-heuristic that automates the design process.

7. To implement the algorithms in the above stated objectives and to perform comparative analysis drawing conclusions based on their performance

   a) in terms of solution quality and computation time using benchmark datasets, and

   b) in case of a metaheuristic approach to investigate the impact of attributes on the algorithm performance.

1.1.2 Contributions

In this research work, we address a number of issues associated with cutting and packing problems and improve the state of knowledge in the following ways:

1. A model is designed that exploits the bounded knapsack subproblem for Column Generation (CG) such that it stabilizes and accelerates the solution process by reducing the number of iterations.

2. A new fast metaheuristic approach for cutting pattern generation is proposed that takes advantage of problem characteristic along with the existing constructive Integer Linear Programming (ILP) techniques capable of solving large problem instances. The major strength of the model

is integration of ILP with GA techniques to form a single comprehensive procedure. Hybridization of these two techniques is used to combine the relative strengths like local exploitation of the dual problem structure with interesting characteristics by Linear Programming (LP), and global exploitation by GA, which accelerates the column generation.

3. A fast and highly effective heuristic model is proposed for multiple length cutting stock problems in limited quantities that significantly reduce the generated patterns.

4. A novel placement approach is proposed for two dimensional strip packing, which modifies the complex placement strategies by simple ones to improve the overall efficiency of the designed system. The developed algorithm for packing is complete and robust, which allows placement with and without rotation of items.

5. The proposed GA is derived from the standard one with novel fitness function for design evaluation and to enforce proper selective pressure.

6. A new hyper-heuristic search approach using genetic algorithm with an automatic selection model is proposed to select the best Low Level Heuristic (LLH) for 2D-strip packing.

7. A new search methodology is proposed to automate the design process. It is based on a hyper-heuristic search approach that uses grouping technique to recursively solve sub minimization problem. Each iteration, fills some part of the empty region and generates new subsections. Such division of work improves the total computation time. The proposed system design techniques were previously neither applied to cutting nor were drawn from the human analysis and artificial intelligence.

8. We demonstrate the improvement for a number of standard benchmark instances using our proposed approaches.

1.2 Organization of the Thesis

In this thesis, we explore two well known optimization problems from cutting and packing research areas. We present two different approaches to solve each

class of problem. Thus, the thesis is divided into four logical sections and is organized as follows:

Chapter 2 details the background study of the previous work undertaken in the field of cutting and packing problems. The chapter starts with the introduction to various classification systems for cutting and packing problems, where the characterization into groups is governed by the problem features. An overview of recent and existing methodology to solve both classes of problems are presented. Initially, background study is presented for cutting problem with focus on exact, heuristic and metaheuristic methods. Similarly, the work from the literature is reported for packing problem by using exact, heuristic and metaheuristic methods. The chapter also introduces and presents a brief literature survey on hyper-heuristic methods.

Chapter 3 is devoted to standard one dimensional cutting stock problem. The chapter describes the proposed mathematical model that is exploited for column generation. An effective column generation techniques are presented that stabilize and accelerate the solution process. We explain how the acceleration is achieved by imposing penalty function on the fitness value for evolution of better population. The GA capability is enhanced by using dynamic behavior in crossover and mutation operators. The chapter details how the approach dynamism helps to improve the solution convergence rate to a great extend and also controls the random behavior to an acceptable level. The chapter finally reports the experimental work, which evaluates the proposed technique as compared to existing metaheuristic, heuristic and exact approaches.

In chapter 4, we address a classical one dimensional multiple length cutting stock problem. An integrated two-stage approach is presented for the cutting pattern generation and customer demand fulfillment. The solution approach minimizes the total trim loss occurring at all the stages of the technological process meeting customers demand for the finished rolls. The chapter describes the case of multiple length stock materials available in limited quantities. In the first stage, cutting patterns are generated to cut the available stocks and the second stage determines the selection of pattern, number of ordered items to be cut and updating of remaining unused stock and demand for ordered items. We analyze its behavior using various randomly generated and reported instances in the literature.

Chapter 5 addresses the scrap minimization problem for 2D-strip packing. It presents a hybrid approach that combined genetic encoding and evolution scheme with the proposed placement approach. A newly developed fast packing strategy is presented, which improves over the existing ones. This is followed by a brief description of the statistical test that is used to compare the algorithms on mean plot for optimality gap. Computational results are reported on the benchmark datasets against metaheuristic, exact and heuristic algorithms reported in the literature.

In chapter 6, presents another solution technique, which is a novel hyper-heuristic decomposition approach for 2D-strip packing problem. The hyper-heuristic approach, which automates the design process for packing of two dimensional rectangular blocks, is discussed. The chapter adds to the state of knowledge by introducing a new search technique, in which genetic algorithm is coupled with the hyper-heuristic for getting the optimal solution at an acceptable rate. The chapter reports the result for proposed method showing the effectiveness and efficiency of the automated system for determining the placement patterns.

Finally, in chapter 7, conclusions are drawn regarding the research results of each of the addressed problems and the overall work in the thesis. The contributions to the state-of-the-art of the tackled subjects are outlined, and some future work directions are discussed.

Chapter 2

Background and Related Work

C&P is one of the oldest fields of research that falls under the sub-discipline of operations research. The main reason for the continuing efforts in these fields is due to the growing need in various industries for an efficient automated system over the manual solutions. Moreover, these problems have implications on the growing economy of developing as well as developed countries. A cutting problem may be defined as cutting of larger objects into small pieces of specified dimensions to meet the required demand and minimize the associated trim loss. Packing, on the other hand, in the best way can be defined as placing smaller size objects into a larger container by minimizing the in-between vacant space or in other words, maximizing the occupancy of the container. The typologies of C&P problems have similar logical structure, resulting from the geometric dimensions and conditions like small items are laid on larger.

2.1 Classification of C&P Problem

The proper classification of C&P problem is done by considering three known typologies from the literature. The basic framework for cutting and the packing problem involves either to cut a larger object into smaller items as in the cutting problems or to pack/arrange the small items into a larger object together, to form geometric alignment or patterns for packing. These typologies are based on the characterization of objects into different groups based on certain common similarity criteria. The very first classification, given by Dyckhoff [2] in 1990, was a comprehensive typology to integrate various kinds of problems based on the logical structure. The main aim is to unify the different use of notions in

the literature and to concentrate further research on special types of problems. An improved typology by Wäscher *et al.* (2007) [3], where they claimed that some deficiencies exist in the Dyckhoff typology, which cannot be generalized with the recent development in C&P problems. A sub typology for the packing problem is given by Lodi *et al.* (2002) [4]. These topologies are discussed in the subsequent sub-sections.

2.1.1 Dyckhoff Typology

This is the very first classification for the C&P problems into distinctly specified groups. The classification is required as both the problems find a wide applicability in different domains. On one hand, cutting is a major problem in manufacturing industries like paper, wood, steel, textile, metals, etc., while packing finds applications in transportation, dealing with vehicles and goods, pallet loading, bin packing, etc. Packing also has different implications like it can be used in the newspaper industries for article arrangement, in VLSI for chip design layout, scheduling problems in computers, memory allocation during data storage, and many more.

Dyckhoff characterization is based on four major features, which are *dimensionality*, the *kind of assignment*, *the assortment of large objects* and *the assortment of small items* as shown in Table 2.1. The first one, points at the geometrical dimension of the large as well as small objects. The *kind of assignment* considers the case whether the problem involves selection of small objects combined to form a pattern assigned to the large object, or whether all small objects are combined that are then assigned to a proper selection of large objects. The third feature, the *assortment of large objects* is classified as whether there is a single large object, a set of identical large objects, or many large objects of varying dimensions. The last, consider the *assortment of small items* whether there are few or many small objects of different shapes, many small items, but with relatively few different shapes, or whether all items are congruent. The *dimensionality* is characterized by four values: 1 stands for one dimension, 2 for two dimension, 3 for three dimension and N$\gg$3 stands for the N dimension problem. Different notations are used to represent the problem subtype. The *kind of assignment* is represented by using two variables B and V, where B

Table 2.1: Different typology for C&P problems.

Dyckhoff C&P typology		Improved Wäscher C&P typology	
Dimensions			
1	One-dimension	1	One-dimension
2	Two-dimension	2	Two-dimension
3	Three-dimension	3	Three-dimension
N	N-dimension	N	N-dimension
Kind of assignment			
B	All objects and a selection of items	OM	Output value maximization
V	A selection of objects and all items	IM	Input value minimization
Assortment of large objects			
O	One object	O	One object
I	Identical figures	Oa	All fixed dimensions
D	Different figures	Oo	One variable dimension
		Om	More variable dimension
		Sf	Several figures
		Si	Identical figures
		Sw	Weakly heterogeneous assortment
		Ss	Strongly heterogeneous assortment
Assortment of small objects			
F	Few items (of different figures)	IS	Identical small items
M	Many items of many different figures	W	Weakly heterogeneous assortment
R	Many items of relatively few different (incongruent) figures	S	Strongly heterogeneous assortment
C	Congruent figures		

Table 2.2: Improved typology by Wäsher *et al.* [3] for intermediate problem types for output maximization problem (small items to be packed to larger items).

Characteristic of large object		Assortment of the small items		
		Identical	Weakly Heterogeneous	Strongly Heterogeneous
	One large Object	Identical Items Packing Problem (IIPP)	Single Large Object Placement Problem (SLOPP)	Single Knapsack Problem (SKP)
All fixed dimensions	Identical	——-	Multiple Identical Large Object Placement Problem (MILOPP)	Multiple Identical Knapsack Problem (MIKP)
	Heterogeneous	———	Multiple Heterogeneous Large Object Placement Problem (MHLOPP)	Multiple Heterogeneous Knapsack Problem (MHKP)

Table 2.3: Improved typology by Wäscher *et al.* [3] for intermediate problem types for output minimization problem (small items to be packed to larger items).

Characteristic of Large Object		Assortment of the small items	
All fixed dimensions	Identical	Weakly Heterogeneous Single Stock Size Cutting Stock Problem (SSSCSP)	Strongly Heterogeneous Single Bin Size Bin Packing Problem (SBSBPP)
	Weakly Heterogeneous	Multiple Stock Size Cutting Stock Problem (MSSCSP)	Multiple Bin Size Bin Packing Problem (MBSBPP)
		Residual Cutting Stock Problem (RCSP)	Residual Bin Packing Problem (RBPP)
One Large Object Variable Dimension(s)		Open Dimension Problem (ODP)	

indicates the selection of smaller objects that are used to determine packing patterns on all larger objects, while V indicates the selection of all smaller objects assigned to a larger object. For the *assortment of larger object*, three variables are used, O indicates only one object is packed, I stands for the case when multiple identical large objects are to be packed, and D indicates multiple large objects of varying size. Similarly, for *assortment of small items*, four variables are used, namely, F denotes few items of different shapes, M stands for many small items of different shapes, R indicates the case with many items with a few different shapes and C for small items concurrent in nature. Thus, as per this typology, there are 4 x 2 x 3 x 4 = 96 different types of C &P problems. The general way of representation is by four-tuple symbol for respective features λ/ β/ γ/ δ.

2.1.2 Wäscher's Improved Typology for C&P Problems

Wäscher *et al.* [3] report deficiencies in the existing typology for C&P problems. As, the major concern was the notation used to represent 2D-Bin Packing Problem (BPP). Dyckhoff suggested 2/V/D/M to represent the problem. However, many researchers considered 2/V/O/M as a most appropriate notation for this problem class. The inconsistency is due to lack of clear discrimination between the bin packing problem and the strip packing problem. Gradišar *et al.* (2002) [5] highlighted the fact that there can exist another possibility for assortment of large objects, which consists of few groups of identical objects denoted by variable G, thereby raising the possible combination of typology to 128. Thus, the notation removed the ambiguity between the case when items being packed are variable size large object and the case where many large objects can be sorted into different groups containing identical size into which the small items are packed. The problem representation is 1/V/D/R for the first case and 1/V/G/R for the second, which clearly differentiate the two classes of problems. All these existing weaknesses are the basis for refined classification proposed by Wäscher *et al.* [3] shown in Table 2.2 and Table 2.3. These tables put together all the intermediate problems that may arise. They represent the abbreviated form corresponding to the problem for unique problem identification.

2.1.3 Lodi *et al.* Sub Typology

Lodi *et al.* [4] introduced a new sub typology to clearly differentiate between the strip packing problem and the bin packing problem. They used a notation with three fields dP|X|Y, where small d stands for the dimension of the problem with P to represent the type of packing whether bin packing or strip packing. Some more specific notations as per problem like for Continuous Bin Packing (CBP) [6], Level Strip Packing (LSP) and Level Bin Packing (LBP) [7]. X indicates whether the items are oriented (O) or they can be rotated (R) by 90 degrees. Further, a sub-category for this case is proposed by Boschetti *et al.* [8], where few items are oriented and rest is allowed to rotate by 90 degrees denoted by variable M. The last variable Y is used to denote whether it is a guillotine cut (G) or a non-guillotine cut (F). Guillotine cut means only orthogonal cuts are allowed that bisect one component of the sheet, *i.e.*, the cut is parallel to the edge of the container. Non-guillotine refers to the condition where guillotine restriction does not follow and the items can be packed other than parallel or perpendicular to the container surface.

2.2 Cutting Problems Solution Methodology

C&P problems have always been a research area of great interest within the academic community and also received popularity in manufacturing sectors as a mode of increasing productivity and profit at the cost of low waste produced. The areas that are exploited are waste minimization, reducing setup cost, availability of resources and constraint satisfaction. There are various categories in packing, which includes cutting stock problem, knapsack, bin packing, strip packing, nesting problem and loading problem. Cutting stock involves cutting of available raw stock to meet customer demand such that trim loss is minimized. Knapsack problem is best described as consisting of a larger object called the knapsack of limited capacity, and the number of items to be placed. Each item has some fixed size and is associated with certain benefit. The objective is to pack the item to maximize the benefit, but limited by the capacity constraint. The bin packing aim is to pack items into bins, the dimensions are bounded such that the remaining space in use bin is minimized and overall bin required

to pack all items is minimized. All these problems range from single to multi-dimensions. Strip packing is considered in higher dimensions like two (2D) and three (3D). For 2D irregular object packing onto an irregular rectangular sheet like a nest usually in metal cutting industry is referred as Nesting problem. Loading is associated with three and higher dimensions where boxes are placed into the container.

The BPP is similar to CSP as both these problems have essentially the same logical structure. As per W*ä*scher *et al.* typology, they differ on the fact that in a CSP assortment of small items is weakly heterogeneous whereas in BPP it is strongly heterogeneous. In other words, CSP has high demand, *i.e.*, several copies of similar dimension of small items, whereas BPP has low demand, *i.e.*, typically single copy of small item of similar dimension. Due to the similarity in structure, the solution methodology devised for one problem can be applied to other. However, the behavior of these methodologies may depend on the nature of the problem. In this work, we focus on finding the optimal patterns and determining the number of times a pattern is used. The approach is favorable for the CSP.

A number of solution methodologies exist to solve this class of problem, broadly classified into four groups: exact, heuristic, metaheuristic and hybrid approach. A detail literature study of the approaches is discussed in the next subsections.

2.2.1 Exact Approach

In 1960s, paper manufacturing industries were the most prominent manufacturing industries. A linear programming approach to solve 1D-CSP was first presented by Gilmore and Gomory [9] in the year 1961. Approximation algorithms presented are mostly used to obtain the linear programming relaxation but, obtaining the exact solution is time consuming and would result in the enumeration of cutting patterns. To deal with these difficulties, Gilmore and Gomory (1963) [10] relaxed the integral constraint and proposed the Simplex method with a column generation technique to solve the continuous relaxation of the CSP. Although the technique is suitable for solving medium-size and some larger instances of the 1D-CSP but does not consider supply limitation on the

availability of different stock length.

Various LP-based solution presented by researchers are inspired from Gilmore and Gomory like by Hinxman (1980) [11], Sweeney *et al.* (1992) [12], and most of them have proved to be successful for CSP along the years. Minimization of trim loss is the sole focus of LP-based approach. In some compact model methods, the cutting patterns and the number of times to be used are determined simultaneously. However, as observed by Martello and Toth (1987) [13] and also reported by Valerio de Carvalho (2002) [14], the LP relaxation can be very weak, and Branch-and-Bound (B&B) algorithms based on such models can fail to solve the problem to optimality. Holthaus (2002) [15] studied different stock length CSP, where the proposed approach used column generation based decomposition for exactly solving the continuous relaxation and specific methods for separately solving the arising residual problems. Another such method was proposed by Johnston *et al.* (2004) [16], it considered the generated cutting patterns, which are binary in nature. The problem encountered is thousands of binary variables are needed for most of the practical instances that involve dozens or hundreds of item types, and thousands of single items. Thus, it is difficult to solve the compact models. The number of variables is still larger, if the model is extended to deal with the multiple length stock. Both, the branch-cut and the branch-and-bound, are based on LP relaxation, where in branch-cut additional constraints are enforced by introducing cutting planes that tighten the LP relaxation in B&B nodes. Belov *et al.* (2006) [17] used a combination of Branch-and-Price (BP) and branch-and-cut approaches, in which, the LP relaxation at each branch-and-price node is strengthened by mixed-integer cuts also known as Branch-Cut-Price (BCP) approach. Alves *et al.* (2009) [18] proposed a new lower bound for column generation with different strategies to strengthen, which are constraint programming and new families of valid inequalities. In BCP algorithms by Jűnger and Thienel (2000) [19], and Feilleta *et al.* (2010) [20], new variables may be added (priced) to the model in any B&B node. The major concern with the exact approaches is their inability to handle large data instance effectively.

For non-linear real world problems, the LP approach fails to find the optimum solution. It usually formulates the problem as an integer linear programming, and solves the related linear programming relaxation to obtain approx-

imate solutions as by Scheithauer *et al.* (2001) [21], Liang *et al.* (2002) [22], because solving it exactly is time consuming. The approach often generates more items than required because of rounding problems. This may lead to poor material utilization when the average demand for an item type is small and the surplus items are taken as waste.

A non-linear model used for multiple stocks, in which combination of different stocks may lead to better stock utilization. However, establishing optimal solutions is more complicated than when dealing with only one standard stock length. Belov and Scheithauer (2002) [23] proposed a case for multiple lengths stock using an exact approach, which combined cutting plane and column generation approach but failed to find solution for larger instances. Alves *et al.* (2008) [24] proposed new dual-optimal inequalities for multiple length cutting stock problem. The authors explored BCP algorithm to get better solutions without using complex rounding techniques. A few exact approach based papers are reported in literature because of their limitation to solve problem with increasing demand.

2.2.2 Heuristic Approach

The 0-1 knapsack problem was solved by Eilon and Christofides (1971) [25] using heuristic approach and applied the same to similar kind of problems. It was observed that heuristic approach may result in optimal solutions in fairly less computation time. In the same year, Haessler (1971) [26] pointed out the fact that while determining the cutting patterns, the other factors like setup cost can also be considered apart from trim loss for generating the new patterns. This helps in reducing the overall cost associated with cutting of raw materials. The approach used heuristic procedure to solve the developed mathematical model. A recursive search approach for multistage guillotine cutting problem a given by Herz (1972) [27] was although an improvement over existing but, was not efficient to handle the medium size problem. Haessler (1975) [28] formulated the problem and gave a concept of setting up of aspiration level at each iteration. At each iteration, the level was checked, if not reached, the level was lowered for next iteration until solution falls within the level.

Chambers and Dyson (1976) [29] looked at the cutting stock problem from

a different angle. In some ways it was similar to Wolfson (1965) [30], in that they considered the best stock size to hold rather than trying to simply reduce the trim loss using the available stock size. However, Wolfson considered the one dimensional problem, and Chambers and Dyson applied the problem to a two dimensional situation (glass cutting). Both the work used two techniques, namely integer programming and a heuristic method. The use of these methods made real savings when applied to real-life problems.

Heuristics are characterized as complete and sequential based on the process of handling the demand, where the complete considers all demand at once whereas sequential handles them one by one. Due to limitations of the exact approach, in these approaches basically the problem is formulated as an integer linear programming and solves the corresponding related linear programming problem to get the approximate solution. However, rounding off solution often results in poor material utilization with surplus items produced considered as waste.

Another solution category is the heuristic approaches like the Sequential Heuristic Procedure (SHP) is also known as the repeated pattern exhaustion procedure presented Yanasse *et al.* (2006) [31]. It has the flexibility to consider various practical restrictions or objectives and can eliminate rounding problems by working only with integer values. The SHP is greedy and often leads to high trim loss towards the end of the pattern generation process. The hybrid approach usually combines complete and sequential approaches. As, it may solve the linear programming relaxation to obtain the fractional solution (complete), round the solution down, and use a sequential approach to solve the residual problem so that the demands are fulfilled exactly. Trkman *et al.* (2007) [32] presented another variant for CSP with solution to cutting stock in the consecutive time period. The number of published algorithms dealing with pattern reduction is relatively small and briefly discussed by Cui (2012) [33]. The bottleneck to heuristic approaches is that the solutions take infinite time or very long time to compute.

2.2.3 Metaheuristic Approach

The other class of problem is the metaheuristic search that observes the growing influence of the Evolutionary Algorithms (EAs) as by Bačk (1996) [34] and Zhao *et al.* (2012) [35] in solving combinatorial optimization problems. The capability of evolutionary approaches to deal with complex and non-linear problems has increased as they are able to find optimal or near-optimal solution within a reasonable amount of time. Cutting and packing problem has been solved by many researchers using GA techniques. Holland (1975) [36] first proposed the basic principles of GA. Thereafter, a series of literature and reports are made available.

Wang *et al.* (2005) [37] gave a multi-stage genetic algorithm for approximately solving the 1D-CSP with only one stock length. Khalifa *et al.* (2006) [38] presented a GA based model for the 1D-CSP to optimize construction steel bars waste. Jahromi *et al.* (2012) [39] concluded that simulation annealing has better performance over Tabu Search (TS) in finding solutions for 1D-CSP. Ant Colony Optimization (ACO) is also applied to cutting stock problem as by Jap *et al.* (2008) [40] and Yang *et al.* (2009) [41] proposed improvement in techniques for better solutions. However, both the approach were carried for limited instances and compared with evolutionary programming. The major concern with the metaheuristic approaches is that they do not guarantee to find the optimum solution for some class of instances. We have considered adaptive mutation rate for column generation. Here, we briefly discuss literature review for it.

The adaptive mutation rate was introduced by Fogarty (1989) [42]. Bắck [43] applied these concepts and proposed a need for selective pressure. Fogarty and Smith (1996) [44] used cloning mechanism to achieve the selective pressure and the mutation rate, where before applying, it was modified with certain probability. Based on this observation, it is reported that the GA performance over generation grows with an exponential decrease in the mutation rate. The idea is very similar to the simulation annealing concept, setting an analogue between the mutation rate and temperature particularly when both are constant. The optimal rate of mutation performance differs from problem to problem, it varies with evolution time and also depends on the search space being explored. Rather than discrete values for these rates

Stone and Smith (2002) [45] used a continuous variable for the mutation rate under the log-normal adaptation, and showed it to be better over standard scheme. This idea was exploited to devise a deterministic automation approach for mutation rate with optimal value by many researches like Srinivas *et al.* (1994) [46], Hoehn *et al.* (1999) [47], and San (2007) [48]. Srinivas *et al.* described an adaptive GA, where each chromosome has an individual crossover probability P_c and mutation probability P_m, where these rates are adapted with respect to the maximum population and mean fitness. Hoehn *et al.* proposed a modified GA, where the mutation was not limited only for the selection of offspring, but also in the evaluation and selection of individual in the population. This method was adopted by Bingul (2007) [49] for multi-objective problems and reported it to be better than simple GA in dynamic environment suitable for problems like robot navigation, model identification and controller design, etc. Ghosh *et al.* (2003) [50] changed only the mutation probability keeping the crossover rate as constant. The rate was varied between equal intervals. The growing practice of metaheuristic approach to solve complex combinatorial problems getting promise results motivate their use. However, the behavior of these approaches varies from problem to problem.

2.3 Packing Problems Solution Methodology

The solution methodology used to solve the packing problems is basically divided into three groups: exact, heuristic and metaheuristic. Exact approaches are deterministic in nature that guarantees to give an optimal solution in practical time, but fails on a larger problem instances. Heuristics, on the other hand, are a clear and systematic pathway of problem solving, flexible to changes with the ability to give approximate solutions in reasonable computing time. Metaheuristic is a higher level procedure over heuristic for better exploration of the search space to find the best solution (optimal or sub optimal) with less computation effort. This section presents a brief introduction to these methodologies used for solving the strip packing problem.

2.3.1 Exact Approach

The approach uses linear or integer programming along with the column generation stage. In column generation, a number of feasible columns are generated denoting the possible placement into the container formed by the items to be placed. Integer or linear programming is used to select the best pattern that can optimally fulfill the demand. The first approach was given by Gilmore & Gomory (1965) [51], which was an extension of their prior work on one dimensional packing. However, the approach was enumerative in nature in terms of the number of patterns generated, and even becomes intractable for larger instances. To keep a check on the number of patterns, Gilmore & Gomory introduced guillotine constraint. A further reduction was made by two-stage cutting patterns, in which at each level, it was checked to determine whether the placed items are of equal height, if so, then the trimming was not allowed, else third stage cut was allowed with trimming. Lodi *et al.* (2004) [7] proposed a computational bound with a new mathematical model having polynomial constraints and variables showing their LP relaxation dominate the standard area relaxations. Martello *et al.* (2003) [52] developed a new relaxation for B&B method to solve 2D-strip packing and bin packing problems. Heuristic approaches are used either at the root or the descendent node to improve the search. Belov *et al.* (2006) [53] reduced the number of branching states in B&B by the use of equivalence and dominance patterns. Pisinger and Sigurd (2002) [54] proposed branch-and-price approach to solve strip packing and bin packing that uses DantzigWolfe decomposition to obtain lower bounds of very good quality. The LP relaxation of the decomposed problem is solved through delayed column generation. Hifi and Zissimopoulos (2003) [55] improved the solution for the constrained two dimensional cutting problems by Christofides and Whitlock (1977) [56]. Cui (2008) [57] proposed to generate homogenous three-staged cutting patterns for the constrained two dimensional guillotine-cutting problems of rectangles. The approach combined B&B with dynamic programming and used tree search procedure. A recursive approach is given by Cui (2008) [58] combined with B&B, in which, lower and upper bounds are used to prune unpromising branches. Exact approaches like a B&B were used by Martello *et al.* (2003) [52] and Lesh *et al.* (2004) [59]. They modified the approach by adding pruning method with B&B

to solve a small subset of this problem. Kenmochi *et al.* (2009) [60] proposed perfect packing based on the B&B approach having a number of branching and bounding conditions for both classes of with and without rotation strip packing. Boschetti *et al.* (2010) [61] gave a new exact B&B approach with new lower bound and upper bound, where the constructive heuristics are used to set the new upper bound whereas a mathematical model and relaxation are used for setting the lower bound. Arahori *et al.* (2012) [62] presented branch-and-bound method for strip packing with and without rotation is based on the partition into subset represented by g-staircase placements and determines a new lower bound. There are plenty of methods in the literature, which are based on the exact approach in order to solve two dimensional packing. We did not review all the methods as our focus is on metaheuristic approach. Although, all these approaches guarantee to give an optimal solution in practical time, but fails on larger instances.

2.3.2 Heuristic Approach

These algorithms were able to produce an approximation solution at a reasonable computational time. Baker *et al.* (1980) [63] approach Bottom Left heuristic (BL) considered sequential placement of rectangular blocks. Each rectangle was placed at the top right corner of stock sheet, which was later moved down to the possible lowest position and then shifted left most without the overlapping placed rectangles. Lodi *et al.* (1999) [64] gave four heuristic techniques for all four possible cases of two dimensional strip packing, *i.e.*, with and without rotation of 90 degrees, and with and without guillotine cut. They also presented a general framework of tabu search to handle this class of problems. The BL heuristic was improved by Hopper and Turton (2001) [65] as Bottom Left Fill (BLF) heuristics, in which, the in-between gap for the placed rectangles was given preference during the placement of the rectangle. Burke *et al.* [66] used best-fit heuristic that did not require any prior pre-processing of blocks. To place any block into the container, the lowest available space was searched and an appropriate rectangle was selected from the unplaced in order to maximum fit the vacant space. The approach was effective in comparison to existing. Work by Lesh *et al.* (2005) [67] presented an interactive user interface that allows

users to change the solution. Ntene and Vuuren (2009) [68] gave Size Alternating Stack (SAS) algorithm on observation that FFDH algorithm performs poorly when the large difference exist in heights of rectangles to be packed in same level. They performed placement by division of rectangles into two subset narrow and wide based on height to width relation. Ortmann *et al.* (2010) [69] discuss a series of heuristic approaches like SASm algorithm, the stacking procedure for narrow items for floor-packed items, stack ceiling (SC) and Stack Ceiling with Re-sorting (SCR) algorithms, and made use of the idea of packing onto ceilings as in the Floor Ceiling (FC) algorithms, and the stacking of items as in the SAS algorithms. Imahori and Yagiura (2010) [70] considered both horizontal and vertical gaps for the best placement location. They used efficient data structures to maintain the current skyline, store remaining rectangles to be packed and efficiently search for the best-fit rectangle at each step. Stephen and Zhang (2011) [71] proposed Fast layer-based Heuristic (FH) with the strategy of stacking rectangles. The selection of rectangle to be packed was determined by the fitness value rule. It first places one reference rectangle and stacks some rectangles to obtain the reference line. It then finds one lowest available rectangle space under the reference line, computes the fitness value of each unplaced rectangle, and selects one rectangle with the maximum fitness value to place. At last, greedy search is used to improve the placing result. Moreover, when the problem becomes complex in terms of number of rectangles to be packed, it increases the execution time for several algorithms and generally rises to find an appropriate solution. Wei *et al.* (2011) [72] proposed Iterative Doubling Binary Search (IDBS), which, when combined with tabu search (TS) outperformed many of the approaches from the literature. The tabu search basic technique is to apply a best improvement local search by using short term memory to escape from local minima and repetitive cycles [73]. The data structure tabu list is used to keep track of most recently visited solutions and forbids moves toward them. Thus, neighbourhood search is restricted to solution not present in tabu list. At each iteration best solution from the feasible set becomes the new current solution. The tabu list is updated to include the new solution. Due to this dynamic restriction on the allowed solution it is also termed as dynamic neighborhood search technique. The algorithm stops when a termination condition is met [74].

A number of new and improved level based heuristics are reported in the

literature. Considering both horizontal and vertical placement Asik *et al.* (2009) [75] improved the best fit heuristic by considering the Bi-directional Best Fit (BBF) placement. This work was modified and improved by Özcan *et al.* (2013) [76] by Best Fit based Bidirectional heuristic (BBFM) taking pair of rectangles to be placed rather than single placement. The heuristic approaches exploit the feature of the problem to get optimal solution in reasonable time. However, the approach does not guarantee to get optimal for all cases. Yang *et al.* (2013) [77] recently presented a Simple Randomized Algorithm (SRA) based on least waste strategy with simplified parameter adaptation. It can quickly find a solution, but the quality of the solution is worth improving.

2.3.3 Metaheuristic Approach

Meta in the Greek translation stands for *beyond*, operating at a level higher than heuristics. They may be viewed as upper level general methodologies that can be used as a guiding strategy in designing underlying heuristics for efficient exploration of the search space. Metaheuristics are general-purpose algorithms that can be applied to solve almost any optimization problem. In packing, most of the widely used metaheuristic algorithms are nature inspired, which may be genetic algorithm, ant colony optimization, simulated annealing, etc. The metaheuristic approaches used in packing are briefly described here. These include genetic algorithms by Jakobs (1996) [78], Ramesh Babu & Ramesh Babu (1999) [79], Dowsland (2006) [80] and Gonalves (2007) [81], simulated annealing algorithms by Dagli and Hajakbari (1990) [82], Lai and Chan (1996) [83] and Martins et. al (2010) [84], neural network algorithms by Dagli and Poshyanonda (1997) [85], and some hybrid metaheuristic algorithms by Hopper and Turton (2001) [65], Hifi and M′Hallah (2003) [55] and Bortfeldt (2006) [86]. The neural network based approach given by Dagli *et al.* (1990) [82] combines the hybrid approach, but is not preferred due to high computation time. Martins *et al.* (2010) [84] used crystallization heuristic and performed a limited depth binary search to find a scale factor that when applied to the polygon, would allow it to fit in the container. Simulation annealing methods are often comparable with heuristics and use an additional approach like B&B. The main concept of GA for the strip packing problem is the evolution of rectangle packing sequence as

presented in the approaches by Jakobs (1996) [78] and Liu and Teng (1999) [87]. They combined GA with a heuristic approach like BL. For orthogonal packing by Ramesh Babu & Ramesh Babu [79] proposed GA in combination with deterministic heuristic. The BL heuristic was improved by Hopper and Turton (2001) [65] Bottom Left Fill (BLF) heuristic in which the gap that existed in between the already placed rectangle was given preference while the placement of the new rectangle.

Hopper and Turton (2001) investigated an amount of metaheuristics to create a placement order, including simulated annealing and GAs and conjunct them with certain of heuristic strategies such as BL and BLF. Withal, those non-deterministic algorithms are time consuming and small application for the problems with an immense enumerate of rectangles. The GA approach considering strip packing as the permutation problem was solved by Yeung *et al.* (2003) [88] named, as Lowest-Fit-Left-Aligned (LFLA) heuristic approach. Bortfeldt (2006) [86], gave a sequence algorithm titled SPGAL that did not use any encoding and worked directly on the resolution layouts. Goncalves (2007) [89] proposed Hybrid Genetic Algorithm(HGA) for rectangle placement using random keys. Burke, Kendall, and Whitwell (2009) [90] enhanced the Best Fit (BF) rule (by Burke *et al.* (2004) [91]) with the crossbred simulated annealing and BLF rule of Hopper and Turton (2001) [65]. Alvarez-Valdes *et al.* (2009) [92] planned an activated Greedy Randomized Adaptive Search Procedure (GRASP). Influence is presented by learning few information to fix the suitable parameter settings for the strip packing difficulty. GRASP improved performance for small size instances (Alvarez-Valdes, Parreo & Tamarit (2008) [93]). Belov, Scheithauer, and Mukhacheva (2008) [94] presented two unvaried heuristics based on one dimensional heuristics, which are SVC (SubKP) (stands for Sequential Value Correction (change knapsack job)) and BS (BLR) (stands for Bubble Search (bottom-left-right)). It is found that SVC is a coercive rule and performs well in most instances. Later, the Least Waste First, heuristic (LWF) was presented by evaluating positions, and is improved by union simulated annealing algorithm by (Wei, Zhang & Chen (2009) [95]. LWF performs outmatch for rectangle packing difficulty. SPGAL, Discernment, BF + SA, SVC and LWF are supported on formulating strategies and can produce real promising solutions within a sensible instance, but they are more involved,

especially the execution of these algorithms significantly depends on the settings of the parameters. Moreover, several algorithms generally order untold quantify to obtain a preferable answer when the problem size grows. Stephen *et al.* (2010) [96] proposed Hybrid Simulation Annealing (HSA) algorithm that use fitness to evaluate and introduce to jump out of the local optimal trap using greedy strategy. Leung *et al.* [97] proposed Intelligent Search Algorithm (ISA) a two stage approach that used scoring mechanism along with local search and simulated annealing. The performance of the approach was reported better than GRASP approach. But by the use of metaheuristic approach like simulated annealing the performance is dependent on parameter settings. In general, the results obtained shows that these improved approaches are competitive with other approaches reported in the literature.

2.3.4 Hyper-heuristic Approach

The concept of using hyper-heuristic technique is not recent. The approach was used in many problems since 1960s. A prior mention of the term is found in a technical report by Denzinger *et al.* (1996) [98], but used in the context where range of artificial intelligence algorithm was used for theorem proving. However, it was first reported in a peer reviewed conference paper in 2001 by Cowling *et al.* (2000) [99]. A state of the art survey of hyper-heuristic is presented by Burke *et al.* (2013) [100]. Burke in his work, states the crucial work that leads to hyper-heuristic method comes from the contribution by Fisher and Thompson (1963) [101] and Crowston *et al.* (1963) [102]. In production scheduling based problem, the idea was to combine scheduling rules for better performance in comparison to applying single rule. In 1990s, the similar work was discussed by Storer *et al.* (1992, 1995) [103], [104] work on job shop scheduling problem. The scheduling was encoded with the objective of finding the best combination of problem specific heuristic. Another approach to search heuristic sequence using genetic algorithm for open shop scheduling is discussed by Fang *et al.* (1993, 1994) [105], [106]. Drecheler (1996) [107] presented a solution to Ordered Binary Decision Diagram (OBDD) by using evolutionary algorithms and developing strategies to the problem that learns from good heuristics. The main idea is to develop a system, which automates the task of exploiting problem structure

by either evolving or intelligently selecting the low level heuristic. Burke *et al.* (2010) [108] employed a genetic programming approach for strip packing. The approach is inspired by best-fit and use different placement schemes to evolve. Our work presented differs from the existing genetic programming approach for strip packing as the latter uses evolving criteria for low level heuristic and report results on limited data instances. The main motivation behind our work is same as automation of the process, but we have exploited the features of genetic algorithm to achieve the same.

2.4 Chapter Summary

The aim of this chapter has been introduced the notion of cutting and packing problems. It gives a short description of the existing typologies for cutting and packing problems. The chapter presents a description of almost all classes of methods like heuristic, exact and metaheuristic developed over the last few decades for C&P problems. The first discussion is carried out for 1D-CSP and later followed for 2D-SPP. It also discusses hyper-heuristics, which have emerged as a new hybrid approach to solve problems with higher generality in C&P and various other fields.

Chapter 3

Column Generation and 1D-Cutting Stock Problem

3.1 Introduction

Column generation is an efficient solution technique, which is applied to various combinatorial optimization problems. These large scale optimization problems have two major concerns. First is the volume of data to handle, as it has an impact on the computing capabilities. Second is the available algorithms as the solution approach for the subproblem is based on enumeration, which is a computation bottleneck for the model as the number of columns become exponential. As the number of iterations grow in the column generation process, the rate of change of the solution to find optimal is low, and the algorithm converges at a very slow rate. To overcome this unwanted behavior, many approaches are reported in the literature, which are capable of accelerating the column generation process. However, most of them were studied exclusive within modified contexts for the linear relaxations of the models. The convergence rate is a crucial aspect of column generation it is defined as the speed at which the master problem converge to optimal solution. We analyze and propose a different technique based on metaheuristic search, to improve the convergence rate of a column generation algorithm for the cutting problem. The problem of cutting stock is a waste minimization problem resulting from the cutting operations to meet the demand for different order of rolls of specified widths. The rolls are cut from the available raw stock of fixed or varying width. The

former is called standard and later class of problem is termed as multiple length cutting stock for one dimension (1D-MCSP).

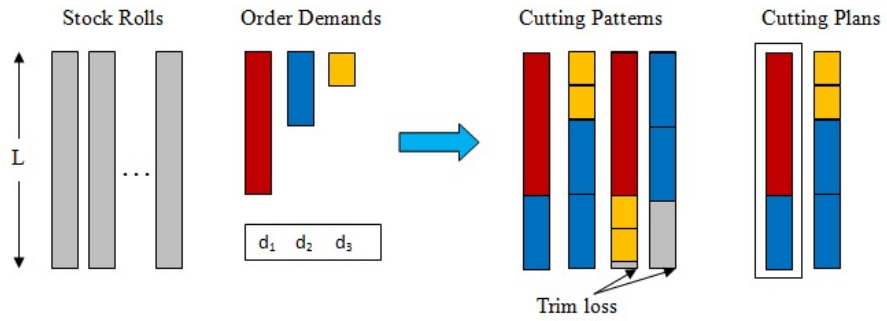


Figure 3.1: One dimensional cutting stock problem with standard stock type.

The one dimensional cutting stock problem (1D−CSP, Figure 3.1) is defined by the following data: m, L, w= (w_1,w_2, ...,w_m), d=(d_1,d_2, ..., d_m), where L stands for the length of the available raw stock in abundance, m denotes number of order demand to be fulfilled, w_i is the stock length for the i^{th} order demand, and d_i is the corresponding order demand quantity where i=1, ..., m. The problem is solved by finding the optimal *cutting plans* that result in minimum waste per utilized stock. The *plan* refers to number of cut for a particular demand from the selected stock length. The selection of *cutting plans* also has as impact on various other monitoring parameters from the real world scenarios like associated setup cost (determined by the number of different cutting patterns), open stack issue etc. Here, the objective is to minimize the trim loss or in other words number of used raw stocks.

3.2 Terminology

3.2.1 Column Generation

The original cost minimization problem is split into two subproblems: a Restricted Master Problem (RMP) and a subproblem also known as the pricing problem. The CG classical process iterates between a RMP and the pricing problem. The major issue is the RMP size, usually it is less as compared to master problem by leaving out some columns from the master form of the Linear Programming (LP). The motivation behind the reduction is the fact that most of the associated values of the variables turn to zero in the optimal solu-

tion. In the column generation process, initially, the master problem is solved to find the values of the dual variables associated with the constraints. In turn the pricing problem, uses these dual values to update the cost coefficients and generates new incoming columns. The subproblem is subjected to several constraints under which the objective function is minimized. These constraints define the relations among the coefficients of a column. The task is to identify the best non-basic column whose addition in the RMP would improve the objective function value. The RMP is resolved with added variable and the associated column. If no such columns are found, it indicates that the optimal solution of the LP relaxation of the problem is obtained. The columns that make the RMP are called the *cutting plans*. The major concern of CG is to limit the number of columns explicitly included in the Integer Linear Programming (ILP) problem. However, the count of *cutting plans* cannot exceed the number of stock lengths and is usually equivalent to the number of order demand. The performance of the computation methods depends on the number of constraints used to represent the linear problem. The growth is quadratic in terms of constraints with complexity for simplex method as O (m^2n), where m and n are the number of constraints and decision variables respectively [109].

3.2.2 Genetic Algorithm

Genetic Algorithms (GA) are based on analogies to natural evolution process like selection, mutation, crossover and reproduction. It is a metaheuristic search approach to efficiently solve optimization and search problems. The steps involved are briefed as follows:

1. Representation: It presumes that the potential solution of a problem is an individual and can be represented by a set of parameters. These parameters are regarded as the genes of a chromosome and can be structured by a string of values in binary form, as alphabet, real, decimal number, etc. At the initial level, the process of initialization uses these representations to randomly generate a set of initial population.

2. Selection: GA use inheritance at each iteration, where the best individual is selected for next population. The selection of individuals to be part of the next evolution process is determined by the fitness value, higher the

value better is the chance of selection. In this process, for every generation the fitness value of each solution is evaluated. These fitness values are close to the problem objective function, which may be either minimization or maximization. It reflects the credit of the chromosome to solve the problem.

3. Recombination and Mutation: This step is carried to generate the individual of the next generation. The offspring is created by recombination (crossover) between the parents with certain probability termed as crossover probability. The operation is followed by mutation, its role is to that modifies sub-part of the representation. It is applied with a very small rate called the mutation probability. Both these operators are crucial in searching the variable space.

Algorithm 1 shows the standard GA process, which is repeated until the termination condition is reached. The termination condition is problem dependent where it can be a maximum number of allowed iterations or some other criteria like number of fitter individuals exceed certain threshold etc.

Algorithm 1 Standard Genetic Algorithm

```
Initialization
while (not termination condition reached) do
    Selection and Reproduction
    Crossover
    Mutation
    Evaluation
end while
```

3.3 The Mathematical Model

The 1D−CSP model is described by Gilmore & Gomory [110]. Let N be the upper bound on the number of raw stocks required to meet the demand in the case of optimal solutions. The objective is to minimize the amount of raw stock utilized to fulfill all demand. Variable c_{ij} is the column value indicating the number of pieces of demand i in the jth available stock length. A decision variable y_i indicates y_i=1 when stock is used and 0 otherwise.

$$\min \sum_{j=1}^{N} y_j \tag{3.1}$$

$$s.t. \sum_{i=1}^{m} w_i c_{ij} \geq d \tag{3.2}$$

$$c_{ij} \in Z_+, y_i \in \{0,1\} \tag{3.3}$$

Gilmore and Gomory solved the above model by splitting it into two phases. The first phase deals with the LP relaxation of 1D−CSP integer programming model. The second phase is termed as the column generation phase with the major task of generating columns that price out some columns from the master problem at every base step. To generate columns at each iteration, a subproblem namely maximization (integer knapsack) is solved.

3.4 Algorithm Formulation

Here, the column generation model adopted is mapped arbitrarily to the Bounded Knapsack Problem (BKP). Given N number of available stocks, c_{ij} is the number of cut for item i in roll j, w_j denotes the jth available stock used, and λ_i is the dual associated with the i^{th} constraint.

The BKP is formulated as:

$$\max \sum_{i} \lambda_i \;\; c_{ij}, j = 1, \ldots, K \tag{3.4}$$

$$s.t. \sum_{i=1}^{N} w_i c_{ij} \leq L \tag{3.5}$$

$$0 \leq c_{ij} \leq UB_j \tag{3.6}$$

Here, the UB stands for upper bound, which is computed as

$$UB = \lfloor \frac{L}{w_j} \rfloor, \;\; j = 1, \ldots, N \tag{3.7}$$

Equation 3.5 enforces the knapsack constraint restrict each vector (c_{1j}, ..., c_{mj}), j=1, ..., N to fall inside the knapsack polytope, which is the set of linear combination of all feasible cutting patterns. Another simple constraint is:

$$c_{ij} \leq d_i, i = 1, \ldots, m \tag{3.8}$$

as it reduces the search space of the continuous relaxation for instances where the demands d are small.

The common approach to get an optimal integer solution is to solve the LP relaxation, in which column generation is integrated with BCP. The major concern with this approach is the high computational time. In the proposed approach, we have used the metaheuristic technique for column generation. The GA routine is invoked to find the next column to be added to the RMP for getting optimal solution. GA solution proceeds with generating the initial population based on the order length and order type. A binary representation is used to represent each chromosome. Each chromosome is evaluated to determine the fitness value. These values are subject to penalty depending on whether or not constraints are violated. The recombination and mutation help to generate better individuals. The process terminates on reaching the termination condition. The evolutionary process is interrupted when the average fitness in the current population exceeds the threshold, *i.e.*, near to the optimal. The threshold in this case is set in the range of $1-5$ %, which reflects the permissible scrap produce from a single stock. Some fraction of the best individuals are selected and checked, whether they already exist in the RMP if so, the repetitive ones are removed. The best amongst them, which is the one with maximum reduced cost is introduced to RMP and the process continues till the integral solution is obtained.

3.5 Minimize Number of Different Patterns

The master problem is a linear formulation and the subproblem is mapped BKP. The BKP is a generalization of the 0-1 knapsack problem where a bounded amount of each item type is available. This is a first attempt to solve the subproblem using GA for 1D−CSP. The algorithm operational mechanics along with the sub-routine are discussed in Figure 3.2

The effectiveness of GA to generate optimal solution is governed by various parameters like representation, fitness function used for evaluation, selection mechanism and operators used. The proposed approach improves the convergence rate by considering three parameters: firstly, a simple preprocessing of individuals subjected to the constraint in Equation 3.2, if violated the indi-

Algorithm 2 Proposed Algorithm

Input: m Customer demand (w_i), K Available standard stock length (L)
Output: Cutting plans, optimal solution
Begin
    Read input from file
    Formulate master problem subjected to constraints
    Create the input matrix
while (not integral solution) **do**
        Solve the master problem using ILP solver
    if Integral Solution **then**
        Output cutting plans and results
        Exit
    else
        Solve pricing problem for column generation

▷ Invoke GA sub-routine

        Determine the chromosome size
        for i=1:m **do**
            numbits=numbits + $\lfloor \frac{L}{w_j} \rfloor$
        end for
        for i **do**=1:numpopulation
            Perform random initialization
        end for
        while (not Termination) **do**
            Compute fitness of each individual using Equation 3.9
            Check for constraint violation
            Penalize the weaker individuals using Equation 3.11
            Determine the crossover rate using Equation 3.12
            Determine the mutation rate
            Perform crossover and mutation as per parent selection mechanism
            p=p+1

▷ Next generation

        end while
    end if
    Compute 1 - $\sum_{i=1}^{m} \lambda_i^j \, c_{ij}$
    if (negative reduced column) **then**
        Add column to the master problem
    else
        No further column to add

▷ Pricing terminates

    end if
end while

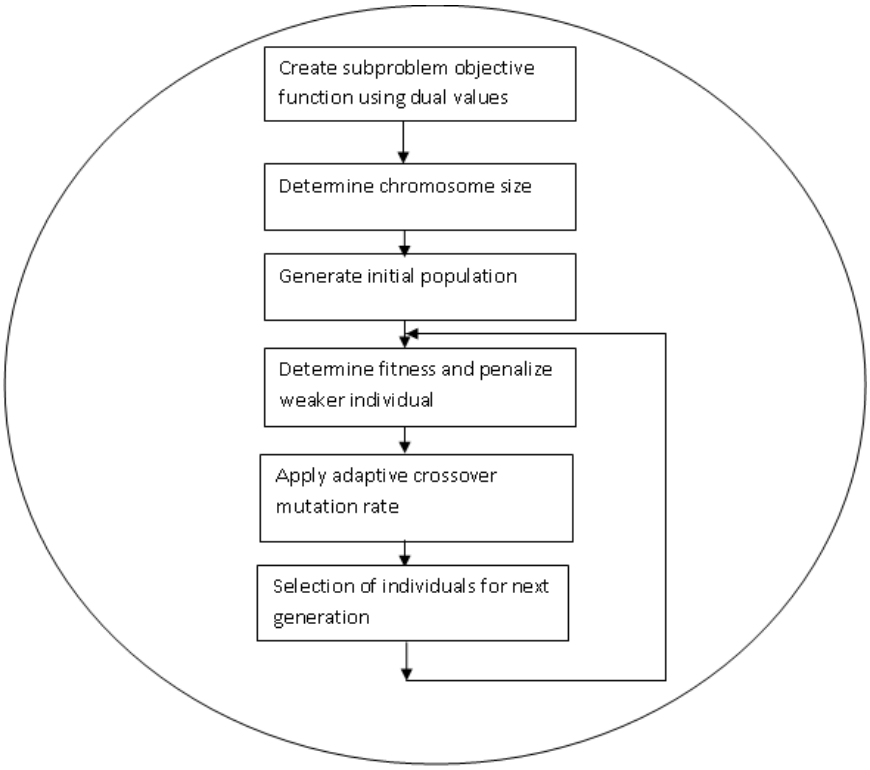


Figure 3.2: Column generation GA sub-routines.

viduals are marked infeasible and eliminated from further reproduction process. Secondly, a penalty function is used to monitor the fitness evaluation of chromosome. The penalty function ensures that the bad fitness value of the chromosome is reduced such that it is not selected by the selection mechanism. Thirdly, a novel problem specific adaptive crossover and mutation rate are designed to improve the convergence rate. These selections, consider a set of better individuals, whereas the least fit individuals are penalized or simply eliminated. The best ones, then undergo recombination under the action of the crossover and mutation operators. The crossover and mutation rates may vary from one generation to another as the population evolved is based on the problem. The observed feature motivated researcher to introduce dynamism in selecting the crossover and mutation rates rather than fixed rates over iterations. Each component of GA is discussed in the following subsections.

3.5.1 Encoding Scheme: Chromosome Representation

GA are attractive as they can handle problem constraints by simply embedding them into chromosome coding. Since it is a technique independent of the error surface, it is ready to solve multi-modal, non-differentiable, noncontiguous, or

even NP-complete problems. The genetic representation may differ considerably from the natural form of parameters of the solutions. In the evolutionary system, a phenotype in the original problem context refers to the object that form possible solution. In order to solve this optimization problem using GA, it is required to code the problem in form of genotype. The genotype for this problem is represented by n-order demand or gene, where the size of the gene is determined by the demand values. Often when the problem is coded as genes, the actual value corresponding to alleles are obtained by converting back to the respective base number. Binary coding is the most common form of coding that poses a cardinality of 0 or 1. The other form of coding include real, array, alphabetic, tree, etc. GA uses many forms of representations. However, the real-coded and binary-encoded strings for representing the solutions have dominated the GA research as they provide the maximum number of schemata and are amenable to simple implementation. Although, real-coded are simple, but there exist some known problems. Theoretical analysis, presented by Holland [36] and by Goldberg [111] suggests the fact that the binary encoding with the low cardinality allows quick and effective convergence to a good solution. Even, Hollands Schema Theorem, which answers how and why binary-encoding enables GAs convergence to the best solutions, but fails to explain the same for real-coded. To overcome this limitation, Goldberg [112] presented a schema theorem, in which the large cardinality is manipulated into virtual small cardinality. However, the theory also states that for certain type of problem, it can prevent from finding the global optima. Thus, the BKP system is represented using binary coding with each parameter coded as a 16-bit binary number because of the simple and traceable nature. Further, during evaluation a noticeable improvement in the system performance was noticed when binary coding was compared to real-coded. Although, it was observed that the number of generations required to converge were more, but the evolve design was much fitter. As mentioned above, each genotype for an individual design contains n- genes.

The size of the chromosome is determined by the number of bits used for the representation of order demand length with respect to the available length. The representation is shown with help of Figure 3.3. In this work, we have designed an intelligent encoder and decoder. The encoder takes the input from the file to read the input parameter like customer demand order, number of demand and

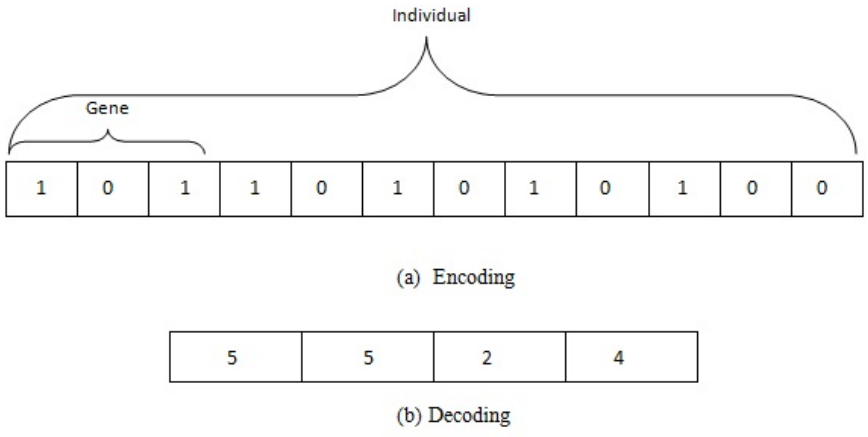


Figure 3.3: Chromosome representation.

available stock length to generate the random initial population.

3.5.2 Fitness Function

The objective function of a problem is the main part providing the mechanism for evaluating the status of each chromosome. Fitness value is a quality measure of an individual. It is an important link between the GA and the system as it determines the selection in the next step. They may be a simple computing or may be determined by some complex experimentations. In many cases the objective function and the fitness function are same. However, we evaluate the individual fitness using Equation 3.9. The population checks for the termination condition, if met, then the process is discontinued, else it will proceed to evaluate each individual and assign a fitness value to the chromosome. This is an optimization problem the associated constraints can be handled in this step by enforcing the penalty function to penalize the weaker individual. It degrades the value of the weaker individual, thus, preventing them from greater influence on the final solution.

The individual obtained after this rigorous searching would have the higher fitness value.

$$\textit{fitness chromosome} = \sum_{i=1}^{N} w_i \;\; c_{int}^{i} \tag{3.9}$$

$$\textit{where } c_{int}^{i} = \sum_{j=0}^{k} 2^j gene_k \tag{3.10}$$

Here, fitness is the summation of the product of a factor (c^i_{int}) and corresponding width for i^{th}order demand, c^i_{int} refers to the integer value corresponding to the k number of bits representing the gene. It indicates the number of pieces for each order demand being cut. The higher the fitness value, the less is the waste produced. These better individuals are the new column generation patterns used for solving the master problem. The objective function is to solve the RMP and obtain the integral solution. If no integer solution is found, then it determines the dual value (λ) associated with the master problem constraints. Further, in order to facilitate the selection mechanism to select good chromosome, we penalize the fitness value of bad chromosome by applying the penalty function. Moreover, it is observed that the rate of change in the fitness function has a vital effect on the GA performance.

3.5.3 Penalty Function

The efficiency of the constraint optimization problem is improved by adding penalty functions. Penalty functions are quite useful techniques when used with the GA and constraints are the part of almost all optimization problems. Thus, applying the penalty function helps in improving the solution convergence and produces a feasible solution as a result of evolutionary search subjected to satisfying all constraints. To solve this optimization problem, the penalty to the fitness function is a violation of constraints by the individual. The value keeps on varying to avoid getting caught in local optima and try to find a global optimum with each evolutionary stage.

Similar to work reported by Kazarlis *et al.* [113] we have used Square function which is given as:

$$P(f) = \frac{f}{(G)^2} \tag{3.11}$$

where f is the fitness value of chromosome and G is the current generation number.

It was observed that the search time is cut down by the use of penalty function. Since, it avoids the large region exploration and prevents it from stalling outside the feasible region.

3.5.4 Selection Mechanism

The selection process is an important phase of genetic algorithm, which deals with the selection of a better individual from the given population. The objective is based on survival of fittest concept, *i.e.*, the fittest individual should have a better chance for survival in comparison to others. Thus, the weaker individuals are not considered without a chance. The generation of good offspring is dependent on an efficient mechanism for selecting a better parent. The selection mechanism used in the proposed approach is a variant of elitism discussed by Goncalves *et al.* [114], to preserve the best individual. In this process, the entire population is divided into two sets, namely elite and non-elite population, where the size of the elite is less than the non-elite. Elite population consists of the individual, which have the highest fitness value. The main objective behind the creation of this group is that the features of a good chromosome, which stand a good chance to solve the optimization problem, are copied as it is from one generation to another during the selection process. The group size is problem depended, however, here the size is kept reasonable, which is out of the fittest individual 1/8 of the total population are selected to be the part of the next generation. Selection of only a small fraction ensures that our model can tackle the worst case scenario where the individual, selected as elite may not be optimal or sub-optimal. As this fraction of the population are directly involved in the evolution of next generation, a check is enforced by not transferring more individuals as the elite population. The remaining individuals form the group of non-elite population. Elitism selection mechanism improves the performance in many cases in terms of optimality and convergence of GAs. Controlling the degree of elitism is a crucial factor, because high selection pressure may lead to premature convergence.

3.5.5 Adaptive Crossover Mutation Operator

Controlling the selection of GA parameters are very crucial as wrong or sub optimal parameters can lead the search algorithm to local minima. Thus, researchers are interested in developing the new GA based optimization techniques. The main idea is to develop a genetic approach that automates itself to adjust the parameter settings based on the performance to evolve better individual with the

higher fitness value. The crossover operation is the soul of GA. The operation produces a random exchange of genetic features from parents to the offspring during the evolution process. Hence, it proceeds with the possibility that good individuals, can generate better ones. Crossover between parents takes place with any probability P_c (crossover probability or crossover rate).The crossover operator results in exchange of features between the mating parents. The various classifications of crossover techniques include the single-point, the two-point, and the uniform types, which are selected based on the problem type. In our approach the designed adaptive crossover probability is based on the fitness evaluation of the generation. The mutation operation involves some changes in the values of individual with probability P_m (mutation probability). A mutation in GAs has an important role as it avoids the premature convergence of the GA to sub-optimal or local optima and tries to restore lost or unexplored genetic material into the population. Here, we have considered two-point crossover between the two individuals, one from elite group and other from a non - elite group of population. The flip bit operator is used for mutation. In this technique, bits are flipped randomly. The crossover and mutation rates are modified to increase the convergence rate of the solution. The rate, at which the solution is subjected to crossover, is determined by the crossover probability P_c calculated using Equation 3.12. Thus, the rate of introduction of the new solution increases with the P_c rate. But as P_c increases, the solutions may be disrupted faster than the selection mechanism can exploit them. The best selected values of P_c are in the range of 0.4-0.8. Nevertheless, the choice of P_c is critical to GA performance and has been emphasized in DeJongs inceptional work [115]. In the proposed approach, we have preferred dynamic rate over classical GA approach, where the value of the P_c and the P_m are kept constant for all the generations during the evolution. As a result, solutions with high fitness values as well as solutions with low fitness values are subjected to the same levels of mutation and crossover. Due to this constant value, when a population converges to a globally optimal solution (or even a locally optimal solution), then the P_c and the P_m increase and may cause the disruption of the near-optimal solutions. Thus, the population may converge to the global optimal at a very low rate. However, it is observed that the values of these parameters must depend on the problem type and should be changed to the fitted values for all the solutions of

the population. Thus, the changes made not only prevent the GA from getting stuck in a local optimum, but also the performance of the GA (in terms of number of generations required for the convergence) will certainly improve. Here, we have varied P_c between 0.4 (K_1) to 0.8 (K_2), the formulation is given as

$$P_c = ((\ f_{max} - \ f_{min})/\ f_{max})(K_2 - K_1) + K_1 \tag{3.12}$$

where f_{max} and f_{min} are the maximum and minimum fitness values for the generation respectively. The values for these constants are determined by carrying out statistical analysis using regression analysis for different values of K_1 and K_2 and the corresponding value of P_c using statistics open for all (SOFA) software. The result indicates for the chosen probabilities for changing the crossover operator, there was no significant difference at 95% confidence interval in the performance of the algorithm. The behavior of the algorithm was robust for such a selection.

Mutation is a secondary operator to restore genetic material. Study by Smith *et al.* [116] on using adaptive mutation rate indicates that different mutation rate used is not much important in comparison to the mutation values used. A mutation must be selected such that it considers both high and low values, which prevent the GA from being struck into local minima. We have considered a set of discrete mutation rates to solve this optimization problem. The selection of high or low mutation rate is based on the threshold parameter, where threshold is based on the fitness value of the individual. If the number of individuals with high fitness value is less than 35%, the mutation is adapted to high rate. Similarly, if the number of individuals with high fitness value is higher than the 70 %, we choose low mutation rate. In average cases, we go with the normal mutation evenly distributed between 0.5/length and 2.0/length, where length denotes bit string length. The convergence of the GA is improved by having a higher fitness solution, on the other hand, the local optimum is avoided by the population with a low fitness solution. The impact of such change is discussed next, *i.e.*, in the computed result.

3.6 Computational Results

In order to evaluate the performance of the proposed GA based column generation technique, the data instances have been collected from the literature and some are randomly generated using CUTGEN1 software Gau and Wäscher [117]. The number of finished product types determines the size of the problem instance, which has been varied to tackle different kind of problems. The approach modeled the demand lengths (w_i) as the uniformly distributed integer random variables, which are of particularly smaller length in comparison to available stock. All experiments were conducted on an Intel Core2Duo 2.20 GHz processor. The algorithm is coded in Matlab and ILP solver is used for computation. This section highlights the impact of using, adaptive crossover and mutation. A comparison of computational result is presented along with the LP-based approach and other metaheuristic approaches.

Table 3.1: Trim loss and number of columns generated for LP-based and proposed approach.

Sr. No.	Dataset	LP Approach		Proposed approach	
		Number of columns	Trim loss %	Number of columns	Trim loss%
1.	D1	20	9.66	10	2.670
2.	D2	91	2.894	34	0.981
3.	D3	314	3.787	179	2.320
4.	D4	1031	1.562	259	0.621
5.	D5	29	1.600	32	1.20
6.	D6	115	5.977	52	3.25
7.	D7	1083	2.942	386	1.341
8.	D8	1099	1.849	412	0.741
9.	D9	2013	3.971	519	1.830
10.	D10	2349	1.798	533	0.651

3.6.1 Dataset Description

Experimentation is carried out on 10 different datasets. The size of the order demand varies from minimum 20 to the maximum 600. Most of the instances used for testing the approach, are taken from the literature, where some are random and others are real-time data from the manufacturing industries. The approach is also verified for uneven demand quantity and for the cases where

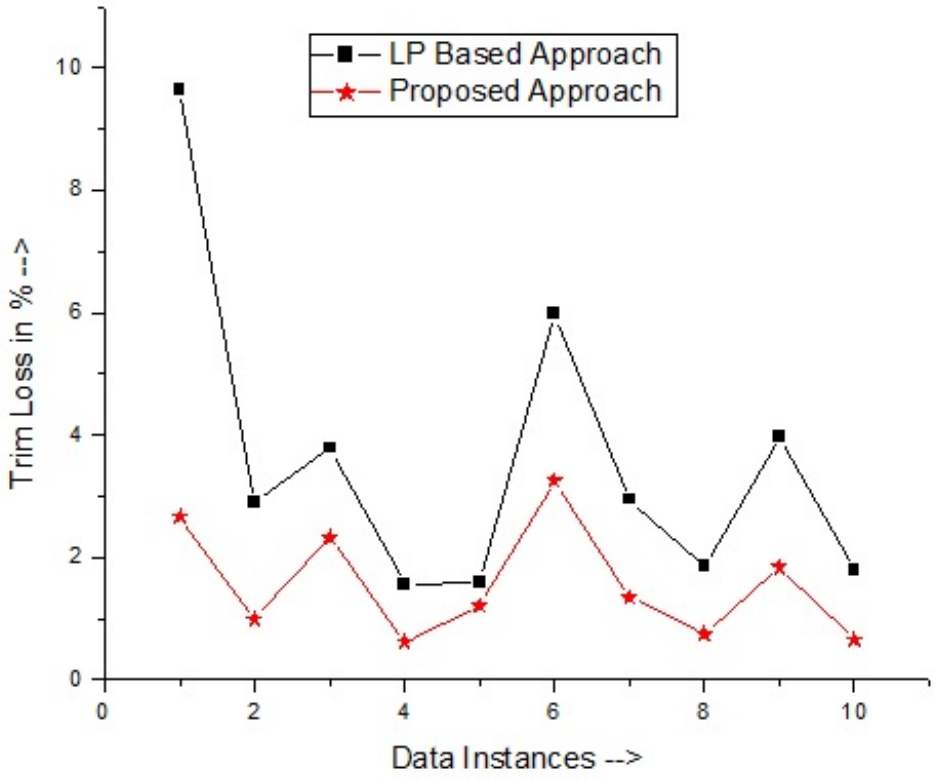


Figure 3.4: Trim loss produced by different approaches.

the demand to the length ratio is high or low. The parameters used to carry out this test are as follows: the population size is kept as 100 with the number of generations 50. Further, two-point crossover operators are used with the flip mutation technique. Note that the population size is reported for medium to large problem instances. The benchmark datasets discussed in the chapter are as reported by Liang *et al.* [22].

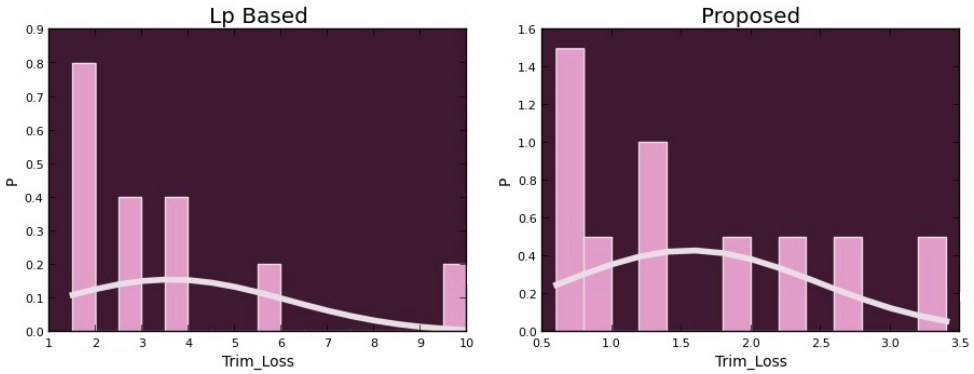


Figure 3.5: Statistical analysis of integrated LP-based and proposed approach.

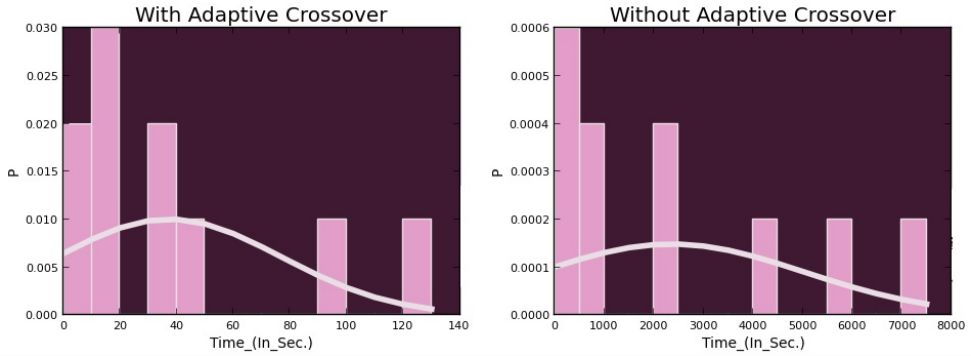


Figure 3.6: Statistical analysis for adaptive crossover.

3.6.2 A Comparison with Integrated Approach

The proposed approach is compared to the LP-based another hybrid approach that combines LP with GA by Umetani *et al.* [118] as reported in Table 3.1. The results are reported for 10 experimental runs. The number of runs depend on the problem size of the instances. It is observed that due to the simple calculative nature of our approach, good patterns are generated in a reasonable time in comparison to LP approach. The table also reports the results in terms of reduction in trim loss and the number of columns generated by both the approaches. This effect is significant while dealing with bulk production in the manufacturing industries. On an average the reduction in number of generated columns on the random dataset is found to be around 70.33%. The performance behavior of both these approaches in terms of trim loss reduction is shown in Figure 3.4. It is observed that the proposed approach is able to minimize trim loss due to its capability to determine better cutting patterns.

Statistical Analysis

The statistical analysis t-test is performed at 95% confidence interval. It is observed that the difference between the exact and the proposed approach is statistically significant. Figure 3.5 shows that there exists a difference between at least two groups. Hence, we conclude that the hypothesis, we selected at the beginning of t-test, which states both the approaches have the same effect on the data sample, is not significant. It is found that the impact of the proposed approach is significantly greater than the LP integrated approach. Similarly, Figure 3.6 shows the results of statistical analysis of finding the effect of with and without adaptive crossover on the randomly generated dataset. As shown

by the curve, the performance with adaptive crossover outperforms the other one *i.e.*, without adaptive crossover for this class of the dataset. Such a result motivates to consider an adaptive crossover during population evolution rather than keeping it a constant factor for all generations.

Table 3.2: Comparison of trim loss % for different metaheuristic approaches.

Sr. No.	Dataset	Tabu search	Simulated annealing	Ant colony optimization	Proposed approach
1.	D1	3.781	4.521	2.491	2.670
2.	D2	3.271	3.437	1.050	0.981
3.	D3	4.971	5.485	3.862	2.320
4.	D4	4.328	4.837	2.853	0.621
5.	D5	7.561	7.931	3.624	1.20
6.	D6	9.651	10.52	5.370	3.25
7.	D7	11.692	10.479	3.749	1.341
8.	D8	4.873	5.684	2.593	0.741
9.	D9	13.671	15.862	8.641	1.830
10.	D10	16.592	20.748	2.457	0.651

3.6.3 Comparison with other Metaheuristic Approaches

The approach is also compared with other metaheuristic approaches of a similar nature like TS and SA by Jahromi *et al.* [39] and ACO by Jap *et al.* [40]). The observed results are summarized in Table 3.2 for 10 experimental runs. The comparative study is analyzed to draw conclusions on the integrated model performance over the individual approach. The results reported shows that the integrated model outperforms all other approaches because of better exploration of the search space. The worst performance is by simulated annealing with an average of 8.95% of trim loss. The ant colony optimization and proposed approach produce the minimum trim loss with an average of 3.67% and 1.56% respectively. However, the proposed approach has been comprehensively able to minimize trim loss and satisfies the specified constraints than the ACO on all the instances.

To illustrate the performance efficiency of the designed model a comparison is further carried out on a different approaches. Table 3.3 summarizes the results for trim loss produced by a metaheuristic on randomly generated data sample for the specified parameter value using CUTGEN1 software.The demand is varied between 10 and 100 for dataset generation. It is observed that on an average

the integrated approach produces minimum trim loss for almost all cases. The maximum trim loss is reported for simulated annealing approach of 12.79%, where as minimum of 10.4% by proposed integrated. TA and ACO show a competitive nature with trim loss as 12.05% and 11.4% respectively.

Table 3.3: Computation results of metaheuristics for random instances generated by CUTGEN1 software.

Class	V1	V2	Demand	Tabu Search	Simulated Annealing	Ant Colony Optimization	Proposed Approach
1	0.01	0. 2	10	4.47	5.09	2.79	2.05
2	0.01	0. 2	100	1.47	2.67	1.83	1.56
3	0.01	0. 2	10	2.52	3.42	2.24	1.98
4	0.01	0. 2	100	1.37	1.53	1.30	0.25
5	0.01	0.8	10	16.97	17.30	16.78	15.41
6	0.01	0.8	100	16.01	16.58	15.63	15.0
7	0.01	0.8	10	15.12	16.81	12.15	11.34
8	0.01	0.8	100	13.68	15.61	11.99	10.72
9	0.2	0.8	10	19.11	19.17	19.27	18.72
10	0.2	0.8	100	18.48	18.55	18.66	18.61
11	0.2	0.8	10	17.48	18.29	17.09	14.76
12	0.2	0.8	100	18	18.49	17.08	14.67

Table 3.4: Effect of adaptive crossover mutation on subproblem evaluation.

Sr. No.	Dataset	Crossover Mutation Convergence Time(in sec.)	
		With change	Without change
1.	D1	0.45	33.71
2.	D2	2.36	181.23
3.	D3	13.15	669.30
4.	D4	13.56	723.90
5.	D5	13.21	13.05
6.	D6	34.56	2074.61
7.	D7	35.63	2053.41
8.	D8	49.21	4437.40
9.	D9	90.50	5892.10
10.	D10	120.3	7498

3.7 Result Analysis

This section presents the result analysis in terms of the impact of adaptive crossover and mutation rates. A complete random search of GA behavior is observed due to large values of P_c, thus, the mutation is required to prevent the

premature convergence of the GA to sub-optimal solutions. Here, the values of P_c, and P_m are varied with respect to the average fitness value and maximum fitness value for the population. As the chosen selection mechanism is elitist, thus, such a change causes a higher convergence rate. Table 3.4 summarizes the effect of change of mutation rate on the convergence. It is observed that the change in P_m value with respect to fitness acceptably reduces the execution time for each generation. It is observed that the proposed approach outperforms on most of the data instances, except for D5 where the convergence rate is nearly equal or in other words the change in crossover rate had no significant impact. But, in general proposed approach consistently performed better than the other heuristic algorithm and also reduces the number of required iterations.

A statistical independent t-test carried to check the behavior of GA with and without adaptive crossover and mutation rates. The parameters used to carry out the t-test are confidence interval of 95% and 10 degrees of freedom. We started with the hypothesis that adaptive crossover and mutation does not influence the GA behavior. The result obtained after computation shows the value of p to be very small, less than 0.001. The result indicates that there exist difference in variance for the two classes, thus, the hypothesis assumed is incorrect. The results are statistically significant.

3.8 Chapter Summary

This work is a first attempt to solve 1D-CSP bounded knapsack subproblem by applying the genetic algorithm and automating the crossover and mutation rate. To improve the convergence rate of the algorithm, a penalty function is enforced on the fitness value. The approach guarantees to generate optimal and sub-optimal column for solving the cutting stock problem. The results are reported on an average 60.53 % reduction in the execution time with the adaptive crossover mutation rate. The average reduction in trim loss in comparison to other similar approach (Umetani *et al.* [118]) is 52.14%. On the other hand, in comparison to other metaheuristic approach, the average performance of our approach, is 73.5% better. Such results motivate to use proposed techniques to solve complex combinatorial optimization problems.

Chapter 4

The Multiple Length 1D-Stock Problem

4.1 Introduction

In this chapter, we are concerned with the generalization of thoroughly studied standard cutting stock problem in the previous chapter. Here, we consider that multiple length stocks are available instead of single one. The different stock lengths provide a greater possibility of generating feasible cutting patterns that may generate a better solution in terms of stock utilization as compared to standard cutting stock problem. This fact is supported by Gilmore and Gomory [10] in their research contributions. However, finding the optimal solution to this class of optimization problem requires to solve a more complex objective function. We have considered additional constraints of restriction on the raw stock availability. In this case, the selection of the stock to be used is based on the priority rather than selection of the best cutting pattern. This class is also referred as the combined assortment and trim loss minimization problem.

The Multiple Length Cutting Stock Problem (MLCSP) is NP-hard [119]. Unless P=NP, no absolute approximation scheme can be devised that solves it in fully polynomial time. We focus on waste minimization due to the large number of improper pattern selections during the evaluation process. The other factors that are considered reduce the number of generated patterns and eliminate repetition. As, increase in the number of generating patterns implies more cycles and thus, takes more computation time. Exact algorithms are not prefer-

able for NP-hard combinatorial optimization problem as the execution time rises with the problem size and even may be intractable for some class of problems. On the other hand, heuristic takes comparatively less time, but does not guarantee an optimal solution. Thus, it is required to find a trade-off between the quality and the computation time to select the most appropriate method to determine the best solution. As a consequence, most of the research efforts have been devoted to the development of heuristics and approximation algorithms.

4.2 Terminology

In this section, we discuss the basic terminology like BCP, branching strategy and rounding procedure used in the proposed approach.

4.2.1 Branch-Cut-Price

The column generation is done by solving the subproblem or the pricing problem. The BCP integrates B&B with LP relaxations, pricing subproblem and applies separation and cutting throughout the B&B tree to solve large optimization problems. To check optimality, a subproblem, is solved to identify columns to enter the basis. The separation problem is formed when the optimal solution to an LP relaxation is infeasible. It identifies the violated inequalities for the class. If one or more violated inequalities are found, then some are added to the LP to cut off the infeasible solution, and then the LP is re-optimized. Branching occurs when no violated inequalities are found to cut off an infeasible solution. The cuts are introduced in order to reduce the total number of nodes and to find a tight bound at a node of the B&B tree. At each node of B&B tree, the columns are generated to tighten the LP relaxation. In the column generation phase of the BCP, the pricing problem is solved to find the new entering column with negative reduced cost. This column is added to the RMP and then the LP is optimized. Thus, the column generation phase results in the addition of columns, which may be likely to generate optimal solution. In BCP, column with most negative reduced cost is selected to be a part of RMP whose probability of being in the final optimal pattern is quite high.

LP Relaxation

The efficiency of the computation method depends on the number of constraints involved in the linear programming. The increase in the number of constraints for column generation is quadratic. Since, a complete enumeration of the columns is impractical, the partial enumeration is resorted for the pattern set and deal with the RMP. A solution to the RMP is an upper bound on the LP optimum. After the master is optimized, K pricing subproblems are solved to identify those columns with a negative reduced cost that may enter the basis and provide a better cost solution. The solution of the RMP is optimal for the LP relaxation if all variables of the LP relaxation have non-negative reduced cost. Since only a subset of the columns of the LP relaxation is available explicitly, this cannot be checked directly to find whether it is negatively reduced column or not. A pricing algorithm is used to verify the column optimality. If the solution is not optimal, the pricing algorithm identifies at least one column with negative reduced cost. This column is added to the RMP, and the basic procedure continues. The column generation scheme terminates when no columns with negative reduced cost exist anymore. The optimal solution to the RMP is then also optimal to the LP relaxation. This process of adding columns to the RMP when required is called an implicit column generation. In an explicit column generation, all columns are generated in advance and are kept in computer memory. Subsets of columns are then passed to the RMP until a subset is found that contains the optimal solution.

4.2.2 Branching Strategy

The branching scheme basically defines the search strategy, *i.e.*, the order in which the node will be evaluated. The task of B&B terminates when both, the lower and the upper bound becomes equal. The algorithm can follow the branching strategy based on the trade-off between the improvement of the lower bound and quickly determines the feasible solution. In the proposed approach, we have focused on improving the global lower bound. This is achieved by strengthening the weakest LP relaxation. Thus, the search strategy is to select the lowest LP bound node at each iteration. This strategy is termed as best first search. To explore the tree, we used depth-first search strategy. The

experimentation that we carried out gives satisfactory results for the search strategy.

4.2.3 Rounding Procedure

The rounding scheme applied in the BCP is a simple one that considers the total number of available rolls. In this scheme, the positive variable with the fraction part greater than 0.5 are rounded up else rounded down. This results in the case where the supply exceeds the demand quantity. To tackle this problem, we consider the excess as residual and consider it to satisfy other smaller demand lengths. Amongst the available residual patterns the preference is given to the one that results in more saving. Computation shows that a simple selection scheme improves the overall performance.

Table 4.1: Notation and interpretation.

Symbols	Definition
N	Total number of ordered demand
M	Total number of available multiple length stock
W_j^k	k^{th} available stock of width W_j
Q_L	Vector of available stock quantity
w_i	i^{th}order item piece length
d_i	i^{th}order demand quantity
c_{ijk}	Pattern decision value
UB_{ik}	Maximum possible i^{th} demand that can be satisfied by the k^{th} stock

4.3 Problem and Mathematical Formulation

Various notations and their interpretation are listed in Table 4.1. The multiple stock size is an extension of 1D−CSP, where the number of available stocks is variable in size and limited in quantity. The problem can be defined as: Given m number of demand lengths (w_i, ..., w_m) with order demands as (d_i, ..., d_m) to be met from M number of stock type of varying stock length L_1, ..., L_M. Given the total available stock W_L (L=1, ..., M), where each stock is available in quantity vector Q_L (L=1, ..., M), order for each item is length, w_i (i=1, ..., N) with the required demand as d_i (i=1, ..., N). The problem is to satisfy customers ordered demand with an overall minimum trim loss. To cut the available stock, the correct cutting patterns are to be determined.

Thus, the generation of these cutting patterns is considered as the subproblem to CSP. This RMP is solved by using the BCP approach for integer linear programming because the linear programming approach is suitable and easy to restore feasibility. The subproblem of pattern generation is solved by heuristic techniques. Integrating heuristic approach with LP helps in speeding up the process as it facilitates to modifying the fathom criteria for the node that helps in removing a large number of nodes, and thereby, greatly reduces the search space. Further, it also guarantees that the optimality gap is reduced.

Here, the c_{ijk} is the cutting decision column value, which indicates the number count of order demand i in the j^{th} cutting pattern for the available stock length k, i=1, ..., N, j=1, ..., M, k= 1, ..., K. A column vector (c_{1jk}, c_{2jk}, ..., c_{Njk}) indicates j^{th} cutting pattern for a stock length type k, is feasible if it satisfies

$$\begin{aligned} w_1c_{1jk} + w_2c_{2jk} + \cdots + w_Nc_{Njk} \leq W_k^{Q_{L(k)}}, \\ 0 \leq c_{ijk} \leq d_i, integer \end{aligned} \tag{4.1}$$

The classical 1D-CSP model with multiple stock is formulated as follows:

$$\begin{aligned} min \sum_{j=1}^{Q_L(1)} {W_1}^j + \sum_{j=1}^{Q_L(2)} {W_2}^j + \cdots + \sum_{j=1}^{Q_L(K)} {W_K}^j \\ s.t. \sum_{j=1}^{Q_L(1)} w_ic_{ij1} + \sum_{j=1}^{Q_L(1)} w_ic_{ij2} + \cdots + \sum_{j=1}^{Q_L(1)} w_ic_{ijK} \geq d_i, \\ i = 1, \ldots, N \end{aligned} \tag{4.2}$$

In this model, the c_{ijk} refers to column value for the type of item i in the j^{th} cutting pattern for the selected available stock W_k. Equation 4.1 specifies the constraint that the summation of product for the cutting pattern generated to meet the given demand lengths cannot exceed the available stock length. Note, that the value of cutting pattern (c_{ijk}) generated is a positive integer less than or equal to the i^{th} demand. Equation 4.2 specifies the constraint for multiple stock where the objective is to minimize the use of different available stock subjected to constraint that order demands are fulfilled.

The *Pat-Gen* model for 1D-CSP integer programming problem is formulated as follows:

$$\max \sum_{i=1}^{N} w_i c_{ijk}, j = 1, \ldots, N \tag{4.3}$$

$$s.t. \sum_{i=1}^{N} w_i c_{ijk} \leq W_k - w_i, i = 1, \ldots, N \tag{4.4}$$

$$0 \leq c_{ijk} \leq UB_{ik} \tag{4.5}$$

Here, the UB is stand for upper bound and is computed as follows:

$$UB = \lfloor \frac{W_k - w_i}{w_j} \rfloor, j = 1, \ldots, N \tag{4.6}$$

The objective function in Equation 4.3 is a knapsack problem to maximize the utilization of remaining space after removing the length of a selected single item (w_i). The first constraint enforces that the cutting pattern decision value (c_{ijk}) must not exceed the upper bound. The computational upper bound, corresponding to each item is computed using Equation 4.6. Setting up an upper bound helps in restricting the decision value and always generating feasible cutting patterns. The second constraint ensures that the total length cut out can never exceed the remaining stock length.

4.4 Proposed Methodology

The basic concept of heuristic approach is to generate an optimal cutting pattern after each iteration and use it exhaustively to meet the demand without exceeding the availability of associated stock. The proposed heuristic approach uses two stages of execution. In the first stage, is the pattern generation phase where the columns are generated using the proposed model. The second stage uses heuristic where the tasks are the selection of pattern, determining the number of patterns to be cut, updation of parameters like the availability of the remaining stock and remaining order demand are carried out. Both the stages are explained in detail in the next subsections.

4.4.1 Pattern Generation Approach to Accelerate Column Generation

The *Pat-Gen* is the main model for the pattern generation, which guarantees the determination of the least cost cutting patterns in terms of optimality. This

model is similar to a knapsack, in which the objective function is to maximized the profit by selecting appropriate objects from the available. In this model, we have also proposed *one-at-a-time scheme*, which is motivated by the fact that at least a portion of the selected item order demand is fulfilled from the available stock. Thus, it generates some pattern for the item order (say i) from the available greater stock length. It is presumed that at least one part of the ith item demand will be fulfilled by the selected available stock. Thus, the remaining portion of the selected available stock is subjected to *Pat-Gen* model. The output cutting pattern obtained after applying the *Pat-Gen* is updated at ith index by incrementing the value by one.

We start the pattern generation process with available greatest length stock for each of the order demand stock. This ensures that its ordered demand is met with the available larger stock. The process starts by taking each ordered demand at a time. For each order item, a corresponding *stock set* is created where each *stock set* consists of the available stock whose length is greater than or equal to the selected item, which ensures the solution feasibility. Thus, in the worst case, for each item the number of possible patterns generated would be N x M in the initial iteration. However, as the iterations are carried out further, the number would definitely reduce, since some ordered demand gets fulfill with the completion of each iteration.

In the *Pat-Gen* model, we have modified the classical column generation model by reducing the right hand coefficient of the constraint by the currently selected item length for which the patterns are generated. This modification of constraint helps to generate specific optimal pattern possible for the available stock, which results in the reduction of computation time by some fractional amount. The integer linear programming is used to solve this problem generated after selection of best feasible columns amongst the generated ones. The problem can be solved by using any of the existing integer linear programming solvers. The proposed approach uses efficient BCP exact method (Feilleta *et al.* [20]) to determine whether the selected pattern results in integer solution or not. The BCP algorithm, which is based on a delayed column generation approach within a branch-and-bound framework, is proven to work well and finds the optimal solutions for multiple cutting stock practical scenarios within the reasonable time.

Algorithm 3 Integrated Heuristic with Integer Programming

```
Input: Cutting stock demand and available stock
Output: Optimal CSP solution with minimum trim loss
Begin
 Arrange item piece order demand in decreasing order
for each order item piece demand do
    for each multiple available stock ( s.t W_L ≥ w_i) do
        Generate cutting pattern (using Pat_Gen model )
        Compute produced waste
    end for
end for
for each order demand do
    Select a pattern with minimum waste
    Update stock availability and demand order of stock fulfilled by selected
pattern
    if no demand order is satisfied then
        Move to next order demand to select pattern
    else
        Update order demand
    end if
end for
 Output solution and stop
End
```

The obtained cutting patterns are stored and the respective waste produced is computed for the current order item demand. Amongst the entire patterns, the one that produces the minimum waste is selected. Equation 4.7 is then used to determine the possible number of cuts to meet demand for the item. The next task is the updation of all parameters as discussed in the next subsection. The pattern generation can encounter three cases:

Case-1. Available stock and unmet demands In particular, this is a general case in which, our algorithm considers plenty of multiple length available stocks as well as the demand requirement for the evaluation. In every iteration, some part of the demand is met and the process continues till either out of stock or the required demand are fulfilled.

Case-2. Available stock and few fulfill demand Another case is when, not all, but a few of the order demand requirements are met. This case may occur when the selection of patterns with minimum waste may possibly lead to a state that some order demands are met and few are remaining. In such a case, the items for which the demand is fulfilled, are not considered further in the pattern generation process. This helps in speeding up the computation task, as

the number of items to be considered are reduced. Such change in computation does not affect the functioning of the model. Thus, the model supports flexibility as dynamic changes can be easily accommodated to generate a pattern.

Case-3. Demand unmet with available stock This case discusses the possible scenarios in which the available stocks are fully utilized and still the demands are unmet. The chance of occurrence is when the greater order demand for stocks are not fulfilled, whereas the shorter order demands are preferred and cut from the larger available stock. This case can happen only during the selection of pattern with minimum waste. To resolve it, if two patterns generated for different order item from the different available stock results in the same trim loss, then the preference is given to available stock with lower length. Thus, while designing the model, this aspect was addressed by giving selection priority to the order demand with maximum length.

4.4.2 Computational Selection and Updation

The number of patterns to be cut for the ordered item demand w_i from the available stock W_k is determined by the equation below.

$$NumPattern_{w_i} = min(Avl_{w_i}, \lfloor \frac{d_1}{c_{1jk}} \rfloor, \ldots, \lfloor \frac{d_N}{c_{Njk}} \rfloor) \quad \forall c_{ijk} \neq 0 \tag{4.7}$$

Here, Avl_{w_i} refers to the quantity of the selected available stock to meet order demand (w_i), $NumPattern_{w_i}$ is the total number of patterns that will be selected for the given demand. The value is computed as the minimum amongst the quantity of selected available stock (Avl_{w_i}) and demand fulfillment for different order item, *i.e.*, the maximum number of demands that can be fulfilled using the selected pattern. The symbol $\lfloor\ \rfloor$ represents the floor value corresponding to the obtained number. After the selection, other parameters like availability of stock and remaining requirement of demand are updated.

$$Avl_{w_i} = Avl_{w_i} - NumPattern_{\ w_i} \tag{4.8}$$

$$d_i = d_i - NumPattern_{\ w_i}\ c_{ijk} \tag{4.9}$$

The trim loss associated with each cut pattern selected (based on the cutting pattern matrix) with respect to available stock used is determined as follows:

$$TrimLoss(TL_{ij}) = W_k - \sum_{i=1}^{N} w_i c_{ijk} \tag{4.10}$$

Setting up the upper bound on cutting decision value helps to reduce the number of patterns. If the subproblem of pattern generation is solved to optimality, then the new solution obtained is at least equal and in general, better than the existing solution. This helps in the selection of optimum pattern with minimum trim loss.

Table 4.2: Random instances for CSP.

Instance	n	m	L	v_1	v_2	j_{min}	j_{max}	k_{min}	k_{max}	$\tilde{d}$
$Rand_1$	1	40	1000	0.01	0.2	1	15	1	10	43.2
$Rand_2$	1	40	1000	0.01	0.2	1	15	1	15	17.4
$Rand_3$	4	40	1000	0.01	0.2	1	15	1	15	30.5
$Rand_4$	4	60	1000	0.01	0.2	1	15	1	15	16.2
$Rand_5$	6	60	1000	0.01	0.2	1	15	1	10	8.64
$Rand_6$	4	60	1000	0.01	0.2	7	15	1	10	30.21
$Rand_7$	6	100	1000	0.01	0.2	7	15	1	10	23.32
$Rand_8$	1	100	1000	0.01	0.2	7	15	7	15	51.3
$Rand_9$	4	100	1000	0.01	0.2	7	15	7	10	18.3
$Rand_{10}$	4	100	1000	0.01	0.2	7	15	7	15	49.32
$Rand_{11}$	6	40	1000	0.1	0.8	1	15	1	15	52.8
$Rand_{12}$	1	40	1000	0.1	0.8	1	15	7	15	9.34
$Rand_{13}$	1	40	1000	0.1	0.8	1	15	8	15	32.1
$Rand_{14}$	4	40	1000	0.1	0.8	7	15	1	15	14.2
$Rand_{15}$	6	60	1000	0.1	0.8	1	15	1	15	62.7
$Rand_{16}$	1	60	1000	0.1	0.8	8	15	1	15	43.6
$Rand_{17}$	1	60	1000	0.1	0.8	1	15	1	10	33.7
$Rand_{18}$	4	100	1000	0.1	0.8	1	15	1	15	15.28
$Rand_{19}$	4	100	1000	0.1	0.8	1	15	1	10	19.4
$Rand_{20}$	6	100	1000	0.1	0.8	8	15	1	15	27.32
$Rand_{21}$	1	40	1000	0.2	0.7	8	15	7	15	29.6
$Rand_{22}$	1	40	1000	0.2	0.7	1	15	1	10	36.6
$Rand_{23}$	4	40	1000	0.2	0.7	1	15	1	10	50.23
$Rand_{24}$	1	60	1000	0.2	0.7	7	15	8	15	45.2
$Rand_{25}$	4	60	1000	0.2	0.7	7	15	7	15	35.1
$Rand_{26}$	1	60	1000	0.2	0.7	7	15	7	15	54.12
$Rand_{27}$	1	60	1000	0.2	0.7	1	15	1	10	41.5
$Rand_{28}$	4	100	1000	0.2	0.7	1	15	1	10	18.3
$Rand_{29}$	6	100	1000	0.2	0.7	1	15	1	10	23.77
$Rand_{30}$	4	100	1000	0.2	0.7	1	15	1	15	26.4

Table 4.3: Real world instances from paper tube industry in Japan Matsumoto *et al.*(2011) [120].

Instance	n	m	L_{min}	L_{max}	j_{min}	j_{max}	$\tilde{d}$	l_{min}	l_{max}
P1	3	14	1300	1800	5	10	87.1	66	544
P2	1	15	1800	1800	10	10	100.0	88	411
P3	2	16	1700	1900	5	12	67.1	114	550
P4	3	17	1350	1800	5	5	60.3	642	1057
P5	4	15	1350	2100	10	10	52.0	608	1058
P6	4	17	1350	2100	5	5	60.3	642	1057

Table 4.4: Real world instances from fiber industry in Japan Matsumoto *et al.*(2011) [120].

Instance	n	m	L_{min}	L_{max}	$\tilde{d}$	l_{min}	l_{max}
F1	1	6	5180	9080	33	520	1250
F2	1	7	5180	9080	27.5	650	1020
F3	1	8	5180	9080	56.87	500	1200
F4	1	9	5180	9080	29.88	500	1500
F5	1	10	5180	9080	34.9	750	1250
F6	1	11	5180	9080	32.18	500	1450
F7	1	13	5180	9080	21.46	920	1250
F8	1	14	5180	9080	15.92	635	1440
F9	1	15	5180	9080	22	500	1250
F10	1	16	5180	9080	26.18	500	1340
F11	1	17	5180	9080	31.06	500	1200
F12	1	18	5180	9080	27.44	500	1440
F13	1	19	5180	9080	35,10	650	1350
F14	1	20	5180	9080	8.85	500	2000
F15	1	23	5180	9080	31.26	530	1520
F16	1	26	5180	9080	43.11	500	2000
F17	1	28	5180	9080	17.85	500	1250
F18	1	29	5180	9080	13.13	500	1360

4.5 Computational Results

This section discusses the data instances used, and the comparison result, amongst various proposed and existing heuristic approaches.

4.5.1 Data Instances

We carried out computational experiments on random as well as real industrial data instances. The random instances for 30 classes were generated by setting up the parameters in the CUTGEN1 software by Gau and Wäscher [117]. The generated instances use different combinations for the parameters L, m, *v1*, *v2*, $\tilde{d}$, where the demand is computed by two random variables j and k such that

$d_i = j_i \cdot k_i$. The length of each item l_i is randomly selected within the [v_1L, v_2L] interval. The parameter to compute demands j_i and k_i are random in the range [j_{min}, j_{max}] and [k_{min}, k_{max}] respectively. Table 4.2 summarizes the details where L is set to 1000, j_{min} and j_{max} varies in range (1, 15) and (7, 15), and k_{min} and k_{max} varies in range (1, 7) and (7, 15). Here, $\tilde{d}$ denotes the average demand. To generate the random instances, we have considered three ranges for v_1 and v_2, *i.e.*, for classes 1-10 the value is set (0.01, 0.2), for classes 11-20 the range is (0.1, 0.8) and for classes 21-30 the range is (0.2, 0.8). This random variation in the input helps in rigorous testing of the proposed approach. On the other hand, the industrial data instances were collected from Matsumoto *et al.*(2011) [120] which is a real data from Japan paper tube reported and fiber factory.

The real data includes 6 instances that are taken from the real applications in a paper tube industry in Japan as shown in Table 4.3. The dataset size is a maximum of 17 instances with the available multiple stock ranging from 1 to 4 with the maximum length of the available stock up to 2100 and the maximum demand for item is 420. Table 4.4 shows another real world data of a chemical fiber company in Japan. These are 40 instances with m ranging from 6 to 29, with L equals to either 5180 or 9080, l_i ranging from 500 to 2000, and d_i ranging from 2 to 264.

4.5.2 Assessing the Performance of Proposed Heuristic Against Other Existing Heuristics

As the problem size grows, the difficulty to solve combinatorial problems in polynomial time increases because of the expansion of the solution space. This feature promotes the use of heuristic algorithm as they produce an acceptable solution in reasonable time with advances in computing technology. The proposed work is compared against three well known algorithms from the literature: a standard column generation algorithm for multiple stock CSP (by Gilmore and Gomory [10]), sequential heuristic (by Cui Y. *et al.* [121]), and a TS algorithm (by Matsumoto K. *et al.* [120]), which are based on shift neighborhood that solves the auxiliary bin packing problems with the first-fit decreasing heuristic algorithm.

Table 4.5: Comparison amongst heuristic algorithm on trim loss and time (in sec.).

Instance	CG		Tabu Search CG		Sequential Heuristic		Proposed	
	Trim Loss(%)	CPU Time(sec.)	Trim Loss(%)	CPU Time(sec.)	Trim Loss(%)	CPU Time(sec.)	Trim Loss(%)	CPU Time(sec.)
$Rand_1$	0.0	0.43	5.43	3.57	0.0	0.0	0.0	0.35
$Rand_2$	5.7	1.37	12.5	5.46	8.3	0.23	4.3	0.58
$Rand_3$	0.0	1.09	6.92	2.3	0.5	0.58	0.00	0.56
$Rand_4$	0.0	1.34	2.37	12.56	0.9	1.07	0.00	0.31
$Rand_5$	0.0	0.57	1.49	2.8	0.0	1.02	0.00	0.49
$Rand_6$	15.4	2.47	27.4	4.12	15.4	1.29	12.4	1.51
$Rand_7$	13.9	1.39	30.9	2.59	-	-	10.05	0.56
$Rand_8$	0.0	5.19	4.39	110.4	-	-	0.00	3.56
$Rand_9$	0.0	3.48	8.7	14.1	-	-	0.0	2.49
$Rand_{10}$	11.4	6.31	31.4	70.36	-	-	9.4	9.48
$Rand_{11}$	6.2	5.21	18.3	16.4	5.4	3.19	5.7	4.46
$Rand_{12}$	0.0	9.26	9.2	112	1.3	7.45	0.0	7.34
$Rand_{13}$	7.4	10.05	19.43	38.1	7.4	8.62	7.4	9.54
$Rand_{14}$	7.5	1.57	15.3	70	7.5	0.57	6.7	1.37
$Rand_{15}$	0.0	9.50	6.8	4.8	0.0	9.50	0.0	6.56
$Rand_{16}$	9.2	7.42	25.3	22.8	11.4	6.21	8.1	6.4
$Rand_{17}$	5.8	4.59	14.2	12.7	6.2	3.53	5.8	3.3
$Rand_{18}$	10.4	12.29	37.2	117	-	-	8.17	10.37
$Rand_{19}$	12.6	5.08	14.7	107	-	-	9.2	4.9
$Rand_{20}$	0.0	10.34	12.7	4.6	-	-	0.0	9.54
$Rand_{21}$	7.9	3.05	16.9	9.2	7.9	3.05	6.4	1.01
$Rand_{22}$	0.0	8.39	13.68	9.31	0.0	8.39	0.0	7.5
$Rand_{23}$	5.2	4.21	19.4	9.4	5.7	3.39	4.5	3.6
$Rand_{24}$	4.8	8.32	6..59	9.2	4.95	6.45	4.12	5.3
$Rand_{25}$	0.0	6.31	17.4	3.6	0.4	4.51	0.0	5.12
$Rand_{26}$	0.0	4.11	4.9	5.4	0.9	3.15	0.0	2.41
$Rand_{27}$	6.7	15.3	28.2	5.74	7.5	13.25	5.5	13.4
$Rand_{28}$	15.4	10.47	28.3	190	-	-	12.8	9.45
$Rand_{29}$	0.0	6.45	12.7	5.4	-	-	0.0	5.30
$Rand_{30}$	5.9	11.57	23.6	147.23	-	-	4.15	10.3

Table 4.6: Exact approach performance using real world data (tube factory) from Matsumoto *et al.*(2011) [120].

Instance	BP-no cut CPU-Time(sec.)	BCP CPU-Time(sec.)	Proposed Integrated CPU-Time(sec.)
P1	5.04	7.92	2.40
P2	2.75	5.75	1.43
P3	16.3	24.7	7.50
P4	3.26	8.26	2.61
P5	4.1	7.95	3.67
P6	0.93	0.94	1.35

Table 4.7: Performance of Exact approach using fiber industry instances from Matsumoto *et al.*(2011) [120].

Instance	BP-no cut CPU-Time(sec.)	BCP CPU-Time(sec.)	Proposed Integrated CPU-Time(sec.)
F1	2.46	3.61	2.41
F2	4.74	43.4	3.76
F3	7.33	4.67	3.61
F4	12.2	8.85	6.94
F5	6.88	6.37	5.92
F6	6.52	11.1	5.64
F7	2.09	2.07	1.63
F8	3.86	5.81	2.85
F9	3.57	3.04	1.57
F10	5.59	13.4	2.72
F11	3.17	5.3	2.86
F12	18.48	147	15.7
F13	6.05	11.2	5.40
F14	-	35.7	29.5
F15	148	232	130
F16	77	31.5	27.8
F17	207	15.4	85.4
F18	19	8.58	13.85

The computational results show that the method performs favorably with respect to other global optimization procedures. Table 4.5 summarizes the effectiveness of the approach by comparing it with others heuristics from the literature. The results obtained are promising, quite close in many cases and better in many cases as compared to CG approach. Results are far better than the tabu search in terms of computation time as in this searching approach it is required to explore the solution depth, which sometimes may not possible due to the restarting procedure that may compromise the effectiveness of the approach. We observe that both, the SHP and integrated approach, attain small trim loss, but the approach failed to find a solution on large instances. Based on the ob-

servation of our approach behavior, we can conclude that simple initial design procedures are better to implement, which facilitate in better global iteration.

Table 4.6 reports the experimental results on real world data instances for all the four approaches. For instance P3, the results obtained by CG, SHP and proposed heuristic based integrated approach are comparable. The proposed approach takes very less computation time. However, only for P6, the proposed approach took slightly higher time. The computation time average reduction of 41.44 % against CG and 65.8% against SHP is observed for proposed integrated. The observation shows the adaptation of the proposed technique by manufacturing industries would result in a huge amount of economic saving.

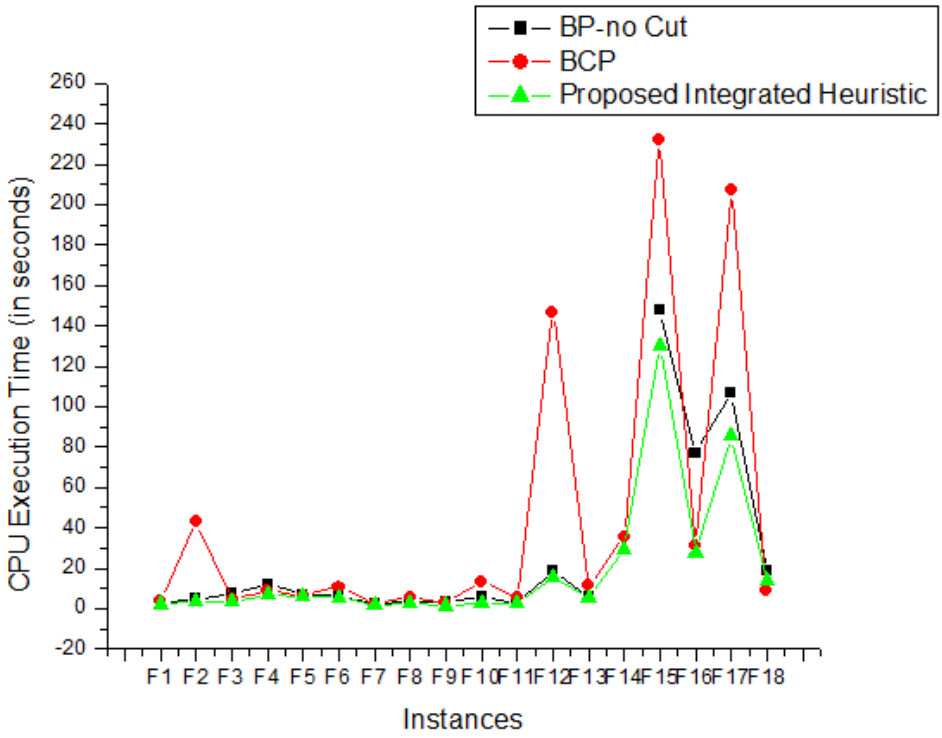


Figure 4.1: Graphical interpretation of execution time with exact approaches.

4.6 Assessing the Performance of Proposed Heuristic Against Exact Approach

The major drawback of exact approaches is their incapability to find the optimal solution for large instances. As the problem size grows the number of computation tasks increases and sometimes the process becomes too time consuming. The enumerative approaches based on B&B try to explore the search space.

Table 4.8: Heuristic algorithm performance on real world data (paper tube factory).

Instance	CG		Tabu Search CG		Sequential Heuristic		Proposed Integrated	
	Trim Loss(%)	CPU Time(sec.)	Trim Loss(%)	CPU Time(sec.)	Trim Loss(%)	CPU Time(sec.)	Trim Loss(%)	CPU Time(sec.)
P1	0.0	0.17	1.0	100.17	0.5	0.94	0.32	0.40
P2	0.0	0.09	1.2	26.59	0.4	0.67	0.0	1.43
P3	0.0	0.28	1.5	158.61	0.5	0.60	0.5	0.50
P4	0.0	0.34	2.5	10.17	2.5	2.50	0.0	0.61
P5	2.5	0.11	3.4	76.83	3.0	1.80	2.7	1.67
P6	2.0	0.20	4.4	148.45	3.9	3.20	3.4	1.35

The space grows for the large scale problems and proves to be computationally infeasible. We exploit this shortcoming by introducing a heuristic in this search process, which greatly reduces the search space. Another principal advantage of greedy approach is that it is cheaper in both the space and the time requirement. In this way it guarantees that the value of the best solution provided by the procedure is no farther than the distance from the optimal value of the problem. Table 4.7 shows the behavior of the exact approach on the paper tube data instance. Here, we have considered two exact approaches the cutting plane algorithm for multiple stock length (Belov G. *et al.*) [23] and BCP (Feilleta *et al.*) [20]. The table summarizes the results for the CPU execution time. It is reported that on an average reduction in computation time is 19% against BP-not cut and about 54.5% against BCP. Figure 4.1 shows a clear distinction in the performance of all the exact approach. It is observed that it is better to invest the time for searching few starting points rather than exploring the entire search space.

The statistical t-test carried between exact and integrated approach for fiber industry data instances reported in Table 4.8. The statistical test is carried to check the significance of the result obtained by both the approaches. The test is carried with 10 degrees of freedom at a confidence interval of 95%. The obtained computation shows that the value of p is very small, which indicates that both the approaches a difference of variance is observed. The result is statistically significant and the null hypothesis that both approaches behave in similarly in this class of data instances is rejected.

The table summarizes the performance comparison on computation time of exact approaches on real world fiber industry data. The results obtained highlight the fact that an integration of a small heuristic approach helps in reducing the computation time to a large extend when proposed in compared with BCP. In comparison to BP, it is observed that the computation time is slightly higher. However, the optimal or near optimal solution is guaranteed in reasonable time.

4.7 Chapter Summary

This chapter has presented a new integrated heuristic method for the determination of the optimal pattern in multiple stock 1D-CSP. Its main contribution to the literature over other existing one is that the approach does not enumerate all possible patterns as the size of combinatorial problem increases. Another benefit is that the approach is able to produce optimal result, even when the ratio between the available stock and the order demand is low. The proposed method is a good tool to solve the real world problems with an acceptable solution in reasonable computation time.

Chapter 5

Metaheuristic Approach to 2D-Strip Packing

5.1 Introduction

Manufacturing and production industries very often come across the problem where a given stock material must be cut into a smaller set of rectangular shapes. This problem is termed as two dimensional orthogonal rectangular strip packing problem (SPP), which is NP-hard in nature (Garey *et al.* (1979) [119]). This class of combinatorial optimization finds significant relevance in different domains of operations research. In industries like paper and pulp, wood, textile etc., the problem is to determine how the arbitrary rectangular block set would be cut from the available stock. The problem variant like arrangement of articles, reports and advertisement, is considered in the newspaper field. In pharmaceutical packing industry, many strip packaging approaches seem ideal for high speed sealing of coated or uncoated tablets, capsules or lozenges of any shape or size in aluminum foils, polythene, cell phone, etc. It is an interesting real world industrial problem, where the objective is to provide the best arrangement with the aim of waste minimization. There are two broad categories of the solution approach, namely exact and inexact. A major bottleneck with the exact approaches is that as the problem size grow and become complex, the computation time also grows exponentially. Thus, the researchers focus more on the inexact approaches in comparison to exact.

In today's world of industrialization, where the mass production and high material utilization are the crucial factors for growing industries. It necessitates, the need of finding correct cutting patterns, which may result in small improvement. As in long term run, it leads to huge economical saving. These problems occur at wide scale in many industries, where the complexity is determined by

the shape of the item to be cut and the number of applicable constraints. Textile and shipping industries basically deal with irregular shaped items. Automation in a packing system saves much economic time and is preferred over the manual. Textile industry tackles this problem, but under different head, namely Nesting and Marker making. It involves finding the best layout for the cutting of irregular shapes, where the shapes being allowed a rotation from 0 to 180 degrees. Regular and irregular shape packing are being addressed in the ship-building industry, where it is required to investigate how the irregular size items can be packed and transported in huge containers. As in todays scenario of surplus demand an automated system is required for efficient packing, thereby, reducing the transportation damage risk. The packing problem is not limited to industries, but can also be seen in other dimensions like in very-large-scale integration (VLSI) design, memory allocation during storage, and in the field of optical fiber communication.

5.2 Terminology

This section presents a brief introduction to two dimensional strip packing and biased random key approach used with genetic algorithm.

5.2.1 Two Dimensional Strip Packaging

The 2D-SPP deals with a set of rectangular blocks that must be arranged in a given container of fixed width and infinite height with the objective of minimizing the highest point of any rectangle in the solution. A feasible placement is one, where no rectangles overlap one another within the container and are arranged parallel to the container edge, *i.e.*, orthogonally. Additional constraint like rotation of blocks by 90 degrees is considered in this work.

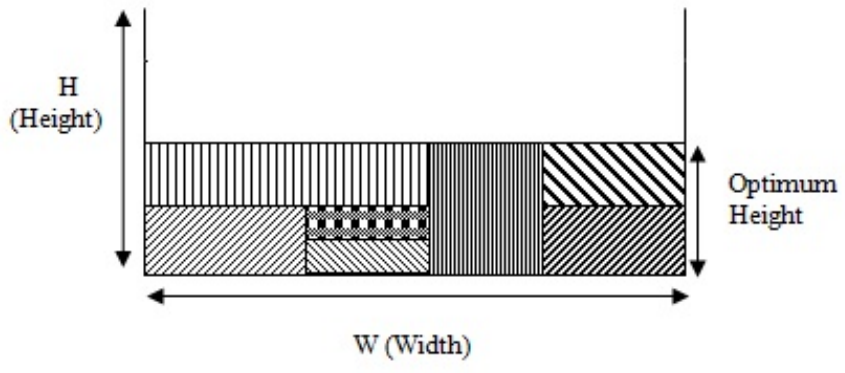


Figure 5.1: A view of 2D-strip packing.

Given a huge rectangular container of dimensions say, where W stands for fixed width, and H denotes the infinite height of the container. Considering m rectangles of smaller dimension as compared to container are to be packed where the ith rectangle dimensions is w_i x h_i.

The placement is subject to non-guillotine cut, *i.e.*, the rectangles are placed parallel to the edge of the container. We worked on the problems, with integer and real dimensions easily without any extension or change. The SPP is well explained in Figure 5.1. The shaded rectangles show different arbitrary size rectangular blocks that are packed into the given container. Optimum height denotes the best possible height that can be achieved after placing all the rectangles. In this subproblem for strip packing, *i.e.*, 0-1 knapsack, is indeed NP-hard in nature and this de-generative case makes the entire problem NP-hard. The model can be formulated as maximizing the area occupancy that in turn results in minimizing the overall height of the layout, the representation of which is as follows:

$$\max \sum_{i=1}^{m} w_i h_i \lambda_i \tag{5.1}$$

λ_i can take the value 0 or 1 indicating whether the ith rectangle is placed or not. We designate each rectangle bottom left most corner coordinates of the rectangle as (x_i, y_i). The governing constraints are defined as

$$x_i + w_i \leq W, \quad i = 1, ..., m \tag{5.2}$$

$$y_i + h_i \leq H, \quad i = 1, ..., m \tag{5.3}$$

$$\begin{aligned} x_i + w_i \leq x_j \;\; or \;\; x_j + w_j \leq x_i \;\; or \\ y_i + h_i \leq y_j \;\; or \;\; y_j + h_j \leq y_i \\ \forall i, j \;\; where \;\; i \neq j \end{aligned} \tag{5.4}$$

$$\lambda_i \in 0, 1 \tag{5.5}$$

$$x_i, y_i \geq 0 \tag{5.6}$$

The constraints 5.2, 5.3 and 5.6 ensure that the rectangles are placed within the boundary of the designated container used for packing. Constraint 5.4 checks the condition that no two rectangles overlap each other. Here, we have considered the generality that all dimensions are integer. Constraint 5.5 indicates whether a rectangle is placed or not. The objective is to place all the smaller dimension rectangles in the given container such that occupancy is maximized and no constraints are violated. Thus, a design layout is feasible, if it satisfies the above constraints.

5.2.2 Biased Random Key Based Genetic Algorithm

To solve sequential problems in combinatorial optimization, Bean (1994) [122] introduced random key or random key genetic algorithm. Thereafter, the approach is modified by many researchers to handle a large class of problems. Goncalves *et al.* (2011) [114] proposed biased random key genetic algorithm (BRKGA), which is a variant of random key and has a different approach for selecting the parents for mating. In biased random key approach, each chromosome is represented by randomly generated vectors having real values lying in between the range [0,1]. The values obtained are sorted to obtain the chromosome sequence, which is the rectangular block placement sequence in our case as shown in Figure 5.2. The initial population consists of p vector of m random keys, where p is the number of population in each generation G and m are the number of rectangular blocks to be placed. The population is partitioned into two sets: elite (e) and non-elite. For each population, the fitness value for each individual is computed. The elite is a small group containing individual with maximum fitness values. In order to evolve population for the next generation, the elite population is copied as it. The remaining p-e population is evolved from performing crossover between elite and non-elite chromosomes. During the mating process, one of the parents selected is biased to have higher fitness value as compared to other. As the number of elites is less, which indicates that one parent can produce more than one offspring. Mutants are introduced into the population to allow escape from local optima. The most promising feature of BRKGA is separation between dependent and independent algorithm modules. As, in standard GA operators like crossover and mutation vary depending on the nature of a problem, however, BRKGA is not much dependent on crossover operator. The approach is competitive and suitable to tackle complex combinatorial optimization problems with least user affords.

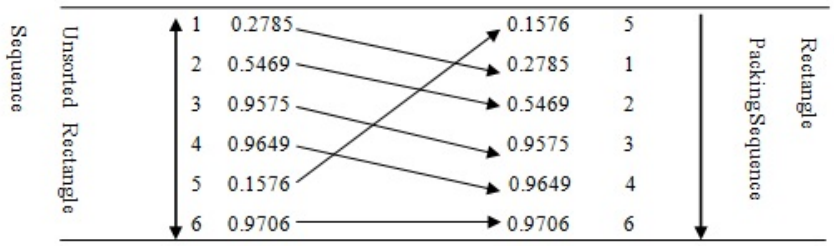


Figure 5.2: Biased random key sequencing.

5.3 Two Dimensional Placement Heuristic

This section introduces a new placement heuristic algorithm for the 2D-SPP.

The algorithm inspiration is drawn from best fit approach and the general methodology, but with many significant differences. So far, the existing heuristic and metaheuristic approaches require a pre-processing on the available set of rectangles to select the best one amongst them that can fill the current gap exactly. But, in the proposed approach we follow a simple technique, where the rectangles are placed in the sequential order according to the designed placement policies into an appropriate empty rectangular block. Thus, it avoids the computational overhead of pre-processing and selection of the best fit rectangle. In many heuristic algorithms the solution is determined by considering the rectangle in sorted order of their width. This sorting operation adds complexity to these algorithms. At each iteration, a suitable gap needs to be found, which is obtained by searching the current container status and then, choosing an appropriate rectangle from the remaining unplaced rectangles input list to find the best match. The placement approach starts initially with an empty container considered as coordinate system with the left most corner coordinate as (0,0), which is also termed as origin. Whenever a new rectangle is placed, the coordinate points are available for placement changes. A list is maintained to keep track of possible placement positions. Whenever a rectangle is placed at any location that particular coordinate position is removed from the list. The newly created placement position coordinates is inserted onto the list. This allows the subsequent rectangles to be placed on top or around the newly placed rectangle. For every rectangle placed, a number of empty rectangular block regions are generated. However, many amongst them are merged to form a larger rectangular block region to accommodate larger rectangle. The complexity of region update procedure is $O\ (n)$ where n is the number of rectangles already placed.

5.3.1 Placement Approach

The idea of the placement approach is to consider the given container to be divided into x and y coordinates. The limit range of x-coordinate is from 0 to Width (W) and y-coordinate is from 0 to Height (H), where theoretically H is considered to be infinite height. However, practically we have set an upper limit for H based on the calculated optimal height values, which depends on the type of rectangle to be packed and the width of the given container. The placement of each rectangle results in the creation of new empty rectangular block space.

The creation of empty space for rectangle placement is illustrated in Figure 5.3. Each time a new rectangle is to be placed, one of the best suited empty rectangle is selected from the available ones. The selection of empty rectangle is governed based on the following constraints:

1. Area of the rectangle to be placed is less than or equal to the empty rectangle area

2. Amongst the empty rectangle fulfilling the premier first constraint the one is chosen with lowest y-coordinate

3. The third governing criteria in the selection that the first constraint must be satisfied and there exist say more than one empty rectangle space with same minimum y-coordinate. Amongst all, the approach selects the one with least x-coordinate.

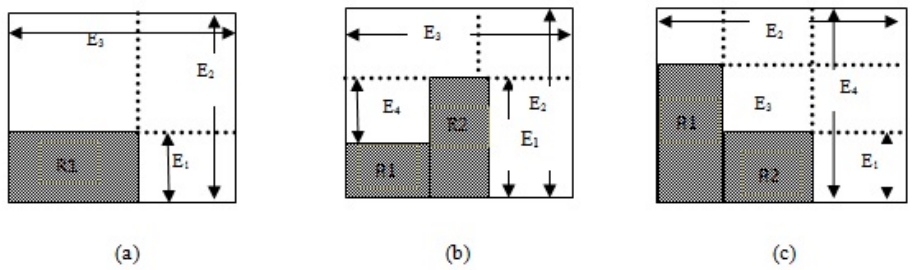


Figure 5.3: Creation of empty rectangle blocks.

5.3.2 Empty Block Creation

As the placement of the rectangle determines the empty block creation, we in this subsection, discuss all possible cases for placement and their corresponding possible empty block creations.

Case-1: For Figure 5.3 (a) the placement of initial rectangle in the container results in the number of empty block creation. E_1 denotes the block space of height equal to the placed rectangle height, and width equal to W-w$'$, where w$'$ is placed rectangle width. This creation of empty block may seem to be not essential because of larger block space, *i.e.*, E_2 creation, but such blocks are required when we discuss the general cases. Another empty block E_3 is created for the empty area above the placed rectangle. In the initial case, its width is W that off the container and height h$'$ equal to H-h$'$, where h$'$ is the placed rectangle height.

Case-2: In Figure 5.3 (b) it shows how the placement of next rectangle results in the creation of new or merging/removal of old rectangles. This case happens when R_2 height is more than that of already places R_1. The creation of E_1, E_2 and E_3 are same as discussed in Case-1, and E_4 is the left section of empty block created. Creation of new E_1 will remove the earlier block formed by initial placing of R_1. The diagonal coordinates of E_4 block will be as follows: x-coordinate as (x_1 coordinate of R_1, y_2 coordinate of R_1) and the y-coordinate as (x_1 coordinate of R_2, y_2 coordinate of R_2).

Case-3: For Figure 5.3 (c) the third placement, case where the height of R_2 is less than that of already places R_1. In this case the new creation will replace most of the already existing empty rectangle.

Thus, it is required to keep track of the creation of empty block, as each iteration results in the formation of new as well as removal of some of the existing empty blocks. The removed rectangles are basically either those, which overlap with the newly placed rectangle or the one, in which the rectangle will be placed. We designed a *MergeRectangle* routine that combines the small rectangle formed during placement into a larger one. This routine keeps a check on the growing number and stacking of empty block rectangles. The summary of the proposed strategy is presented in Figure 5.4.

Algorithm 4 Genetic Placement Approach (GPA) Algorithm

```
Input: a set of rectangles to be placed (R), Container(C)
Output: placement pattern for the set of rectangles, optimum height
repeat
    if not (initial generation) then
        Select 1/8 of the fittest offspring for next generation
    else
        Generate random initial population
    end if
    repeat
        call MPX_CROSSOVER (parent1, parent2)
        Use of placement approach to obtain sequence for packing of rectangle
        call Placement Position Search Strategy
        Creation of new empty block
        call MergeRectangle routine
        Use novel function to compute fitness of population
    until number of population
until number of generation
```

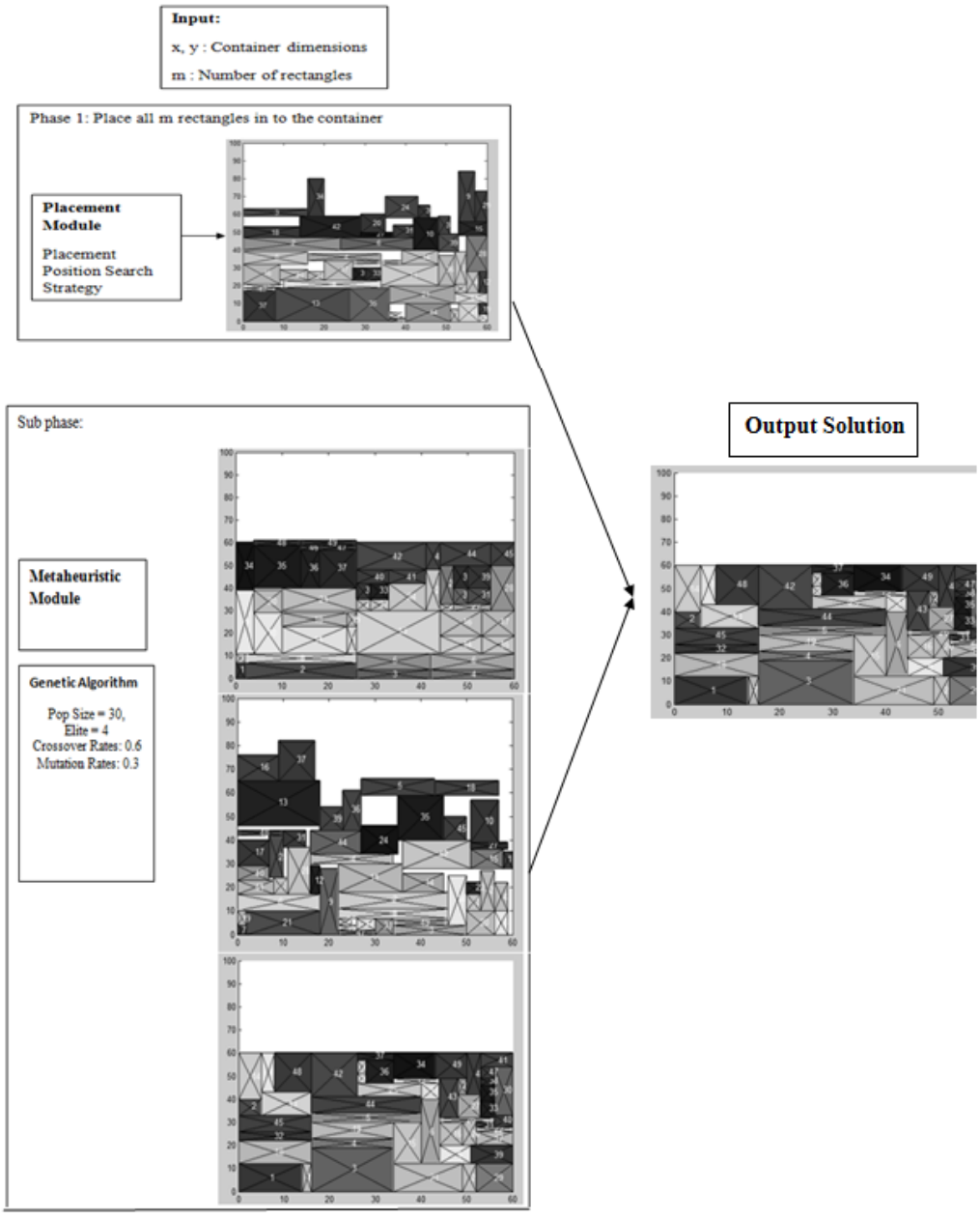


Figure 5.4: Summary of proposed strategy.

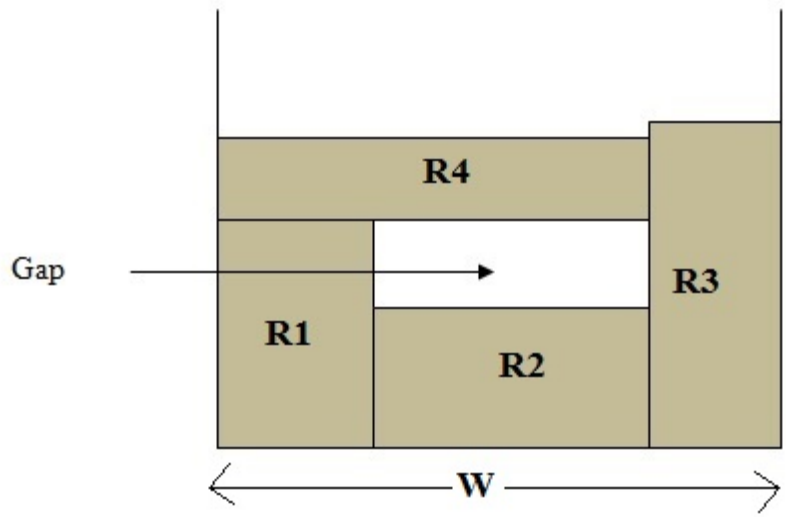


Figure 5.5: Strip packing with gap.

5.3.3 The Placement Position Search Strategy

The most common placement approach used for packing two dimensional objects is the Bottom Left (BL) heuristic. It is a very native method that involves placing the rectangle sequentially from the input list to the bottom-leftmost position in the given container. A major limitation of this approach is that gaps may occur at number of places, and the subsequent rectangles may not be placed. The scenario is shown in Figure 5.5. The best solution to the problem is to keep track of the gap and assign priority based on increasing distance from the leftmost coordinate (0,0) with lowest assigned the highest priority. The placement of each rectangle is checked for previously placed rectangles for collision. This approach has certainly improved over the BL approach. The proposed Placement Position Search Strategy (PPSS) is an extension of BL approach. It determines the best placement position for a given rectangle by comparing the x-coordinate of all empty rectangle blocks formed. It ensures that the new rectangle is placed at the bottom left most available corner. The PPSS method processes, each rectangle as input to be placed, and also has the list of current available empty blocks. The method is independent of any kind of intelligent reordering. The pseudocode for the PPSS algorithm is provided in Algorithm 5. The *getAllEmptySpaces* method returns all the usable empty block spaces created by the placement of the rectangle. The method *fits* makes a dimension check to ensure whether the rectangle can be placed in the vacant space or not, *i.e.*, make sure that the rectangle is placed without overlaps. If a suitable empty space is found, then is marked as invalid for another placement.

Algorithm 5 Placement Position Search Strategy

Input: Given an unplaced rectangle r, a container C, and empty rectangle coordinate E
Output: Rectangle r with associated placement position P
P $\leftarrow \emptyset$, $x_{min} \leftarrow \infty$
E $\leftarrow$ *getAllEmptySpaces (C, E)*
placed $\leftarrow$ false
for i=1:E **do**
  if (*placed* == false) and *fits*(r_i, *E*) **then**
    if $E_{min} \leq x_{min}$ **then**
      $x_{min} \leftarrow E_{min}$, index $\leftarrow$ i
    end if
  end if
end for
place r_i and add placement information to P
placed $\leftarrow$ true
return P

5.3.4 Merge Rectangle Routine

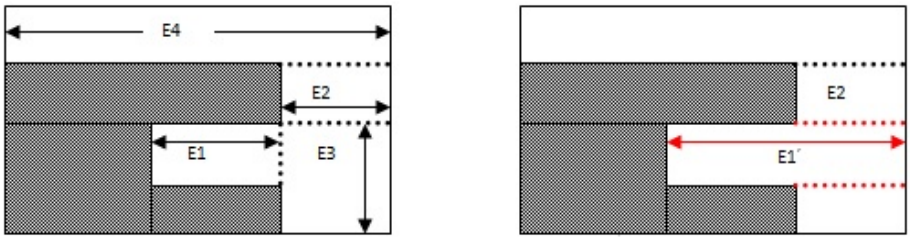


Figure 5.6: Merging empty rectangular block (x-coordinate).

The merge routine deals with two major cases. Let us consider the that case we have two empty rectangular block spaces say E_1 and E_2 with diagonal coordinates (x_{11}, y_{11}), (x_{12}, y_{12}) and (x_{21}, y_{21}), (x_{22}, y_{22}). The cases are:

Case-1: Combine the rectangles along the x-axis as in Figure 5.6, in which the difference of x_2-coordinate of first and the x_1-coordinate of the second is zero, *i.e.*, (x_{12} x_{21}), or vise versa. The new empty block formed (E_1) as shown in Figure 5.4 will have the diagonal coordinate as (x_1^*, y_1^*) and (x_2^*, y_2^*), where x_1^* = min (x_{11}, x_{21}), x_2^*=max (x_{12}, x_{22}) and y_1^* = y_{12} or y_{21}, y_2^* = min (y_{12}, y_{22}). Here, the merging property of x-coordinate holds for E_1 and E_3.

Case 2: Combine rectangles along the y axis as in Figure 5.7, having difference of y_2-coordinate of first and the y_1-coordinate of the second is zero (y_{21}-y_{21}) or vise versa. The new block coordinates will be (x_1^*, y_1^*) and (x_2^*, y_2^*) where x_1^*=x_{11} or x_{21}, x_2^*=min (x_{21}, x_{22}) and y_1^*= min (y_{11}, y_{21}), y_2^*= max (y_{12}, y_{22})

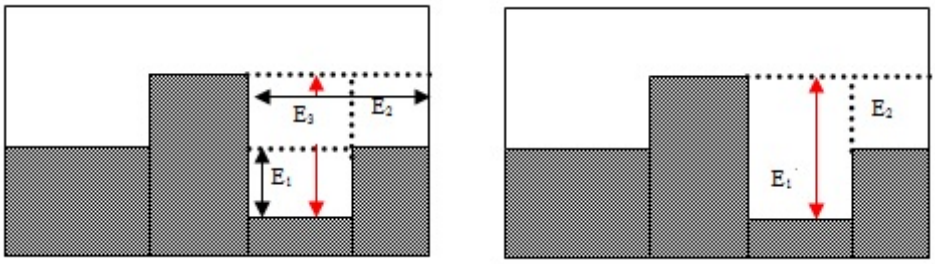


Figure 5.7: Merging empty rectangular block (y-coordinate).

The merging module removes the smaller ones by combining smaller empty spaces into large thus, facilitating the placement of the larger rectangles at lower Y-coordinate. The module also helps to overcome algorithm space limitations.

Algorithm 6 MPX_CROSSOVER (*parent1*, *parent2*)

Find two crossover point from elite population
Copy the segment from elite to the offspring
Scan the non elite population to fill remaining gene of the offspring

if (gene already present in the offspring) **then**
Move to the next gene segment Skipping already placed gene

else
Copy the gene to the offspring

end if

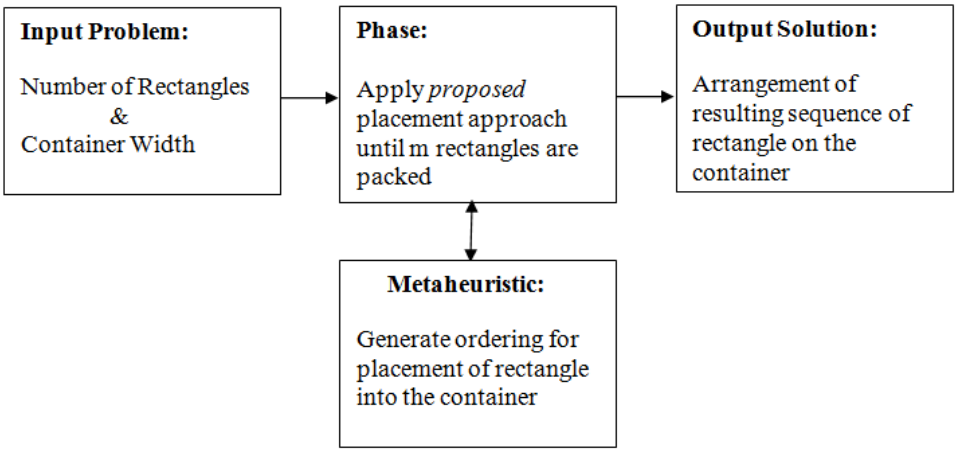


Figure 5.8: General overview of proposed hybrid approach.

5.3.5 Metaheuristic Enhancement to Strip Packing

The results obtained from the placement heuristic although represent a good quality of solution in short computation time, however, it does not guarantee to

find the optimal solution in most or all cases. Researchers finding suggest that with a hybrid heuristic/metaheuristic packing strategy, the quality of the solutions may be further improved. The main objective of implementing metaheuristic to the placement strategy is to increase solution quality while maintaining short computation time. This results in a hybrid system, which generates the placement sequence using metaheuristic where the computational intelligence is applied to determine the correct sequence resulting in minimum space lost. It follows the placement of these generated sequences as per the placement heuristic. The evolution helps to generate better and best solutions. The overview of the developed system is shown in Figure 5.8.

Chromosome Representation

Each chromosome in the population represents a set of parameters to any problem that GA is trying to solve. The initial population in the proposed approach is generated by random strings of real numbers in the interval range of [0,1] having the same length, where length represents the chromosome size. The sizeof chromosome is the number of rectangles to be placed into the given container. This evolutionary strategy was proposed by Bean (1994) [122] and also used by Goncalves (2007) [114]. The obtained random sequence is then sorted in the increasing order to generate the placement sequence for the strip.

5.3.6 Crossover Function

The placement sequence after sorting is further modified by the crossover operation. The evolution cycle for GA is enhanced by two fundamental operators: crossover and mutation. Here, we have considered biased random key method, which integrates mutation operator. The approaches discussed by Gonalves (2007) [89], Goncalves *et al.* (2011) [114], highlight that rather than performing mutation on the entire population, certain mutants with specific mutation rates should be introduced in the non-elite section of the population. However, in the proposed scheme rather than introducing mutants to non-elite set, we kept the mutation rate to be very low, as the approach itself is capable of dealing with the low convergence rate problem. Thus, we preferred the existing non-elite population rather than mutants introduced for crossover. As the elite parents are more feasible to be the solution for the given GA problem, BRKGA proceeds by selecting one random elite parent and other non-elite. Such a selection ensures that the new generated offspring would carry some features from elite parent. As the number of individuals in elite population is less as compared to non-elite, thus, the repetition of parents is allowed. That is one elite parent can

generate multiple offsprings.

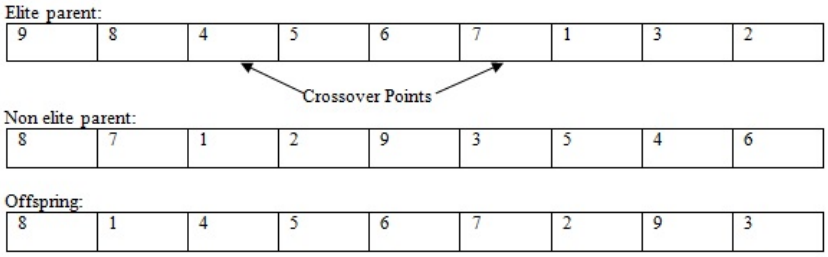


Figure 5.9: Maximal preservative crossover

In difference from the original approach, we apply Maximal Preservative Crossover (MPX) over the parameterized uniform crossover used for the mating process of the chromosome. MPX is preferable as it produces offspring by directly copying a small segment from the elite parent to the offspring. In this crossover approach, two random crossover points are generated. These crossover point decide the range of segment that would be copied in the offspring from the elite parent. The remaining offspring is generated by copying the gene from the second parent provided that the gene is not already present in the offspring. The iteration steps are presented in *MPX_CROSSOVER* Algorithm.

An example of MPX is illustrated in Figure 5.9, while copying gene from second parent to offspring gene value 7 is already present in the offspring. Thus, it is skipped and the next gene is evaluated and placed, if not already present in the offspring. As, random key vector is used to encode the solution, and resulting offspring from mating is always valid that can be decoded back as a solution to the combinatorial optimization problem. When the next population is complete with p individuals, the fitness values are computed for all of the newly created random-key vectors and then the population are partitioned into elite and non-elite individuals to start a new generation.

5.3.7 Fitness Evaluation

The quality of any evolved solution is judged by its corresponding fitness value. These values help in selection process to select the fitter individual to be the parent, which can share their properties to evolve better offsprings for the future generations. Since, the goal is to minimize the packing height of the strip in a given container, thus, the fitness evaluation of any individual is based on the height parameter. Hence, we calculated individual score for each placed rectangle, which is determined by the vacant space left in packing between the placed rectangles. The overall score of the layout would be the summation of score for each individual placed rectangle. This layout score is considered in the evaluation of the fitness function. The utilization factor or score is given by

$$Score(r_i) = \frac{w_i h_i}{WH - \sum_{i \neq j, j \in B} w_j h_j} \tag{5.7}$$

Where r_i is the rectangle with width w_i and height h_i to be placed in a container. W denotes the width of the container and H represents Height. To determine the score, we set an approximate height of the container to a value slightly higher than the computed optimal height. This is done to tackle the worst cases as the ordering is random in nature. The optimal height for placement is calculated as summation of area for all the rectangles to be placed divided by the width of the container as given by Equation 5.8. Here, B is the number of rectangles already placed. The optimal/expected height is calculated as

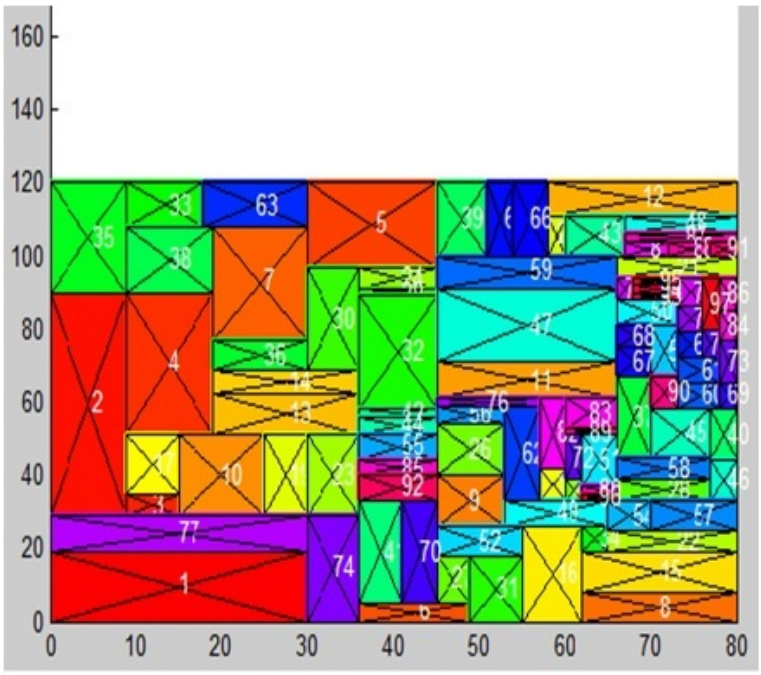
$$Optimum\_Height = \frac{\sum_i Area(r_j)}{W} \tag{5.8}$$

The second parameter that we use for fitness evaluation is the best height (BH). This parameter indicates the current height achieved by the layout. The fitness function must consider both the computing aspects, which are the score and BH. Thus, utilization factor or score is not the only criteria for fitness value assignment. Hence, the fitness function for the selected rectangle layout sequence is evaluated as

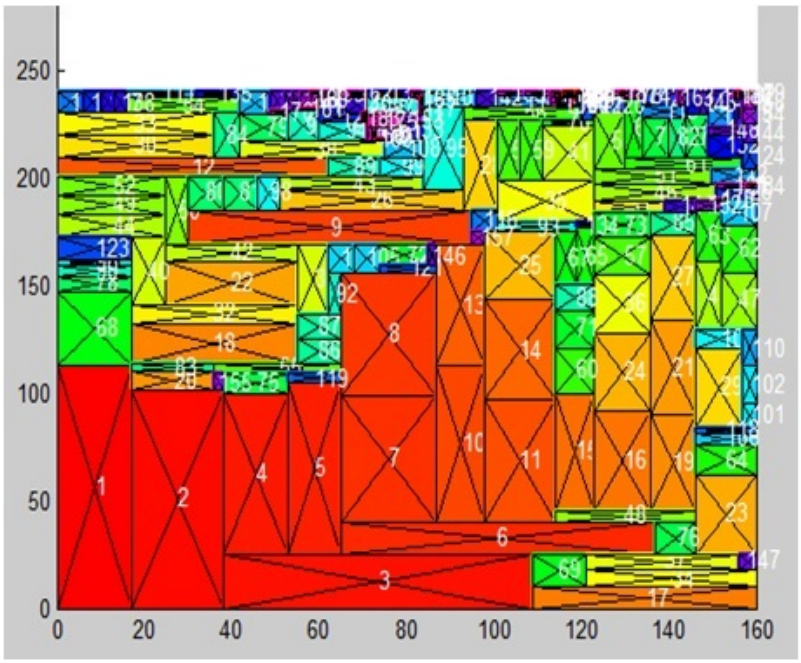
$$Fitness(j) = Score(j) \cdot BH(j) \quad \triangleright\, j^{th} rectangle\ sequence\ pattern \tag{5.9}$$

5.4 Improvement on Existing Genetic Algorithm Approach

The genetic algorithm based approaches like Hopper and Turton [65], Bortfeldt [86], Gonalves [81] show that the performance of the algorithm basically depends on the evolution of the rectangle placement sequence. But, in comparison to other approaches, we do not perform any pre-processing like sorting of rectangles based on maximum area, perimeter, etc. The selected MPX-crossover operation helps us to find a different ordering sequence to determine a better solution, and thus, improves the convergence rate of the algorithm towards optimum. The Genetic based Placement Approach (GPA) has two main computation stages, one for a number of generations which involves initial population, crossover and fitness evaluation and another for the empty block creation, selection and removal. For each population, the placed rectangle block is governed



(a) C6.3 instance



(b) C7.3 instance

Figure 5.10: Solution for dataset instances.

by simple placement policy, *i.e.*, PPSS. The crossover operation helps us to maintain the ordering sequence, thus, improves the height in each generation. Each population is assigned a fitness value that helps in further population evolution. Another improvement shown over existing GA algorithm is the reduced computation time to obtain the optimal solution and the ability to handle large instances. In the next section, we compare our results with the latest algorithms on benchmark datasets. Figure 5.10 shows the packing for data instances of Hopper and Turton.

5.5 Comparative Evaluation of the Packing Strategy

The proposed model is implemented on 2.2 GHz Intel core duo CPU processor with 2GB DDR3 primary memory. The programming platform used to implement the approach is Java, a multi threaded programming language. The proposed algorithm performance is evaluated by carrying out exhaustive testing on strip packing benchmark instances, from the vast available literature. The dataset instances that are used and our algorithm's performance is reported in the subsequent sections.

5.5.1 Dataset Description

The details of datasets with known optimal height used for comparison are as follows:

1. Jakobs [78]: Two small instances J1 and J2 with number of rectangles to be placed are 25 and 50 respectively.

2. Babu & Babu [79]: A single instance with 50 rectangles to be placed.

3. Hopper and Turton [65]: Test set C contains 21 instances. These instances are divided into seven categories, each category contains three instances with number of rectangles to be packed ranging from 16 to 197. These groups are categorized on the basis of similarity in achieving optimum height and container width.

4. Wang and Valenzela [123]: The dataset contains real values of the rectangle dimensions, which are rounded down to an integer by multiplying by 10. In some cases like, while comparing with the GA based approach, we have considered no rounding up for odd dimensions. The dataset discusses two different types of instances, which are grouped in two categories of Nice

Table 5.1: Computational results for Jakobs, Babu & Babu and C test instances.

Dataset		Instance		BL-DH	BLF-DH	BF	FH	BBF	BBFM	GPA	Time (Sec.)
	m	W	Optimum								
J1	25	40	15	-	-	-	-	16	15	**15**	30
J2	50	40	15	-	-	-	-	16	15	**15**	25
RB	50	1000	375	-	-	400	-	400	375	**375**	50
C1.1	16	20	20	23	22	21	20	21	20	**20**	23
C1.2	17	20	20	22	22	22	20	21	21	**20**	18
C1.3	16	20	20	21	21	24	21	21	21	**20**	27
C2.1	25	40	15	17	17	16	16	16	16	**15**	36
C2.2	25	40	15	26	26	16	15	17	15	16	29
C2.3	25	40	15	17	17	16	15	16	16	**15**	34
C3.1	28	60	30	33	33	32	31	32	30	**30**	40
C3.2	29	60	30	33	32	34	31	33	31	**30**	31
C3.3	28	60	30	34	34	33	32	33	32	**30**	46
C4.1	49	60	60	67	66	63	61	62	62	**60**	56
C4.2	49	60	60	68	63	62	61	63	61	**60**	59
C4.3	49	60	60	64	63	62	61	62	61	**60**	62
C5.1	73	60	90	94	94	93	91	91	91	**90**	100
C5.2	73	60	90	99	95	92	90	92	91	**90**	89
C5.3	73	60	90	97	94	93	91	92	91	**90**	121
C6.1	97	80	120	130	126	123	121	123	121	**120**	220
C6.2	97	80	120	130	123	122	121	123	122	**120**	245
C6.3	97	80	120	131	128	124	121	123	121	**120**	231
C7.1	196	160	240	252	249	247	241	243	242	241	400
C7.2	197	160	240	264	247	244	241	242	242	**240**	452
C7.3	196	160	240	257	249	245	241	243	241	**240**	470

and Path. Nice contains data with similar dimensions, on the other hand, Path has a rectangle with varying dimensions. The rectangle range of both the datasets is from 25 to 1000.

5. Burke *et al.* [91]: Burke generated test instances to test BF algorithm. The number of rectangles to be placed are subsequently increased from small as 10, (*i.e.*, N1) too large as 3152, (*i.e.*, N13).

Table 5.2: Computational result for test instance by Burke.

Class	Instance			BF	BF+SA	FH	BBF	BBFM	GPA	Time (Sec.)
	m	W	Optimum							
N1	10	40	40	45	40	40	40	40	40	19
N2	20	30	50	53	50	52	52	50	50	15
N3	30	40	40	52	51	51	52	52	51	29
N4	40	80	80	83	82	83	82	82	80	70
N5	50	100	100	105	103	102	103	102	101	62
N6	60	50	100	103	102	101	102	101	100	90
N7	70	80	100	107	104	102	106	105	103	124
N8	80	100	80	84	82	81	82	81	80	96
N9	100	50	150	152	152	151	152	151	150	340
N10	200	70	150	152	152	151	151	151	151	589
N11	300	70	150	152	153	151	151	150	150	545
N12	500	100	300	306	306	301	302	302	300	842
N13	3152	640	960	964	964	960	964	960	960	1200

5.5.2 Comparison with Other Metaheurisitic Approaches

Table 5.1 reports the computational results for three different data instances for multiple experimental runs. The number of runs are varied as per the data instance size for small sets like J1,J2, RB and C1.1 to C6.3 the number of runs were 3 to 5. On the other hand for C7 instances 7-10 experimental runs were carried out. It shows a comparison table, where the performance of the proposed approach is compared to the well known existing algorithms as discussed in the literature section like BL-DH, BLF-DH, BF, FH, BBF, BBFM. Here, for all the experiments, m remains variable based on the number of rectangles to be packed. The best solution of ten runs is given for all the problems from the literature. It shows, on Jakobs and Babu & Babu data instances, the approach is at par with BBFM, both giving the optimum results on all instances. In case, of C dataset instances apart from C2.2 and C7.1 that are having a slight gap of one unit from optimal value, the results on all other remaining instances are optimum. The best solution is highlighted in bold font type in the table. For

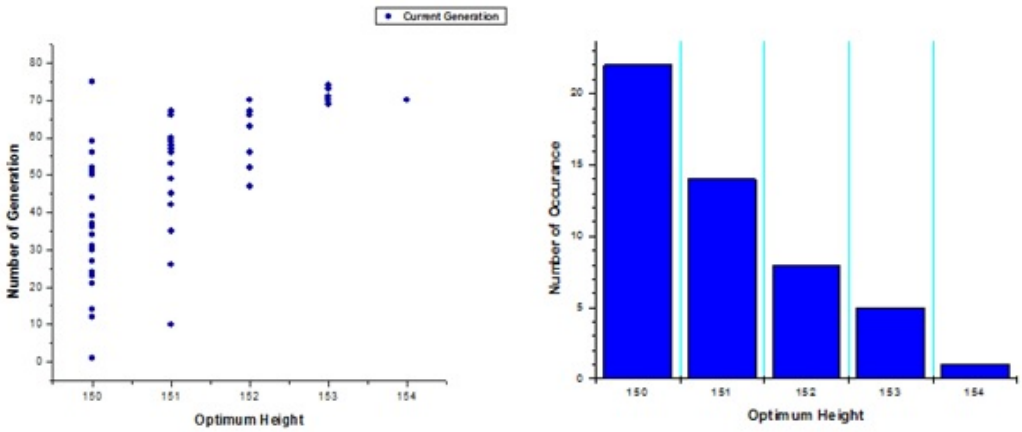


Figure 5.11: Study of (a) number of generation (b) frequency of occurrence of different values of best fitness in 50 runs of the algorithm.

most of the instances, it is observed that applying computation intelligence to generate placement pattern achieves a good solution.

Table 5.2 shows the results for the Burke N1-N13 dataset for multiple experimental runs. In case of smaller instances less than 100 like, N1-N8 the results are reported for 3-5 runs, whereas for N9-N13 the runs were 7-15. The proposed approach is compared to 13 instances with the well known BF approach and other methods. The BF+SA method is considered as the best algorithm amongst all the variant of BF+metaheuristic, Fast Heuristic (FH) better on N large instances, BBF and its modified version (BBFM). The table analyzes the result in all 13 instances with respect to the solution found and GPA computation time. The BF and BF+SA give nearly the same performance on many instances, whereas FH, BBF and BBFM improve the solution, which is furthermore improved by our approach. Not only for small instances, but also the algorithm performs comparatively better for large instances like N12 and N13. The algorithm achieves improved results on nearly three-fourth of the instances and are competitive on the remaining instances. The nature of the dataset is the major reason for better performance of metaheuristic on these instances in comparison to others. The dataset contains few large size rectangles and a more number of small rectangles. Thus, the intermediate vacant space, which is created due to the placement of large rectangles can be filled with smaller ones resulting in a good layout. The proposed approach utilizes the property of the instances to achieve better solutions.

In general, it is observed that the hybrid metaheuristic approach achieves better results on most of the instances with reasonable computation time. The main cause of failure for genetic algorithm is that it suffers due to loss of genetic structure during the evolution process. The two placement sequences may appear, similar in structure, but differ in solution quality. This happens because

in other placement strategies, the placement of successive rectangle depends on the already placed rectangle. Moreover, another concern being during evolution is that the sequence may change as the recombination operator may result in distortion of the structure. We have addressed both these issues, as in the first case, our placement is independent of sorting and searching criteria to determine the best fit rectangle. To avoid the distortion in the evolution of rectangle placement sequence, we have used MPX_CROSSOVER that preserves the ordering (good feature) of rectangles from parent to child, as it occurs at the same position. It is observed that almost all methods outperform the best-fit placement approach. However, overall GPA produces the best results. The number of generations involved in processing large instances is shown in Figure 5.11. We also consider non-zero waste for data instances of Nice and Path by Valenzuela *et al.* [123]. The results of comparing the GPA on these non-zero trim loss over heuristic approach is reported in Table 5.3 for multiple runs.

Table 5.3: Computational results for Nice and Path test instances.

Class	Instance			BF	BBF	BBFM	GPA
	m	W	Optimum				
Nice1	125	1000	1000	1074	1083	1069	1070
Nice2	50	1000	1000	1085	1079	1068	1000
Nice3	100	1000	1000	1070	1067	1063	1040
Nice4	200	1000	1000	1053	1053	1038	1039
Nice5	500	1000	1000	1035	1033	1024	1000
Nice6	1000	1000	1000	1037	1037	1012	1006
Path1	25	1000	1000	1101	1091	1091	1091
Path2	50	1000	1000	1138	1074	1074	1074
Path3	100	1000	1000	1073	1073	1073	1070
Path4	200	1000	1000	1041	1053	1053	1030
Path5	500	1000	1000	1037	1032	1031	1031
Path6	1000	1000	1000	1028	1028	1026	1008

It is observed that amongst 12 instances reported, the GPA finds the optimal solution for 7 instances. GPA finds the smallest height and outperforms all the existing heuristic approaches within reasonable time. The GPA performed worse

Table 5.4: Computational results for large test instances.

Class	Instance			GRASP	SVC	FH	GPA
	m	W	Optimum				
NiceL1	1000	100	100.1	102.2	101.5	100.9	100.3
NiceL2	2000	100	100.1	101.5	100.7	100.5	100.2
NiceL3	5000	100	100.1	101	100.4	100.2	100.0
PathL1	1000	100	100.1	101.9	101.2	100.7	100.6
PathL2	2000	100	100.1	101.5	101	100.2	100.0
PathL3	5000	100	99.9	101	100.2	100	100.0

only for one instance, which is Path 5. The analysis of this worst case behavior for GPA was observed due to a large number of small scale instances, and quite less variation in dimensions. The algorithm behavior also depends on the generated sequence patterns. However, on large instances, the approach is able to find optimal solutions, considering the large non-zero trim loss dataset with real dimensions. Table 5.4 shows the computational results on NiceL1 - NiceL3 and PathL1 - PathL3 for FH, GRASP and SVC for 10-15 experimental runs. As reported, the GPA efficiently solves and obtains an acceptable performance.

Table 5.5 presents a comparison against existing approaches in term of %-gap where it is defined as the ratio:

$$\%-gap = ((ObtainedSolution - OptimalHeight)/OptimalHeight) * 100 \tag{5.10}$$

The results are reported for the Hopper and Turton data instances from the literature, it also reports the total number of optimal solution found by each approach. Further, these calculations are used for the statistical analysis of the our approach. The %-gap is computed against different approaches, where BF is being used in coupling with other techniques. The BF+SA is one of the most used heuristic approaches that gave comparable results than BS+GA and BS+TS. The non-systematic search techniques like squeaky wheel optimization by Burke *et al.* (2011), SVC (SubKP), (reactive GRASP), stochastic approach like ISA, and IDBS combined with tabu search are also used for the comparison. IDBS is state-of-the-art outperforming the all algorithm. Our approach finds the optimal solution in 19 out of 21 cases, and is far better than existing and at par with IDBS in most of the cases. In instances like C4, C5, C6 (all instances), C7.2, and C7.3, the GPA is able to find the optimum height whereas other heuristic approaches fail to achieve the same. In other cases, the solution is near to optimal and misses only by the single unit. The GPA stands second in finding the number of optimal solutions, (*i.e.*, 19) after IDBS, which finds optimal results for all data instances.

5.5.3 Statistical Analysis for Heuristic Approach

Statistical analysis is one of the powerful tools to evaluate algorithm implementation and heuristic. In order to validate the results of the proposed algorithm, the ANOVA is applied to check if the observed differences with the existing algorithms are statistically significant. The analysis helps to identify which algorithm scenario performs the best. The ANOVA works by comparing the variation between groups to the variation within groups. If the ANOVA concludes with the fact that there exists a difference in group mean to identify

Table 5.5: Computational results for large test instances.

Instance	BF	BBF	BF+SA	FH	BBFM	SWP	SVC	GRASP	ISA	IDBS	GPA
C1.1	5.00	5.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
C1.2	10.00	5.00	0.00	0.00	5.00	5.00	5.00	0.00	0.00	0.00	0.00
C1.3	20.00	5.00	0.00	5.00	5.00	0.00	0.00	0.00	0.00	0.00	0.00
C2.1	6.67	6.67	6.67	6.67	6.67	6.67	0.00	0.00	0.00	0.00	0.00
C2.2	6.67	13.33	6.67	0.00	0.00	0.00	0.00	0.00	0.00	0.00	6.67
C2.3	6.67	6.67	6.67	0.00	6.67	0.00	0.00	0.00	0.00	0.00	0.00
C3.1	6.67	6.67	3.33	3.33	0.00	0.00	0.00	0.00	0.00	0.00	0.00
C3.2	13.33	10.00	3.33	3.33	3.33	3.33	3.33	3.33	3.33	0.00	0.00
C3.3	10.00	10.00	3.33	6.67	6.67	0.00	0.00	0.00	0.00	0.00	0.00
C4.1	5.00	3.33	1.67	1.67	3.33	1.67	1.67	1.67	1.67	0.00	0.00
C4.2	3.33	5.00	1.67	1.67	1.67	1.67	1.67	1.67	1.67	0.00	0.00
C4.3	3.33	3.33	1.67	1.67	1.67	1.67	1.67	1.67	1.67	0.00	0.00
C5.1	3.33	1.11	1.11	1.11	1.11	1.11	1.11	1.11	1.11	0.00	0.00
C5.2	2.22	2.22	1.11	0.00	1.11	1.11	1.11	1.11	1.11	0.00	0.00
C5.3	3.33	2.22	2.22	1.11	1.11	1.11	1.11	1.11	1.11	0.00	0.00
C6.1	2.50	2.50	1.67	0.83	0.83	1.67	0.83	1.67	0.83	0.00	0.00
C6.2	1.67	2.50	0.83	0.83	1.67	0.83	0.83	1.67	0.83	0.00	0.00
C6.3	3.33	2.50	1.67	0.83	0.83	1.67	0.83	1.67	0.83	0.00	0.00
C7.1	2.92	1.25	1.67	0.42	0.83	1.25	0.83	1.67	0.83	0.00	0.42
C7.2	1.67	0.83	1.67	0.42	0.83	0.83	0.83	1.25	0.42	0.00	0.00
C7.3	2.08	1.25	2.08	0.42	0.83	1.25	0.83	1.25	0.83	0.00	0.00
# of optimum	0	0	3	5	3	6	7	8	8	21	19

the difference, Tukey multiple comparison test is used. It compares between each pair of means for the multiple testing. Tukey's multiple comparison test is one of several tests that can be used to determine which means, amongst a set of means differ from the rest. Tukey's multiple comparison test is also called Tukey's honestly significant difference test or Tukey's HSD. All statistical analysis presented in this dissertation were carried out at a 5% level of significance. The tool used to carry out the statistical analysis is statistics open for all (SOFA). It is a free open source, user friendly, statistics analysis and reporting package that can handle complex data.

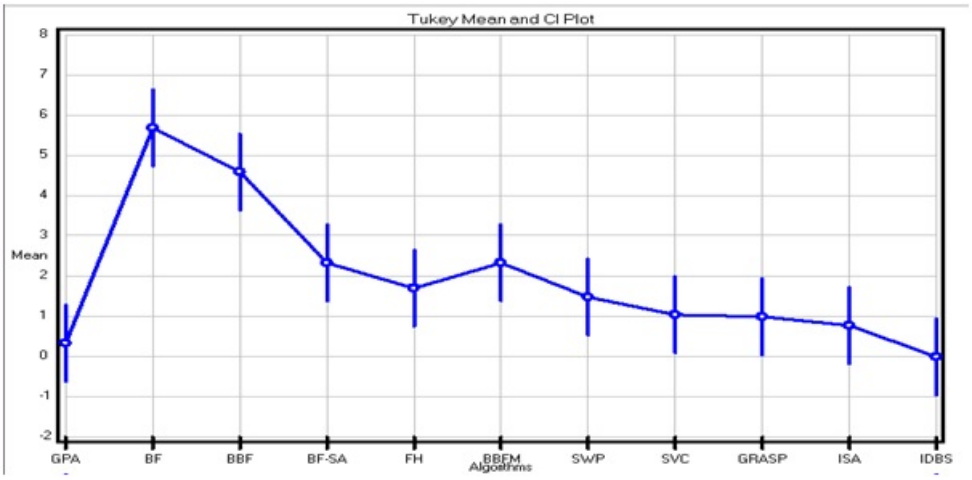


Figure 5.12: Means plot and Tukey's confidence intervals (CI) for the evaluated algorithms.

Figure 5.12 shows the mean plot and Tukey's CI for the evaluated algorithm where encircled are the mean values and line is used for showing the respective confidence interval. The statistical analysis reveals that the proposed approach significantly differs from almost all the approaches and is competitive with ISA and IDBS. The computational results and statistical analysis show the acceptable performance of the proposed GPA algorithm.

5.5.4 Comparison with Genetic Based Approaches

Our metaheuristic approach is also compared to similar genetic based approach based on the relative difference from the optimal solution and a statistical analysis is carried to justify the performance of the GPA. Hopper & Turton investigated SA and GAs metaheuristics approach in combination with the placement strategies BL and BLF like GA+BLF, SA+BLF, and NE+BLF. Table 5.6 reports the comparison against such algorithm along with SPGAL, Iori *et al.* [124] and Goncalves approaches. The SPGAL approach is better and clearly dominates the comparison against the other earlier approaches with the %-gap of

Table 5.6: Comparison of %-gap for genetic based approaches for Hopper & Turton test instances.

Instance	Hopper & Turton			Iori *et al.*	SPGAL	HPA	GPA
	GA+BLF	NE+BLF	SA+BLF				
C1	4.0	5.0	4.0	1.59	1.7	0	0
C2	7.0	7.0	6.0	2.08	0.9	0	0
C3	5.0	4.0	5.0	2.15	2.2	0.53	2.2
C4	3.0	4.0	3.0	4.75	1.4	0.70	0
C5	4.0	4.0	3.0	3.92	0.0	0.33	0
C6	4.0	4.0	3.0	4.0	0.7	0.42	0
C7	5.0	5.0	4.0	-	0.5	0.66	0.14
Average	4.6	4.7	4.0	3.08	1.0	0.38	0.33

just a unit. The HPA gives a better performance, but, the result reported is the average result on the basis on a number of runs, (*i.e.*, 10) for each instance. The performance of the GPA is better even without averaging the number of runs.

5.5.5 Statistical Analysis

The statistical analysis is also performed to evaluate against different meta-heuristic approaches as shown in Figure 5.13. The GPA shows a vast deviation from the mentioned approaches as it is able to find optimal result in most of the cases. The plotting also shows that HPA and GPA are statistically not much different.

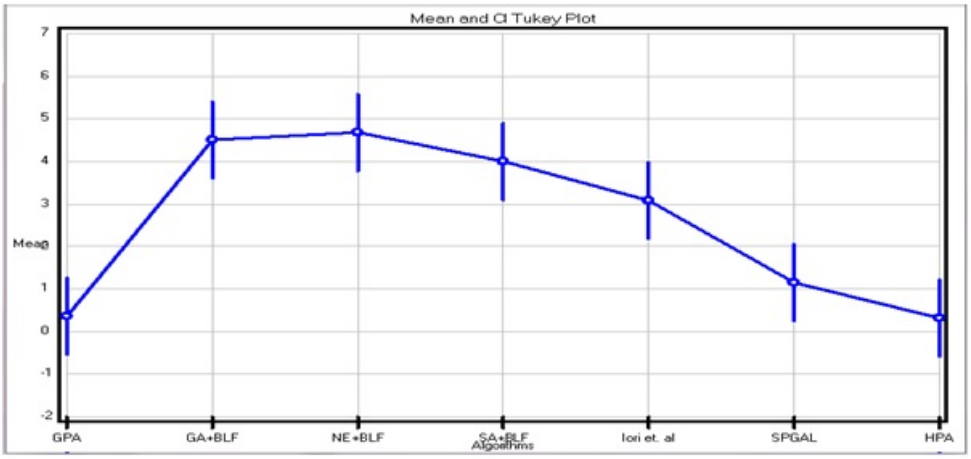


Figure 5.13: Means plot and Tukey confidence intervals (CI) for the metaheuristic algorithms.

5.5.6 Algorithm Complexity

The analysis of algorithms is considered for both space and time. Linear storage space is required for storing the parameters corresponding to the rectangle being

packed and current status of the designed layout. The algorithm is analyzed for all the cases. In the best case, the initial generated placement sequence is the required one, thus, the complexity of the algorithm is O (1). The algorithm does not use level oriented packing. The average and worst case behavior where the population is evolved to find the optimal pattern sequence in O (n^2).

5.6 Chapter Summary

This chapter presents an efficient two dimensional hybrid placement strategy, which tackles the 2D-SPP in an effective manner. It has been shown that two dimensional best fit is a suitably powerful and efficient method of packing rectangle within a container. With some metaheuristic enhancements (coupled with a packing strategy) to this solution, the quality can be improved even further in a very reasonable amount of execution time. In this chapter, a new hybrid approaches has been proposed, which combines the relative strengths of both the placement heuristic and the metaheuristic, for two dimensional orthogonal packing. The proposed methodology is easily extended to include other constraint like rotation. The hybrid method can quickly and significantly improve upon layouts produced by other techniques from the literature and it can achieve good solutions that are less than 1% over optimum in some cases. The time taken to reach suitably high solution quality with this strategy is very reasonable on realistically sized instances and could be implemented in many industrial settings with relative ease.

Chapter 6

Hyper-heuristic Approach to 2D-Strip Packing

6.1 Introduction

Consumerism brought by industrialization is now an integral part of the economy. Economic growth depends on the continued supply of new goods with the best utilization of available resources based on least waste production policy to reduce waste recycling. Thus, it motivates the need of effective automation techniques that can be beneficial to different industries. These techniques are not limited to textile, but can be extended to other industries like pallet loading, paper and pulp, wood, and textile to improve their economical saving. Textile and fabrication industries also deal with similar problems, where irregular shape cloth cutting is done to meet the customer demand. The implication of these problems is not limited to industries, but also in resource scheduling where available resources are allocated to the client as per the requirement or allocation of memory to the running processes.

Packing and transportation is one of the major concerns for developed as well as developing nations. The changing scenario of growing need compels to make government policies and mission for the development of science and technologies to achieve safe, economical, efficient, secure, and meet applicable regulatory requirements for packing and transportation. A numerous efforts are being made to resolve packaging and transportation issues safely, economically, and promptly. First and foremost objective for transport packaging is, it

must serve to protect goods in transit. This can be achieved by proper packing and placement of goods during transit as it can only ensure against unintended shocks, impacts or accidents of any kind. The better utilization has impact on many factors like reducing the number of vehicles in shipment, reducing the cost of damage during transit, improving the transportation cost and economy saving. All these could be achieved, if boxes are packed optimally or near optimally. Packaging decisions have a major impact on logistical productivity and efficiency. Handling efficiency in all of these situations is significantly influenced by package design, unitization, and communication characteristics. The packing problem is a type of NP-hard problem as the number of boxes increases, the solution cannot be obtained in polynomial time. This motivates the use of heuristic and metaheuristic approaches. For the packing combinatorial optimization problem with growing complexity and computational constraints, the solution using the exact approach in reasonable time seems to be infeasible.

This chapter discusses the packing issues related to two dimensional objects like packing of rectangular strips into a larger container using different approaches. The problem is addressed using a constructive greedy hyper-heuristic search technique to find better solutions. A genetic algorithm is used to evaluate the design, improve the solution and prevent it from being trapped in local minima. The proposed approach efficiently tackled the practical issue of solvability as the problem size grows, *i.e.*, the complexity with size, accuracy of the developed model, processing time etc. to find the optimum solution. The motivation behind the use of hyper-heuristic is to determine the best suited constructive heuristic that can obtain a quick solution for the new instance. The general behavior is observed that the time taken by hyper-heuristic is small as compared to metaheuristic for many problems to be solved. In some cases, hyper-heuristic can obtain better results as compared to complex metaheuristic search. The objective is to automate the design and cut down the cost.

Rest of the chapter is organized into following subsection: Section 6.2 introduces hyper-heuristic terminology. The mathematical model designed to represent this class of problem is explained in section 6.3. The proposed technique is discussed in section 6.4, it shows the consecutive subsections creation via decomposition technique. Section 6.5 presents a heuristic design approach, constructive hyper-heuristic applied to model, and the evaluation of genetic al-

gorithm. In section 6.6 experimental results are analyzed for empirical testing of the algorithm against the state-of-the-art methods from the literature, consistently in terms of computation time and occupancy. Section 6.7 gives the chapter summary.

6.2 Theoretical Background

This section discusses the hyper-heuristic approach in detail.

6.2.1 Hyper-heuristic

Hyper-heuristic refers to the search methodology or learning mechanism to solve combinatorial search by either generating or selecting the heuristic. The challenges to solve computationally hard search problems have motivated the use of hyper-heuristic approach for automating the heuristic search design helpful in solving numerous real world problems. The objective of hyper-heuristic is to design some generic approaches with a distinguish feature of operating in the search space of heuristic, and generate the solution of an acceptable level using a set of easy to implement low level heuristic. Hyper-heuristic approaches are proving to be a general problem solver approach motivating the use to obtain a high quality solution across distinct problem domains. These are competent search techniques witnessing the great success to numerous real-world applications. As these applications are huge in size, exact approaches are not inadequate, thus, heuristics are commonly preferred.

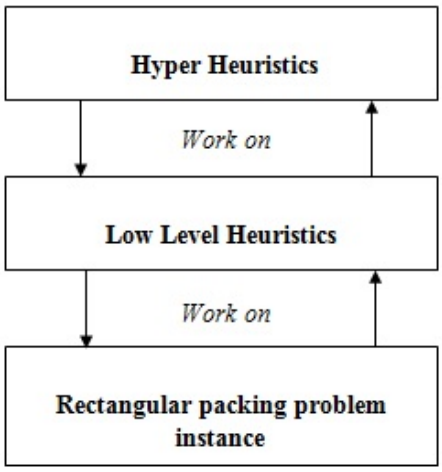


Figure 6.1: Relationship between hyper-heuristic and problem instances.

The aim of a guided hyper-GA is to make the dynamic removal and insertion of heuristics more efficient, and evolve sequences of heuristics in order to produce promising solutions more effectively. Here, we have considered a hyper-heuristic as a high level approach that takes into account a number of low level heuristic and select an appropriate one at each decision point. A general framework of hyper-heuristic is shown in Figure 6.1. Hyper-heuristics are classified into two broad categories based on the nature of heuristic search space and the feedback source during the learning process. The nature of search heuristic depends on the approach, whether it is a heuristic selection or generation, where they are further categorized either to be constructive or perturbation. Similarly, the feedback used during learning is categorized into on-line, off-line and no learning. The proposed methodology falls under the sub-class of constructive heuristic selection for hyper-heuristic methods. The constructive heuristic starts with no solution, but as the solution proceeds, iteratively they select the best ones and finally build the entire solution. The important challenge is to select the best one amongst available for the given current problem state.

6.3 Mathematical Model

The model is formulated based on simple observations, yet a compact model developed helps to find the optimal solution. It makes use of two decision variables: First, p_i stating that each i^{th} rectangular block can be placed only once in the container and the second x_{ij} indicates that i^{th} rectangle is placed at j^{th} level. Each rectangular block is represented using a quadruple (x_i, y_i, h_i, w_i), where (x_i, y_i) represents the leftmost coordinate of the placed rectangle and, h and w stands for height and width respectively.

$$p_i = \begin{cases} 1 & \text{if } \mathrm{i}^{th} \text{ block is placed in the container,} \\ 0 & \text{otherwise.} \end{cases} \tag{6.1}$$

The item packing at the level is modeled by

$$x_{ij} = \begin{cases} 1 & \text{if } \mathrm{i}^{th} \text{ block is placed at the boundary surface of level j,} \\ 0 & \text{otherwise.} \end{cases} \tag{6.2}$$

This model differs from other level based models, as in other models, each level is determined by the highest placed rectangle, the filling of the level takes place by smaller ones and finally the levels are kept in the container. However, in this case, the focus is on, width parameter rather than height. The top surface of each placed rectangle is called the level, and the associated width is termed as level width (W_i), *i.e.*, the length of the ith level as denoted by the bold lines in Figure 6.2. Each level form acts as a sub-maximization problem, where the objective is to maximize these levels width occupancy of the top surface by placement of rectangles. The decomposition of problem into a number of maximization subproblem, which helps to achieve the overall objective of container height minimization by efficient packing. The integer linear programming model for each level is as follows:

$$\max \sum_{i=1}^{m} w_i p_i \tag{6.3}$$

subjected to:

$$\sum_{i=1}^{j-1} x_{ij} + p_i = 1 \tag{6.4}$$

$$\sum_{j=1}^{i} w_j x_{ij} \leq W_i \tag{6.5}$$

$$x_i + w_i \leq W, \quad i = 1, ..., m \tag{6.6}$$

$$y_i + h_i \leq H, \quad i = 1, ..., m \tag{6.7}$$

$$x_i + w_i \leq x_j \;\; or \;\; x_j + w_j \leq x_i or \tag{6.8}$$

$$y_i + h_i \leq y_j \;\; or \;\; y_j + h_j \leq y_i \tag{6.9}$$

$$\forall i, j \;\; where \;\; i \neq j \tag{6.10}$$

$$p_i \in 0, 1 \tag{6.11}$$

$$x_{ij} \in 0, 1 \tag{6.12}$$

The objective function indicates the maximization of the level surface by placement of selected rectangles such that summation of width is less than or equal to the level width. Equation 6.4 imposes that each item is packed exactly once, either to form the level or placed to maximize the top surface for any other level. Equation 6.5 imposes the width constraint, which is the sum of places rectangle's width that cannot exceed the level width to be maximized. Constraints 6.7, 6.8 and 6.9 ensure that there is no overlapping between any two placed rectangles.

6.4 Proposed Technique

In this section, we discuss in detail the proposed methodology for packing of rectangular blocks. It presents a solution design model for packing the problem. It also demonstrates how the cost associated with the automation of design system can be reduced by integrating hyper-heuristic approach with the model.

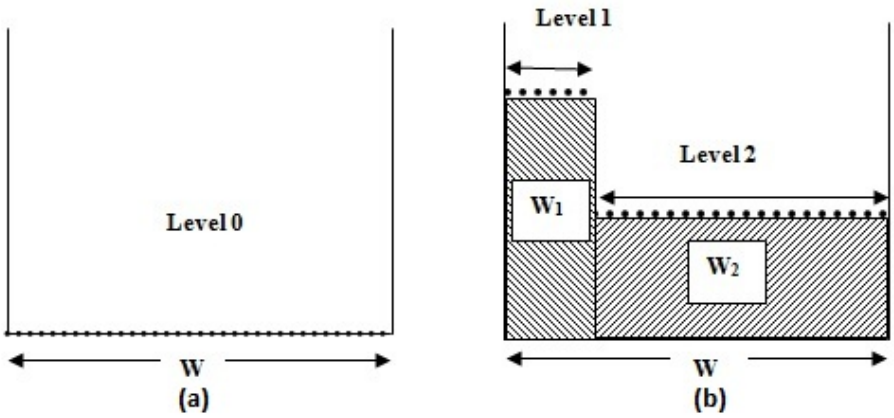


Figure 6.2: Dynamic level creation at the initial stage (a) initial (b) dynamic levels.

6.4.1 Problem Decomposition

To efficiently solve the strip packing problem, the problem is decomposed into a number of subproblems that are recursively solved by the sub-routine using low level heuristic. The best outcome amongst the low level heuristics is selected by the constructive hyper-heuristic. Each low level heuristic has a specific task and the collective framework that constitutes the solution model.

In this technique, we have introduced a new term, namely Level Boundary

Surface (LBS), which refers to the top surface region length for any placed rectangle as shown in Figure 6.2. The technique is called decomposition as at each problem iteration it is divided into a subproblem of maximizing the lowest level boundary surface. The optimal solution to these subproblems are combined to get the final optimal solution. The decomposition for strip packing proceeds in a number of sequential steps. However, it differs from other placement approaches as they are concerned with rectangle selection and then its placement in the appropriate position. On the other hand, the proposed sequential procedure involves the placement of the set of rectangles, where the set may contain one or more rectangles. The placement of the current rectangle set is governed by the already placed ones. The placement of set results in the formation of one or more LBS, this is termed as dynamic level formation. The creation of dynamic level varies as the solution progress. There are two associated possibilities with the formed dynamic level: in the first case, the rectangles that constitute the level surface may either merge with the already existing level or they may result in forming a new level at a different height. Each dynamic level formed has an associated boundary surface to be maximized with an exception to top most boundary level when all rectangles are placed. Thus, at each iteration of the solution process, the short heighted LBS from origin is maximized. It results in changing sheet configuration forming new levels.

At the initial stage, the overall height of the sheet is zero and only a single boundary level surface is available with the width equal to the width of the sheet (W). Figure 6.2 shows the initial sheet configuration and formation of new dynamic levels. As the height of the current level is raised a new level is formed. As shown in Figure 6.2(a), the level_0 is the first level, with height equal to zero and width equal to sheet width. Figure 6.2(b) is the configuration state when two rectangle blocks are placed with the objective of maximizing the current level boundary surface, which results in two different levels as there exist a height difference between the placed rectangles.

Consider the configuration for each level as (x, y, w, h), where (x, y) is the left most corner coordinate, w represents the width parameter and h stands for the height of the current level. Thus, for the initial level, the values are (0, 0, W, 0). The placement of rectangle of same or varying length decides the creation of a number of dynamic levels. The objective at each iteration is to maximize the

occupancy of current LBS width from the set of available rectangles R_1, R_2, ..., R_n. As shown in Figure 6.2, shows the placing of two rectangles with different configuration at $Level_0$. The following conditions may occur

1. If $h_2 \prec h_1$ or $h_1 \prec h_2$ then level will split into two sub-levels as:

   $Level_1 = (x_0, y_0+h_1, w_1, h_0+h_1)$ and $Level_2 = (x_0+x_1), y_0+h_2, w_2, h_0+h_2)$

2. If $h_1 == h_2$, that is, when the placed rectangles are of equal height then levels are combined as:

   $Level_1 = (x_0, y_0+h_1, w_1+w_2, h_0+(h_1 \mid\mid h_2))$

Another case is one where it is required to merge level to form a new dynamic level of larger width, which facilitates the placement of large rectangular blocks. The level can be merged with its neighboring level at the left or right side having the same height to form an extended wide level, as illustrated in designed merging level procedure *MergeRoutine* and Figure 6.3.
In the merge routine, combining of levels occur.
MergeLevels: Given a $Level_i = (x_i, y_i, w_i, h_i)$, a set of rectangles are to be placed at the nearest lowest level either on left or right side of it. The rectangles are first sorted in the ascending order based on the difference of height from the current level. The routine considers two cases with respect to left and right neighbor respectively. For both, the left and right case, the routine terminates when it finds a rectangle, which is higher than current level $Level_i$.

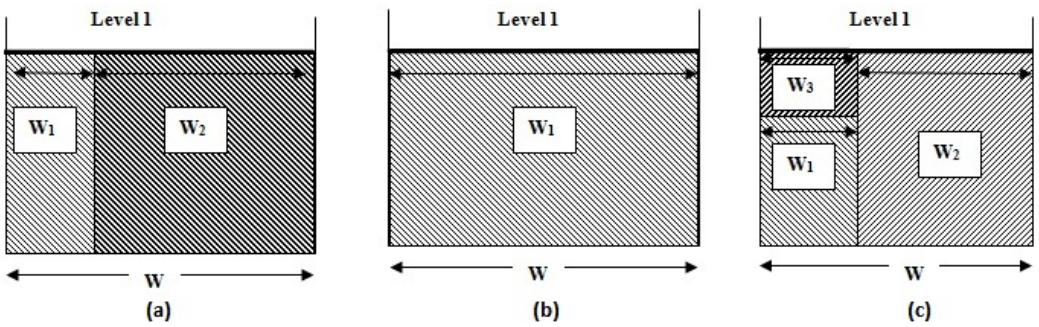


Figure 6.3: Merging of dynamic level.

Another point that must be noted is that during the maximizing of current LBS it may not often result in the formation of new levels, but may also form a single one by either placement of rectangle blocks with same width or by combing with the existing level. All these possible cases are shown in Figure

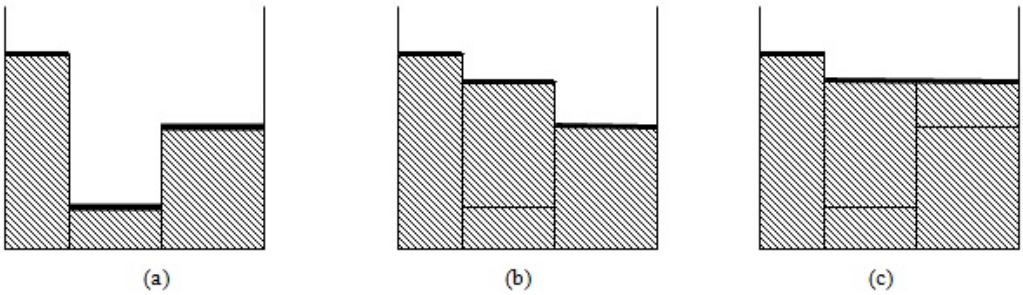


Figure 6.4: Subsequent placement of levels.

6.3. The dark horizontal line is used to denote the level, which is the highest horizontal surface in the current strip configuration. The level may be combined with its neighboring level by the extension of the horizontal line provided both are at the same height.

Figure 6.4 shows the filling at one level and subsequent filling at higher levels as in (a) (b) and (c). The filling always begins to take place with the lowest level surface. These LBS surfaces are selected to be maximized depending on the left most coordinate point having *minimum* value of y^{th} coordinate. Thus, the data structure used to represent level has two measuring parameter the left most coordinates points and the width. In case of one or more levels are combined the width of the new LBS formed extends from the leftmost corner coordinate nearest to the container edge to as far as the vertical edge higher than the current level also described in *MergeRoutine*.

The initial level filling Figure 6.4 (a) shows the maximization of the Level_0 when the height is equal to zero results in creation of three levels. Figure 6.4(b) indicates that at next iteration the lowest level is selected and best placement is performed, raising the height of the current level to form a new one. Figure 6.4(c) shows that the current lowest level is raised by an appropriate strip resulting in expansion of existing level.

The overall system design model is discussed in the *PlacementRoutine*. The routine executes until there exists rectangle to be packed. At each iteration, the lowest LBS are selected to be maximized. The placement of rectangles and formation of dynamic level is carried out in the next step. Finally, the *MergeRoutine* is called for evaluating the scope for expanding the LBS to accommodate rectangles with larger width.

Algorithm 7 MergeRoutine

k=1
Case-1:

if (left placement) **then**

while ((y_k ==y_i)&& ($x_k \prec x_i$)) **do**
w_i=w_i+ w_k
Level$_i$ = (x_k, y_i, w_i , h_i)
k=k+1

end while

end if
Case-2:

if (right placement) **then**

while ((y_k ==y_i)&& ($x_i \prec x_k$)) **do**
w_i=w_i+ w_k
Level$_i$ = (x_i, y_i, w_i, h_i)
k=k+1
end while
end if

Algorithm 8 PlacementRoutine

j=1

while (**do**j $\geq$ n)
Find lowest level$_k$, *i.e.*, one with minimum h_i
Place the rectangle and modify the current Level$_k$
Call *MergeRoutine()*
j=j+1
end while

6.5 Heuristic Decision

Selection of correct group pattern or a set amongst all possible ones is the most decisive part of the algorithm. As for each level, one or more combination can be formed that maximize the LBS width. The selection of the set is determined by a set of heuristic decisions. The Heuristic is stated as determine the shortest heighted rectangle in the generated set and consider two aspects related to it. First, determine whether in the selected set there exists two or more rectangles with equal height or whether there exists some rectangles that can be combined with neighboring LBS. In both the cases, the extension of LBS is possible and the possibility of further solution evaluation. Second, is to check whether current LBS can be maximized or not. The First, heuristic ensures the scope of further evaluation as new LBS can be obtained by placement of the set, and the second ensures that further evaluation would result in creation of vacant space as current LBS cannot be maximized. In case the second governing heuristic is not met by the sequence set, then that sequence is penalized by discarding from further evaluation. This small check helps to greatly reduce the amount of solution search space for each generation. The use of heuristic helps to find the individual with better placement of the rectangle, which becomes much closer to the global optimum.

6.5.1 Hyper-heuristic Combined with Model

The selection of rectangles to maximize the subproblem is the first phase of the model. The second phase is to combine the model with hyper-heuristic technique to select the best set amongst all possible combinations. For all the possible solutions generated by GA, the placement order is determined by these LLH. Then, constructive hyper-heuristic decides the best amongst them for further evaluation. The LLH that we have considered for the placement are:

1. Rightmost Placement: Place at the rightmost position
2. MaxDiff Placement: Place the rectangle so that the difference in the top level with its neighbor is maximal
3. MinDiff Placement: Place the rectangle so that the difference in the top

level with its neighbor is minimal

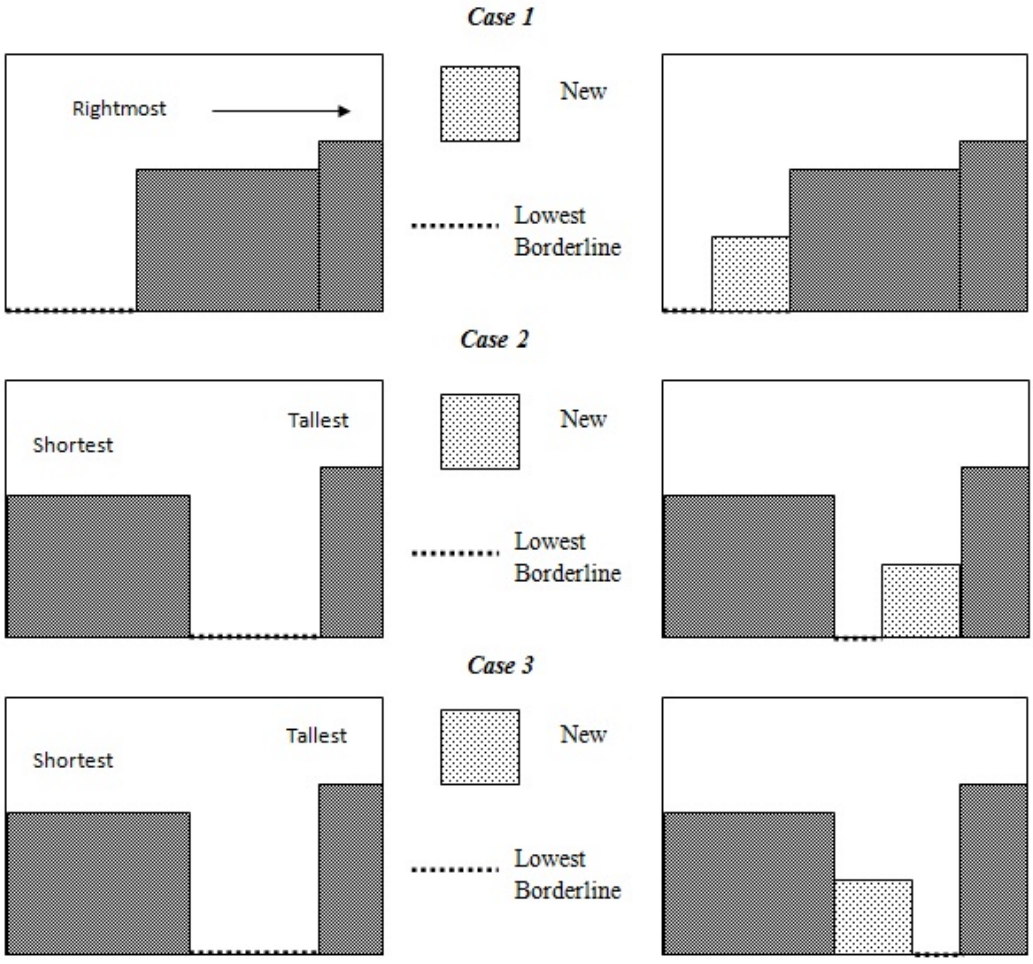


Figure 6.5: Low level heuristic operating on problem instance with 3 cases.

All the three placement policies are applied to the possible solutions. LLH considers the evaluation of all the above three placement policies as shown in Figure 6.5. These placement policies determine the appropriate position for the rectangular block on the top of the layer surface to be maximized. The first placement policy is that the orientation of placement must be right most, *i.e.*, placing the rectangle always towards the right side. The second and third policies are based on the difference in level height and with maximum and minimum difference between the top level surfaces respectively. Based on the placement, a goodness score is assigned to each design layout obtained and the resulting best solution is selected. The heuristic techniques used for the packing resemble the popular best fit approach. However, in this case we have considered all possible placement options and best amongst them is selected for processing of the layout design. The selection amongst the placement options is made by the

evaluation of the goodness function for any design D which is defined as:

$$goodness(D) = Score(H_1) + Score(H_2) + Occupancy\text{-}rate \tag{6.13}$$

where

$$Score(H_1) = \begin{cases} 1 \text{ if shortest heighted rectangle } i \text{ can be merge} \\ \qquad\qquad \text{with existing top level surface } j. \\ 0 \text{ otherwise.} \end{cases} \tag{6.14}$$

$$Score(H_2) = \begin{cases} 1 \text{ if rectangular block exist to maximize the shortest} \\ \qquad\qquad \text{heighted rectangle top level.} \\ 0 \text{ otherwise.} \end{cases} \tag{6.15}$$

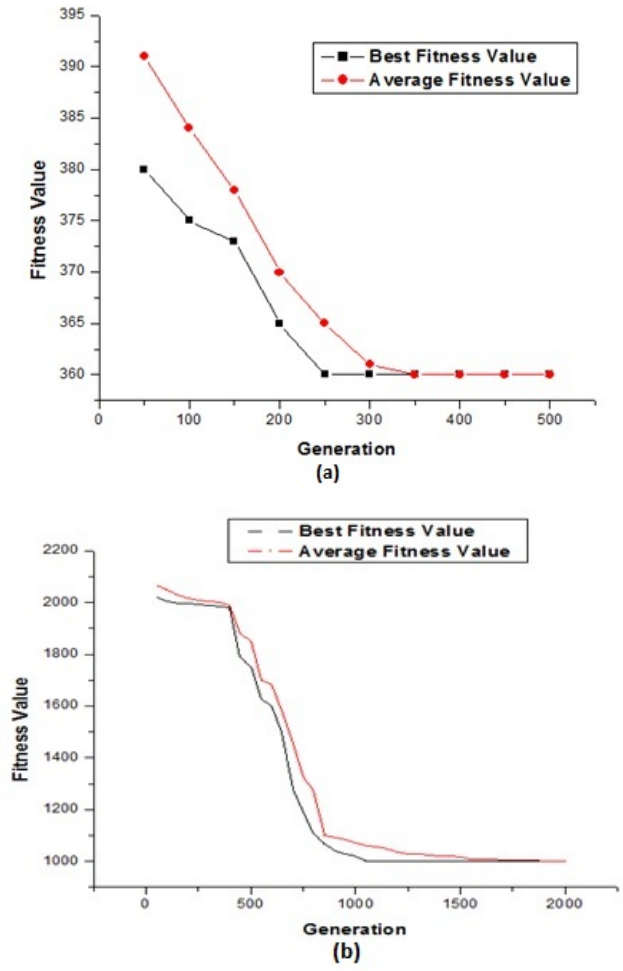


Figure 6.6: Best and average solution against solution cutting (a) 300 pieces (b) 1000 pieces.

Score (H_1) value is 1, if the shortest rectangle can be merged with the existing level j, otherwise 0. Score (H_2) value is 1, when there exists a rectangular block to maximize the shortest highlighted rectangle surface otherwise 0. The occupancy rate stands for the percentage utilization of the selected set in the current design layout. It is measured as the ratio between the total occupied area to the total area, *i.e.*, the maximum height of the design layout multiplied by the width of the container.

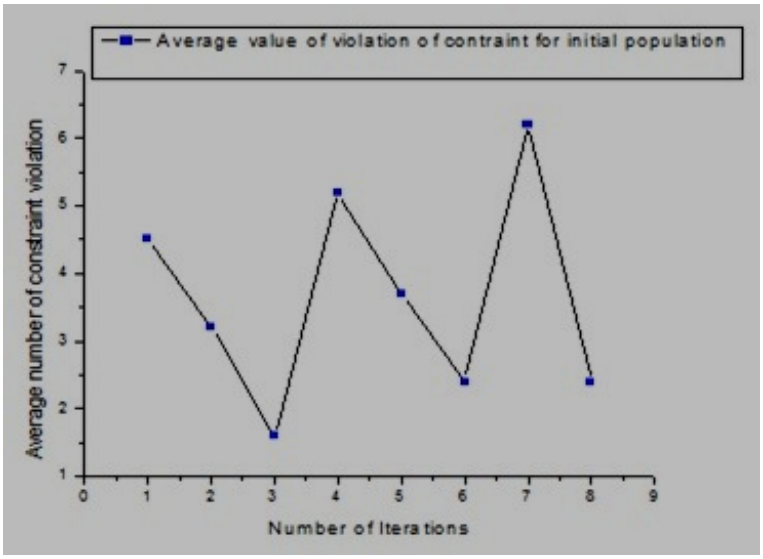


Figure 6.7: Violation of constraints for initial population.

6.5.2 GA Approach to Proposed Model

This section gives a detailed explanation of the process, how GA helps in improving the overall performance efficiency of the decomposition technique. GA is a well known optimization problem for the better exploration of the search space. In the proposed decomposition technique, a number of subproblems are created, which are required to be solved optimally to obtain the overall optimal solution. For each level created either by placement or by possibly merging of neighboring level, a rectangle or a set of rectangles is determined that can maximize the level surface without overlapping. GA task is to find such sets from the available search space of rectangles. The best and average case solution for cutting with different pieces is shown in Figure 6.6. The main benefit of the decomposition can be observed from the fact that most of the initial population combination for the search space would be infeasible, as the sum of their width would exceed the level boundary surface to be maximized. GA penalizes all

such combinations or individuals by not considering them in the further evaluation of the population. Thus, only those individuals survive whose sum is less than or equal to the LBS width. The number of average constraints violated by the initial population is shown in Figure 6.7. It shows that with the number of iterations carried out, the number of individuals violating the constraints is reasonable. The result is shown for a small population size, which indicates that the number will grow as the population size increases. Another benefit of this approach is as the boundary level surface may vary with each subproblem, so the computation time to solve each them is reasonably low.

As the solution approach proceeds iteration to iteration, from bottom most LBS to top most level, the search space is reduced as the marked rectangles placed in the structure at any lower level are never reconsidered. Thus, at each level, GA optimizes boundary surface by determining a single set or multiple set of rectangles that maximizes the occupancy of the each boundary level. As there can be a number of possible combinations, so it is required to select the best possible one. In order to evaluate all possible allocations, the value of individual set is calculated by evaluating the fitness function. The allocation for which the heuristic returns the best value is deemed to be selected by the selection process. The fitness function that determines the selection of a set of rectangles in the design is given as:

$$\textit{fitness(L)} = (H_{maxL} - \frac{\sum_i w_i h_i}{w_L}) + H_L \tag{6.16}$$

Here, the term H_{maxL} stands for the maximum height in the selected set, *i.e.*, H_{maxL}=max(h_1, h_2, ..., h_m), w_i=w_1, w_2,...,w_m where i=1,...,m belongs to the selected set to maximize the current level boundary surface. Here, w_L stands for the width of the current level surface and H_L is the current level height. The fitness is computed with respect to each level boundary surface for each set of possible placement. The fitness value helps in the selection of a set to be considered for the further evaluation of the design layout. The fitness comparison for average and best fitness for cutting 300 and 1000 pieces is shown in Figure 6.6. The computation time grows with the increase in problem size. For all possible allocations, the heuristic is evaluated by computing the value of fitness. The allocation set for which the fitness function returns the best values is the selected set for allocation at the current step.

6.6 Experimental Result

The experimental setup and the datasets used are same as discussed in the previous chapter. We obtained results on a comprehensive datasets from the literature, the details of which are shown in section 5.5.1. In addition, we have also considered 10 class instances from Pisinger *et al.* (2002) [54], which are the random uniformly generated values in the specific selected range. Each class of 10 problems has five instances of size 20, 40, 60, 80 and 100 items.

Number of results exist in the literature for different instances of each dataset. In this section, the results obtained by the proposed hyper-heuristic approach is compared to the best result in the literature for that instance by heuristic, metaheuristic and hyper-heuristic techniques. The heuristic approach includes BLF, FH, BBF, BBFM and LWF. The metaheuristic algorithm like an HGA and HSA metaheuristic approaches are also used for comparison. The approach is also compared with Burke *et al.* [100] hyper-heuristic based on the genetic programming.

Table 6.1 reports the performance of constructive hyper-heuristic compared to other heuristic and metaheuristic approaches on the dataset Jakobs [78], Babu & Babu [79] and Hopper & Turton [65]. Thus, the table contains three groups of datasets with varying number of instances. The results reported in the table is reported for multiple runs for small sets like J1,J2, RB and C1.1 to C6.3 the number of runs were 3 to 5. On the other hand for C7 instances 5 to 8 experimental runs were carried out. The optimal solution (denoted by *Optimum*) is computed beforehand. The 100% filling rates can be achieved only when the total area of the rectangles to be packed is equal to the container area computed as $W * Optimum$. Group-1 is the Jacobs instances with a small number of rectangles to be packed. The proposed approach obtains an optimum solution for both the cases. Group-2 is a single instance of the Babu & Babu dataset. The best-fit approach gives a solution far from optimum, whereas the recent heuristic BBFM and the proposed hyper-heuristic find the optimum solution. Group 3, over the 21 instances, we obtain an average space utilization of 98.67%, which is the best as compared to existing state of art. We achieve 100% space utilisation in 13 instances out of 21.

A computation placement result for HSA, BBFM and proposed approach

Table 6.1: Computational results for Jakobs, Babu & Babu and C test instances.

Dataset	Instance			HGA	LWF	HSA	BF	FH	LFLA	BBF	BBFM	Proposed
	m	W	Optimum									
J1	25	40	15	-	-	-	-	-	-	16	15	15
J2	50	40	15	-	- -	-	-	-	16	15	15	
RB	50	1000	375	-	-	-	400	-	-	400	375	375
C1.1	16	20	20	20	20	20	21	20	20	21	20	20
C1.2	17	20	20	21	20	21	22	20	20	21	21	20
C1.3	16	20	20	20	20	20	24	21	21	21	21	20
C2.1	25	40	15	16	15	15	16	16	15	16	16	15
C2.2	25	40	15	16	15	15	16	15	16	17	15	16
C2.3	25	40	15	15	15	15	16	15	16	16	16	16
C3.1	28	60	30	31	30	30	32	31	30	32	30	31
C3.2	29	60	30	32	31	31	34	31	31	33	31	30
C3.3	28	60	30	33	30	30	33	32	30	33	32	30
C4.1	49	60	60	63	62	60	63	61	62	62	62	61
C4.2	49	60	60	61	61	61	62	61	61	63	61	60
C4.3	49	60	60	62	61	60	62	61	62	62	61	61
C5.1	73	60	90	91	92	90	93	91	91	91	91	90
C5.2	73	60	90	91	91	90	92	90	92	92	91	91
C5.3	73	60	90	91	92	91	93	91	90	92	91	90
C6.1	97	80	120	121	122	120	123	121	120	123	121	120
C6.2	97	80	120	122	121	122	122	121	122	123	122	121
C6.3	97	80	120	121	122	120	124	121	121	123	121	120
C7.1	196	160	240	241	242	242	247	241	240	243	242	240
C7.2	197	160	240	242	241	240	244	241	242	242	242	241
C7.3	196	160	240	243	242	241	245	241	241	243	241	240

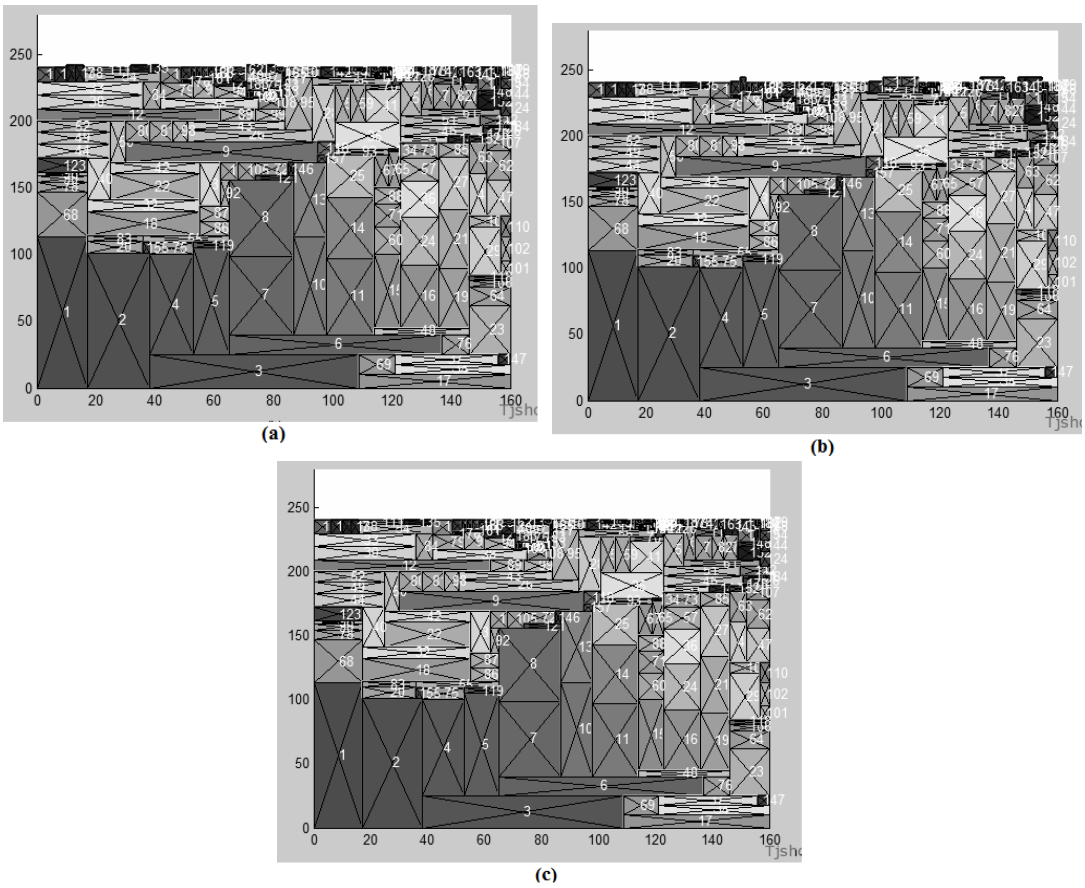


Figure 6.8: A comparison of the (a) HSA, (b) BBFM and (c) Proposed using problem C7.1 from Hopper and Turton instances.

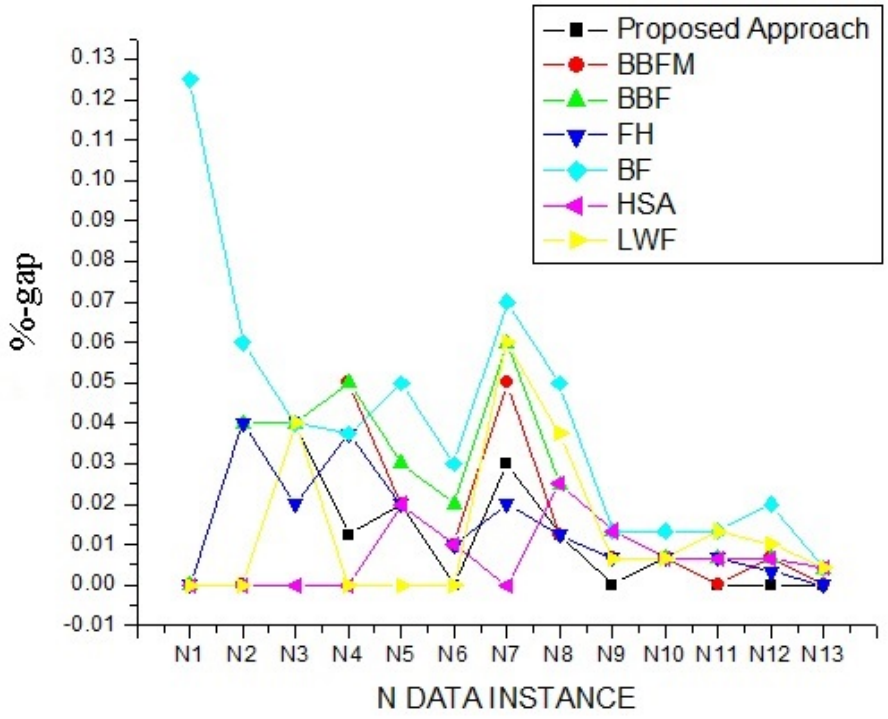


Figure 6.9: Trim loss (%-gap) analysis for Burke instances.

Table 6.2: Computational result for Burke test instances.

Class	Instance			LWF	HSA	BF	FH	BBF	BBFM	Proposed
	m	W	Optimum							
N1	10	40	40	40	40	45	40	40	40	40
N2	20	30	50	50	50	53	52	52	50	50
N3	30	40	50	52	50	52	51	52	52	52
N4	40	80	80	80	80	83	83	82	82	81
N5	50	100	100	100	102	105	102	103	102	102
N6	60	50	100	100	101	103	101	102	101	100
N7	70	80	100	106	100	107	102	106	105	103
N8	80	100	80	83	82	84	81	82	81	81
N9	100	50	150	151	152	152	151	152	151	150
N10	200	70	150	151	151	152	151	151	151	151
N11	300	70	150	152	151	152	151	151	150	150
N12	500	100	300	303	302	306	301	302	302	300
N13	3152	640	960	964	964	964	960	964	960	960

Table 6.3: Computational result for Pisinger and Sigurd test instances.

Class	m	FFDH	BFDH	SAS	SASm	BFS	FC	SC	SCR	Proposed
I	60	86.3	86.6	79.4	86.3	88.6	87.3	88.4	88.3	87.2
II	60	83.7	83.7	80.1	82.2	85.1	85.2	85.4	85.4	84.9
III	60	81.1	81.7	69.6	75.7	82.2	81.9	81.7	81.7	82.0
IV	60	80.0	80.0	76.0	78.0	82.0	83.0	83.0	81.4	82.7
V	60	80.4	81.1	72.6	78.3	81.4	81.3	81.2	81.0	81.0
VI	60	79.1	79.3	76.1	77.1	79.5	80.7	80.5	80.5	80.6
VII	60	79.9	80.2	74.0	80.4	80.9	80.8	80.8	80.9	80.9
VIII	60	80.7	81.1	76.4	79.5	81.6	81.3	81.2	81.2	81.5
IX	60	72.8	72.6	71.6	72.9	73.0	72.6	73.2	73.2	73
X	60	83.3	83.8	73.2	79.4	85.5	85.4	85.3	85.1	85.3

Table 6.4: Computational results for Nice & Path large test instances.

Class	Instance			GRASP	SVC	FH	Proposed
	m	W	Optimum				
NiceL1	1000	100	100.1	102.2	101.5	100.9	100.3
NiceL2	2000	100	100.1	101.5	100.7	100.5	100.2
NiceL3	5000	100	100.1	101	100.4	100.2	100.0
PathL1	1000	100	100.1	101.9	101.2	100.7	100.6
PathL2	2000	100	100.1	101.5	101	100.2	100.0
PathL3	5000	100	99.9	101	100.2	100	100.0

on C7.1 instance is shown in Figure 6.8. Table 6.2 summarizes the result for Burke test instances. In case of smaller instances less than 100 like, N1-N8 the results are reported for 3-5 runs, whereas for N9-N13 the runs were 5-10. The results in the table show that the constructive heuristic has roughly the same performance as compared to LWF and HSA. However, it is able to find the best solution for larger instances like N12 and N13. In comparison to heuristic like BF, FH, BBF and BBFM it is much better. Indeed, the average result for a proposed approach on all instances are better than the best-fit, which indicates that the approach is quite suitable even for large size problem. It also shows that the behavior of constructive heuristic varies with the nature of the problem. The computed result achieves the best outcome for all 7 out of 13 classes. On an average, the proposed approach outperforms all other heuristic and metaheuristic approaches. The trim loss (%-gap) for all the above instances is shown in Figure 6.9, where the proposed approach is with minimum %-gap in most of the cases.

Table 6.3 summarizes the result for Pisinger and Sigurd test instances. It is observed that SAS approach is an exception, on the other hand, other approaches show a fair utilization of available space. The proposed approach gives a better utilization on class VII and IX instance. The approaches like SAS and SASm are preferred because of their fast execution time. The SC approach performs dense packing. The computation time for FC and SCR algorithms are comparatively more. In comparison to other, the proposed constructive approach is capable of achieving higher efficiency at reasonable times.

The statistical independent t-test is carried with respect to the other approaches. The test was carried for a confidence interval of 95% and 10 degrees of freedom. The results reveal that in both cases the value of p is smaller, thus, it can be assumed that the result is statistically significant, *i.e.*, there is a difference between at least two groups and even a difference in variance is observed.

The approach is also evaluated on the larger data instance, as reported in Table 6.4 for number of runs ranging from 7 to 12. It is observed that for all the six instances, the proposed approach is comparativel able to perform better by obtaining minimum packing height. The approach is competitive with FH and better than GRASP and SVC. It is able to find the optimum solution for three

Table 6.5: Comparison between GPA and proposed hyper-heuristic large test instances.

Class	Instance			GPA	Time (in sec.)	Proposed Hyper-heuristic	Time (in sec.)
	m	W	Optimum				
NiceL1	1000	100	100	100.3	865	100.0	548
NiceL2	2000	100	100	100.2	937	100.1	671
NiceL3	5000	100	100	100.0	563	100.0	695
PathL1	1000	100	100	100.6	1537	100.3	1274
PathL2	2000	100	100	100.0	1758	100.1	1562
PathL3	5000	100	100	100.0	5631	100.0	4761

(NiceL3, PathL2, PathL3) out of the six instances. It is observed that for large data instance the data usually contain more number of rectangles with similar dimensions. Most of the approaches fails to effectively handle such dataset and result in sub optimal placement of rectangles. The performance improvement of the proposed approach is governed by the efficiency of the low level heuristic to handle similar and variable size rectangles.

The proposed hyper-heuristic is also compared with the previously proposed GPA approach. The comparison results are reported in Table 6.5. It is observed that for small instances both these approaches have the same behavior. As the problem size increases, although both these approaches are able to find comparatively good solution, but the computation time required by GPA is high. The difference is observed as the proposed hyper-heuristic explore well the low level heuristic, each time selecting the best solution. Another, reason for this behavior is the decomposition of problem into subproblems reduces the problem size in each iterations.

6.7 Chapter Summary

A need for an automated system that generates an efficient pattern resulting in minimum trim loss, which ultimately leads to economical saving. Nowadays, it has become more critical and important. In this chapter, we have designed a robust and effective cutting model. The proposed model is based on the problem decomposition and is flexible to run time changes. Promising results are achieved in addition to many extra benefits like flexibility to change and support to parallel implementations over existing approaches from the literature. Judging

from the results obtained, this model proves to be useful for many industries as it efficiently explores the search space.

Chapter 7

Conclusion

7.1 Discussion

The thesis has shown that there exists a scope for further optimization of the current methods reported in literature, which, if implemented can lead to huge saving, and therefore, future gains exist with the research. We have seen that there exist approaches for generating high quality solutions to both one dimensional cutting stock problem and two dimensional strip packing problem. For 1D-CSP, it is observed that use of metaheuristic approach results in effective column generation technique, which speed up the solution process. Accompanied by the use of heuristic enhancement of these procedures can drive even further improvements in solution qualities. A robust framework for 1D-CSP has been presented, along with a mathematical programming model solver implemented with relative ease. The use of current and considering the future advancement of mathematical programming solver will give incredible solution at a rapid rate.

We have presented a new integrated heuristic method for determining the optimal pattern in multiple length one dimensional cutting stock problem. The main contribution made to the literature over other existing one is that the approach does not enumerate all possible patterns as the size of the combinatorial problem increases. Another benefit is that the approach is able to produce optimal result, even when the ratio between the available stock and the order demand is low. The proposed method is a good tool to solve the real world problems with an acceptable solution in reasonable computation time.

A novel, simplified placement strategy is used for the 2D-SPP framework

which, when combined with metaheuristic proves its dominance even for larger problem instances. We presented a new hybrid approach, which combined genetic encoding and evolution scheme with the proposed placement approach. Such a combination resulted in better population evolution and faster solution convergence to optimal. The approach was subjected to a comprehensive test using benchmark instances. The computation results validate the solution and the effectiveness of the approach.

Further, a robust and effective cutting model based on hyper-heuristic approach is designed that automatically selects the best heuristic by means of exploring the search space of low level heuristics. The proposed approach can be directly applied to future problems in higher dimensions. The performance shown by the statistical analysis indicates that the proposed technique is suitable for placement of items with and without rotation. Lower computation time certainly makes the metaheuristic approach more competitive when compared to other algorithms and therefore, more attractive for industrial applications.

We contribute to the design and implementation for cutting stock and strip packing, in terms of the different data structure and problem representation with varying constraints, which provide an insight into the theoretical analysis and the basis for more constructive algorithm for these problems. The detailed experimentation carried out and the reported results in previous chapters, show the selection of appropriate technique(s) for a given problem. The performance evaluation reports how well these approaches outperform many recent approaches from literature.

7.2 Future Work

As any research project comprises of taking some creative work with an infinite amount of time to it. There are a number of areas that can fall in thesis scope that are quite possibly an extension for future scope. One of the most obvious extensions of the problem is for three and higher dimensions considering the volume utilization, thereby, reducing the transportation cost. Most of the manufacturing and other large industries rely on delivery by vehicles. Hence, efficient measures would result in reducing the number of vehicles used in shipment. A numerous efforts are being made to resolve packaging and transportation issues

safely, economically, and promptly. Good and efficient transportation has a vital role to play in the economy of any nation. Furthermore, the applicability of the developed framework can be tested for different similar domains like scheduling, space allocations, VLSI etc.

The research work reported is based on constructive heuristic, where it can select the best amongst the available solutions. This, can be extended and tested for automatic generation of heuristic based on the search space for higher dimensions, which could result in similar or better performance, but of course need confirming with the appropriate empirical testing. As the scope for optimization exists, with incremental changes to the current methodology can produce significantly larger performance benefits. A more enhanced search technique for generating heuristic can be employed along with the changes in genetic algorithm model to further support parallel implementation and handling problem at large scale.

The work can be further expanded by carrying out theoretical analysis of the heuristic search to analyse support for problem scalability and on parameters like admissibility to determine its ability to accurately predict the performance of the proposed techniques. Metaheuristic technique can be analysed for better convergence and efficiency.

Bibliography

[1] Kantorovich L.V. (1960), Mathematical methods for organising planning production (translated from a report in Russian, dated 1939), Management Science, 6(4), 366–422.

[2] Dyckhoff H. (1990), A typology of cutting and packing problems, European Journal of Operational Research, 44(2), 145–159.

[3] W*ä*scher G., Hausner H., Schumann H. (2007), An improved typology of cutting and packing problems, European Journal of Operational Research, 183(3), 1109–1130.

[4] Lodi A., Martello S., Monaci M. (2002), Two-dimensional packing problems: a survey, European Journal of Operational Research, 141(2), 241–252.

[5] Gradišar M., Resinovič G., Kljajić M. (2002), Evaluation of algorithms for one-dimensional cutting, Computers and Operations Research, 29(9), 1207–1220.

[6] Martello S., Monaci M., Vigo D. (2003), An exact approach to the strip-packing problem, INFORMS Journal on Computing, 15(3), 310–319.

[7] Lodi A., Martello S., Vigo D. (2004), Models and bounds for two-dimensional level packing problems, Journal of Combinatorial Optimization, 8(3), 363–379.

[8] Boschetti M.A., Mingozzi A. (2003), The two-dimensional finite bin packing problem part II: new lower and upper bounds, 4OR (Quarterly Journal of the Belgian, French and Italian Operations Research Societies), 1(2), 135–147.

[9] Gilmore P.C., Gomory R.E. (1961), A linear programming approach to the cutting-stock problem, Operations Research, 9(6), 849–859.

[10] Gilmore P.C., Gomory R.E. (1963), A linear programming approach to the cutting-stock problem part II, Operations Research, 11(6), 863–888.

[11] Hinxman A. (1980), The trim-loss and assortment problems: a survey, European Journal of Operational Research, 5(1), 8–18.

[12] Sweeney P., Paternoster E. (1992), One-dimensional cutting stock decision packing problems, Journal of the Operational Research Society, 43(7), 691–706.

[13] Martello S., Toth P. (1987), Algorithms for knapsack problems, Annals of Discrete Mathematics, 31, 213–258.

[14] Carvalho J.M.V. (2002), LP models for bin packing and cutting stock problems, European Journal of Operational Research, 141(2), 253–273.

[15] Holthaus O. (2002), Decomposition approaches for solving the integer one-dimensional cutting stock problem with different types of standard lengths, European Journal of Operational Research, 141(2), 295–312.

[16] Johnston R., Sadinlija E. (2004), A new model for complete solutions to one-dimensional cutting stock problems, European Journal of Operational Research, 153(1), 176–183.

[17] Belov G., Scheithauer G. (2006), A branch-and-cut-and-price algorithm for one-dimensional stock cutting and two-dimensional two-stage cutting, European Journal of Operational Research, 171(1), 85–106.

[18] Alves C., Macedo R., Carvalho J.M.V. (2009), New lower bounds based on column generation and constraint programming for the pattern minimization problem, Computers and Operations Research, 36(11), 2944–2954.

[19] Jűngerl M., Thienel S. (2000), The ABACUS System for branch-and-cut-and-price algorithms in integer programming and combinatorial optimization, Software: Practice and Experience, 30(11), 1325–1352.

[20] Feilleta D., Gendreaub M., Medagliac A., Walterosc J.L. (2010), Note on branch-and-cut-and-price, Operational Research Letter, 38(5), 346–353.

[21] Scheithauer G., Terno J., Mller A., Belov G. (2001), Solving one-dimensional cutting stock problems exactly with a cutting plane algorithm, Journal of the Operational Research Society, 52(12), 1390–1401.

[22] Liang K.H., Yao X., Newton C., Hoffman D. (2002), A new evolutionary approach to cutting stock problems with and without contiguity, Computers and Operations Research, 29(12), 1641–1659.

[23] Belov G., Scheithauer G. (2002), A cutting plane algorithm for the one-dimensional cutting stock problem with multiple stock lengths, European Journal of Operational Research, 141(2), 274–294.

[24] Alves C., Carvalho J.M.V. (2008), A stabilized branch-and-price-and-cut algorithm for the multiple length cutting stock problem, Computers and Operations Research, 35(4), 1315–1328.

[25] Eilon S., Christofides N. (1971), The loading problem, Management Science, 17, 259–268.

[26] Haessler R. (1971), A heuristic solution to a nonlinear cutting stock problem, Management Science, 17(12), 793–803.

[27] Herz J.C. (1972), A recursive computing procedure for two-dimensional stock cutting, IBM Journal Research and Development, 16(5), 462–469.

[28] Haessler R. (1975), Controlling cutting pattern changes in one-dimensional trim problems, Operational Research, 23(3), 483–493.

[29] Chambers M.L., Dyson R.G. (1976), The cutting stock problem in the flat glass industry selection of stock sizes, Operational Research Quarterly, 27(4), 949–957.

[30] Wolfson M.L. (1965), Selecting the best lengths to stock, Operational Research, 13, 570–585.

[31] Yanasse H.H., Limeira M. (2006), A hybrid heuristic to reduce the number of different patterns in cutting stock problems, Computer and Operation Research, 33(9), 2744–2756.

[32] Trkman P., Gradišar M. (2007), One-dimensional cutting stock optimization in consecutive time periods, European Journal of Operational Research, 179(2), 291–301.

[33] Cui Y. (2012), A CAM system for one-dimensional stock cutting, Advances in Engineering Software, 47(1), 7–16.

[34] Bắck T. (1996), Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms, New York: Oxford University, p. 328.

[35] Zhao S.Z., Ponnuthurai N.S., Zhang Q. (2012), Decomposition-based multiobjective evolutionary algorithm with an ensemble of neighborhood sizes, IEEE Transactions on Evolutionary Computation, 16(3), 442–446.

[36] Holland J.H. (1975), Adaption in Natural and Artificial Systems, Cambridge, MA: MIT Press. Handbook of Genetic Algorithms New York: Van Nostrand Reinhold.

[37] Wang Z., Rao G. (2005), A multi-stage genetic algorithm for one-dimensional cutting stock problems with one stock material length, Journal of Systems Science and Information, 3(3), 643–651.

[38] Khalifa Y., Salem O., Shahin A. (2006), Cutting stock waste reduction using genetic algorithms, In: Proceeding of the 8^{th} Annual Conference on Genetic and Evolutionary Computation (GECCO-06), Seattle, USA, pp. 1675–1680.

[39] Jahromi M.H.M.A., Reza T.M., Makui A., Abbas A.S. (2012), Solving an one-dimensional cutting stock problem by simulated annealing and tabu search, Journal of Industrial Engineering International, 8(1), 8–24.

[40] Jap W.J., Sutanto J.H., Chiaong R. (2008), An implementation of ant colony optimisation for solving cutting stock problem, In: Proceeding of the Fourth IASTED International Conference on Advances in Computer Science and Technology, Langkawi, Malaysia, pp. 225–229.

[41] Yang B., Li C., Huang L., Tan Y., Zhou C. (2009), Solving one-dimensional cutting-stock problem based on ant colony optimization, In: Proceedings of the Fifth International Joint Conference on INC, IMS and IDC (NCM-09), Seoul, Korea, pp. 1188–1191.

[42] Fogarty T.C. (1989), Varying the probability of mutation in genetic algorithms, In: Proceedings of the Third International Conference on Genetic Algorithms, San Francisco, USA, pp. 104–109.

[43] Bắck T. (1992), Self-adaptation in genetic algorithms, In: Proceedings of the First European Conference on Artificial Life, Cambridge, U.K., pp. 263–271.

[44] Smith J., Fogarty T. (1996), Self adaptation of mutation rates in a steady state genetic Algorithm, In: Proceedings of the International Conference on Evolutionary Computation, Nayoya University, Japan, pp. 318–323.

[45] Stone C., Smith J. (2002), Strategy parameter variety in self-adaption, In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-02), New York, pp. 586–593.

[46] Srinivas M., Patnaik L.M. (1994), Adaptive probabilities of crossover and mutation in genetic algorithms, IEEE Transactions on Systems, Man and Cybernetics, 24(4), 656–667.

[47] Hoehn T.P., Pettey C.C. (1999), Parental and cyclic-rate mutation in genetic algorithms:an initial investigation, In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-99), Florida, USA, pp. 297–304.

[48] San L.M. (2007), A new adaptive genetic algorithm for fixed channel assignment, Information Sciences, 177(13), 2655–2678.

[49] Bingul Z. (2007), Adaptive genetic algorithms applied to dynamic multiobjective problems, Applied Soft Computing, 7(3), 791–799.

[50] Ghosh S.C., Sinha B.P., Das N. (2003), Channel assignment using genetic algorithm based on geometric symmetry, IEEE Transactions on Vehicular Technology, 52(4), 860–875.

[51] Gilmore P.C., Gomory R.E. (1965), Multistage cutting stock problems of two and more dimensions, Operations Research, 13(1), 94–120.

[52] Martello S., Monaci M., Vigo D. (2003), An exact approach to the strip-packing problem, INFORMS Journal on Computing, 15(3), 310–319.

[53] Belov G., Scheithauer G. (2006), A branch-and-cut-and-price algorithm for one-dimensional stock cutting and two-dimensional two-stage cutting, European Journal of Operational Research, 171(1), 85–106.

[54] Pisinger D., Sigurd M.M. (2002), Using decomposition techniques and constraint programming for solving the two-dimensional bin packing problem, Technical Report DIKU-03/01, Department of Computer Science, University of Copenhagen, Denmark.

[55] Hifi M., M′Hallah R. (2003), A hybrid algorithm for the two-dimensional layout problem: the cases of regular and irregular shapes, International Transactions in Operational Research, 10(3), 195–216.

[56] Christofides N., Whitlock C. (1977), An algorithm for two-dimensional cutting problems, Operational Research, 25(1), 30–44.

[57] Cui Y. (2008), Heuristic and exact algorithms for generating homogenous constrained three-staged cutting patterns, Computer and Operations Research, 35(1), 212–225.

[58] Cui Y., Yang Y., Cheng X., Song P. (2008), A recursive branch-and-bound algorithm for the rectangular guillotine strip packing problem, Computer and Operations Research, 35(4), 1281–1291.

[59] Lesh N., Marks J., McMahon A., Mitzenmacher M. (2004), Exhaustive approaches to 2D rectangular perfect packings, Information Processing Letters, 90, 7–14.

[60] Kenmochi M., Imamichi T., Nonobe K., Yagiura M., Nagamochi H. (2009), Exact algorithms for the two-dimensional strip packing problem with and without rotations, European Journal of Operational Research, 198(1), 73–83.

[61] Boschetti M.A., Montaletti L. (2010), An exact algorithm for the two-dimensional strip-packing problem, Operations Research, 58(6), 1774–1791.

[62] Arahori Y., T. Imamichi H.N. (2012), An exact strip packing algorithm based on canonical forms, Computers and Operations Research, 39(12), 2991–3011.

[63] Baker B.S., Coffman E.G., Rivest R.L. (1980), Orthogonal packing in two dimension, SIAM Journal on Computing, 9, 846–855.

[64] Lodi A., Martello S., Vigo D. (1999), Heuristic and metaheuristic approaches for a class of two-dimensional bin packing problems, INFORMS Journal on Computing, 11(4), 345–357.

[65] Hopper E., Turton H. (2001), An empirical investigation of meta-heuristic and heuristic algorithms for a 2D packing problem, European Journal of Operational Research, 128, 34–57.

[66] Burke E., Kendall G., Whitwell G. (2004), A new placement heuristic for the orthogonal stock-cutting problem, Operations Research, 54(4), 665–671.

[67] Lesh N., Marks J., McMahon M., Mitzenmacher A. (2005), New heuristic and interactive approaches to 2D rectangular strip packing, Journal of Experimental Algorithmics, 10, 1–18.

[68] Ntene N., Vuuren J. (2009), A survey and comparison of guillotine heuristics for the 2D oriented offline strip packing problem, Discrete Optimization, 6(2), 174–188.

[69] Ortmann F.G., Ntene N., van Vuuren J.H. (2010), New and improved level heuristics for the rectangular strip packing and variable-sized bin packing problems, European Journal of Operational Research, 203(2), 306–315.

[70] Imahori S., Yagiura M. (2010), The best-fit heuristic for the rectangular strip packing problem: an efficient implementation and the worst case approximation ratio, Computers and Operations Research, 37(2), 325–333.

[71] Stephen L., Zhang D. (2011), A fast layer-based heuristic for non-guillotine strip packing, Expert Systems with Applications, 38(10), 13032–13042.

[72] Wei L., Oon W., Zhu W., Lim A. (2011), A skyline heuristic for the 2D rectangular packing and strip packing problems, European Journal of Operational Research, 215(2), 337–346.

[73] Glover F. (1986), Future paths for integer programming and links to artificial intelligence, Computer and Operations Research, 13(5), 533–549.

[74] Glover F. (1990), Tabu search part II, ORSA Journal on Computing, 2(1), 4–32.

[75] Asik O.B., Özcan E. (2009), Bidirectional best-fit approach for orthogonal rectangular strip packing, Annals of Operations Research, 172(1), 405–427.

[76] Özcan E., Kai Z., Drake J.H. (2013), Bidirectional best-fit heuristic considering compound placement for two dimensional orthogonal rectangular strip packing, Expert Systems with Applications, 40(10), 4035–4043.

[77] Yang S., Han S., Ye W. (2013), A simple randomized algorithm for two dimensional strip packing, Computers and Operations Research, 40(1), 1–8.

[78] Jakobs S. (1996), On genetic algorithms for the packing of polygons, European Journal of Operational Research, 88(1), 165–181.

[79] Babu A.R., Babu N.R. (1999), Effective nesting of rectangular parts in multiple rectangular sheets using genetic and heuristic algorithms, International Journal of Production Research, 37(7), 1625–1643.

[80] Dowsland K., Herbert E., Kendall G., Burke E. (2006), Using tree search bounds to enhance a genetic algorithm approach to two rectangle packing problems, European Journal of Operational Research, 168(2), 390–402.

[81] Gonalves J.F. (2007), A hybrid genetic algorithm-heuristic for a two-dimensional orthogonal packing problem, European Journal of Operational Research, 183(3), 1212–1229.

[82] Dagli C.H., Hajakbari A. (1990), Simulated annealing approach for solving stock cutting problem, In: Proceedings of the IEEE International Conference on Systems, Man, Cybernetics (SMC-1990), Washington, DC, pp. 221–223.

[83] Lai K., Chan J. (1996), Developing a simulated annealing algorithm for the cutting stock problem, Computers and Industrial Engineering, 32(1), 115–127.

[84] Martins T., Tsuzuki M. (2010), Simulated annealing applied to the irregular rotational placement of shapes over containers with fixed dimensions, Expert Systems with Applications, 37(3), 1955–1972.

[85] Dagli C., Poshyanonda P. (1997), New approaches to nesting rectangular patterns, Journal of Intelligent Manufacturing, 8(3), 177–190.

[86] Bortfeldt A. (2013), A reduction approach for solving the rectangle packing area minimization problem, Discrete Optimization: European Journal of Operational Research, 224(3), 486–496.

[87] Liu D., Teng H. (1999), An improved BL-algorithm for genetic algorithms of the orthogonal packing of rectangles, European Journal of Operational Research, 112(2), 413–420.

[88] Yeung L.H.W., Tang W.K. (2003), A hybrid genetic approach for garment cutting in the clothing industry, IEEE Transactions on Industrial Electronics, 50(3), 449–455.

[89] Gonalves J.F. (2007), A hybrid genetic algorithm-heuristic for a two-dimensional orthogonal packing problem, European Journal of Operational Research, 183(3), 1212–1229.

[90] Burke E.K., Kendall G., Whitwell G. (2009), A simulated annealing enhancement of the best-fit heuristic for the orthogonal stock-cutting problem, INFORMS Journal on Computing, 21(3), 505–516.

[91] Burke E.K., Kendall G., Whitwell G. (2004), A new placement heuristic for the orthogonal stock-cutting problem, Operations Research, 52(4), 655–671.

[92] Alvarez-Valdes R., Parreo F., Tamarit J. (2009), A branch and bound algorithm for the strip packing problem, OR Spectrum, 31(2), 431–459.

[93] Alvarez-Valdes R., Parreo F., Tamarit J. (2008), Reactive GRASP for the strip packing problem, Computers and Operations Research, 35(4), 1065–1083.

[94] Belov G., Scheithauer G., Mukhacheva E.A. (2008), One-dimensional heuristics adapted for two-dimensional rectangular strip packing, Journal of Operational Research Society, 59(6), 823–832.

[95] Wei L., Zhang D., Chen Q. (2009), A least wasted first heuristic algorithm for the rectangular packing problem, Computers and Operations Research, 36(5), 1608–1614.

[96] Stephen L., Defu Z., Changle Z., Tao W. (2012), A hybrid simulated annealing metaheuristic algorithm for the two-dimensional knapsack packing problem, Computers and Operations Research, 39(1), 64–73.

[97] Leung S., Zhang D., Sim K.M. (2011), A two-stage intelligent search algorithm for the two-dimensional strip packing problem, European Journal of Operational Research, 215(1), 57–69.

[98] Denzinger J., Fuchs M. (1996), High performance ATP systems by combining several AI methods, Technical Report, SEKI-Report SR-96-09, University of Kaiserslautern.

[99] Cowling P., Kendall G., Soubeiga E. (2000), A hyperheuristic approach to scheduling a sales summit, In: Proceeding of the 3^{rd} International Conference Practice Theory Automated Timetabling (PATAT), Konstanz, Germany, pp. 176–190.

[100] Burke E.K., Gendreau M.M., Kendall G., Ochoa G., Özcan E., Qu R. (2013), Hyper-heuristics: a survey of the state of the art, Journal of the Operational Research Society, 64, 1–30.

[101] Fisher H., Thompson G.L. (1963), Probabilistic Learning Combinations of Local Job-shop Scheduling Rules, Industrial Scheduling, Prentice-Hall, Englewood Cliffs, pp. 225–251.

[102] Crowston W.B., Glover F., Thompson G.L., Trawick J.D. (1963), Probabilistic and parametric learning combinations of local job shop scheduling rules, ONR Research Memorandum, GSIA, Carnegie Mellon University, Pittsburgh, p. 117.

[103] Storer R.H., Wu S.D., Vaccari R. (1992), New search spaces for sequencing problems with application to job shop scheduling, Management Science, 38(10), 1495–1509.

[104] Storer R.H., Wu S.D., Vaccari R. (1995), Problem and heuristic space search strategies for job shop scheduling, ORSA Journal of Computing, 7(4), 453–467.

[105] Fang H., Ross P., Corne D. (1993), A promising genetic algorithm approach to job shop scheduling, rescheduling, and open-shop scheduling problems, In: Proceeding of the International Conference on Genetic Algorithms, Urbana-Champaign, USA, pp. 375–382.

[106] Fang H., Ross P., Corne D. (1994), A Promising Hybrid GA/ Heuristic Approach for Openshop Scheduling Problems, John Wiley and Sons, New York.

[107] Drechsler R., Gckel N., Becker B. (1996), Learning heuristics for OBDD minimization by evolutionary algorithms, In: Proceeding of the Parallel Problem Solving from Nature (PPSN), Berlin, Germany, pp. 730–739.

[108] Burke E.K., Hyde M., Kendall G., Woodward J. (2010), A genetic programming hyper-heuristic approach for evolving 2-D strip packing heuristics, IEEE Transactions on Evolutionary Computation, 14(6), 942–958.

[109] Bazaraa M.S., Jarvis J.J., Sherali H.D. (1980), Linear Programming and Network Flows, 2^{nd} edition John Wiley, New York.

[110] Zhao S.Z., Ponnuthurai N.S., Zhang Q. (2012), Decomposition-based multiobjective evolutionary algorithm with an ensemble of neighborhood sizes, IEEE Transactions on Evolutionary Computation, 16(3), 442–446.

[111] Goldberg D. (1989), Genetic Algorithms in Search, Optimization, and Machine Learning, MA: Addison-Wesley.

[112] Goldberg D. (1992), The Design of Innovation, Springer-Science+Business Media, B.V.

[113] Kazarlis S., Petridis V. (1998), Varying fitness functions in genetic algorithms: Studying the rate of increase of the dynamic penalty terms, Parallel Problem Solving from Nature PPSN V, 1498(1), 211–220.

[114] Gonalves J.F., Resende M.G. (2011), Biased random-key genetic algorithms for combinatorial optimization, Journal of Heuristics, 17(5), 487–525.

[115] Dejong K.A. (1980), Adaptive system design: a genetic approach, IEEE Transactions on Systems, Man and Cybernetics, 10(9), 556–574.

[116] Belov G., Scheithauer G. (2003), The number of setups (different patterns) in one-dimensional stock cutting, Technical Report MATH-NM-15-2003, Dresden University.

[117] Gau T., Wäscher G. (1995), CUTGEN1: A problem generator for the standard one-dimensional cutting stock problem, European Journal of Operational Research, 84(3), 572–579.

[118] Umetani S., Yagiura M., Ibaraki T. (2006), One-dimensional cutting stock problem with a given number of setups: a hybrid approach of metaheuristics and linear programming, Journal of Mathematical Modelling and Algorithms, 5(1), 43–64.

[119] Garey M.R., Johnson D.S. (1972), Computers and Intractability: A Guide to the Theory of NP-completeness, New York, NY, USA: W.H.

[120] Matsumoto K., Umetani S., Nagamochi H. (2011), On the one-dimensional stock cutting problem in the paper tube industry, Journal of Scheduling, 14(3), 281–290.

[121] Cui Y., Yang Y. (2010), A heuristic for the one-dimensional cutting stock problem with usable leftover, Discrete Optimization, 204(2), 245–250.

[122] Bean J. (1994), Genetics and random keys for sequencing and optimization, ORSA Journal on Computing, 6, 154–160.

[123] Valenzuela C.L., Wang P.Y. (2001), Heuristics for large strip packing problems with guillotine patterns: an empirical study, In: Proceeding of the 4^{th} Meta-heuristics International Conference (MIC-2001), Porto, Portugal, pp. 417–421.

[124] Iori M., Martello S., Monaci M. (2002), Metaheuristic algorithms for the strip packing problem, Optimization and Industry: New Frontiers. Kluwer Academic.