Solving the Facility Layout Problem Using Genetic and Progressive Algorithms

A Thesis

Submitted to the Faculty of Graduate Studies and Research

In Partial Fulfillment of the Requirements

For the Degree of

Master of Applied Science

in

Industrial Systems Engineering

University of Regina

By

Sodiq Olumide Fowosere

Regina, Saskatchewan

December, 2017

UNIVERSITY OF REGINA

FACULTY OF GRADUATE STUDIES AND RESEARCH

SUPERVISORY AND EXAMINING COMMITTEE

Sodiq Olumide Fowosere, candidate for the degree of Master of Applied Science in Industrial Systems Engineering**,** has presented a thesis titled, ***Solving the Facility Layout Problem Using Genetic and Progressive Algorithms,*** in an oral examination held on September 1, 2017. The following committee members have found the thesis acceptable in form and content, and that the candidate demonstrated satisfactory knowledge of the subject material.

External Examiner:	Dr. Yasser Morgan, Software Systems Engineering
Supervisor:	Dr. Mohamed Ismail, Indusrial Systems Engineering
Committee Member:	*Dr. Mehran Mehrandezh, Industrial Systems Engineering
Committee Member:	Dr. Denise Stilling, Industrial Systems Engineering
Chair of Defense:	Dr. Laurie Clune, Faculty of Nursing

*via SKYPE

ABSTRACT

In modern markets, manufacturers and service systems are faced with the need to constantly respond to changing product mixes, consumer's tastes and demands. Facility planning is a significant strategy used to achieve this goal. A well-planned facility minimizes the materials and personnel flow distances, inventory build-up and consequently improves the production lead time and operation cost. Therefore, this thesis explores two variations of the facility layout problem: the Equal Areas dynamic facility layout problem (EA-DFLP) and the unequal area facility layout problem (UA-FLP). The EA-DFLP involves arranging and rearranging a number of equal departments in a facility such that the sum of material handling and rearrangement cost is minimized. The UA-FLP involves finding the best arrangement for a number of departments that minimizes the material handling cost without violating the shape constraints.

In this thesis, a genetic algorithm is proposed to solve the EA-DFLP. The genetic algorithm is a direct implementation with the addition of an elitism criteria. This thesis also presents a new modeling approach called Progr essive Modeling (PM) and applies it to solve the Unequal Areas Facility Layout Problem (UA-FLP). A problem solution is represented by an object-oriented binary tree and an objective graph. PM introduces a component model to deploy the problem logic and its solution algorithm over several interacting components. Component models isolate the objective space from the search space in a black-box fashion. A novel solution algorithm is presented to demonstrate how the search process is managed and controlled while searching for optimal or near-optimal solutions. The solution process and the optimization algorithm are demonstrated using the developed software framework.

The Genetic Algorithm and the Progressive Modeling were both tested using well-known problems from the literature. The Genetic Algorithm was applied to the EA-DFLP while the Progressive Modeling was used to solve the UA-FLP. The results using the genetic algorithm showed a better performance when compared to other solutions from literature. The genetic algorithm generated a new best solution (0.8% improvement) for one of the test problems and it obtained the best-found results for most of the rest.

For small sized UA-FLP, all optimum or best-known solutions have been obtained. The PM generated the best solutions for the midsize problems (problems with 10, 12 and 14 departments). For eight values of aspect ratios tested for the large size problem, PM generated seven new best solution out of eight problems. The developed problem analysis, solution algorithm, and results demonstrate why the proposed modeling approach is promising and capable of handling more complex real-world problems. The comparison between progressive algorithms and genetic algorithms will be left for future research.

ACKNOWLEDGEMENT

I am grateful to Almighty God for His favour and blessings throughout my master's program.

I would also like to express my genuine gratitude to my supervisor Dr. Mohamed Ismail for being my knowledge hub. His guidance and feedback throughout my program are invaluable.

My sincere appreciation goes to my mother, Olayinka Fowosere and my friends Benjamin Decardi-Nelson and Anima Osei for their support.

DEDICATION

I dedicate this thesis to my mother, Olayinka Fowosere for her support and prayers.

TABLE OF CONTENT

ABSTRACT i

ACKNOWLEDGEMENT iii

DEDICATION iv

TABLE OF CONTENT v

LIST OF TABLES ix

LIST OF FIGURES x

LIST OF ABBREVIATIONS xi

CHAPTER ONE: INTRODUCTION 1

1.1 Background Information 1

1.2 Thesis Statement 6

1.3 Research Objectives 6

1.4 Thesis Outline 7

CHAPTER TWO: LITERATURE REVIEW 9

2.1 Facility Layout Problem 9

2.1.1 Static facility layout problem 9

2.1.2 Dynamic facility layout problem 10

2.2 Models and Formulations for Facility Layout Problem 13

2.2.1 Quadratic assignment problem 15

2.2.2 Mixed integer programming 16

2.3 Methodologies for Equal Area Dynamic Facility Layout Problem (EA-DFLP) 17
2.3.1 Exact approaches 17
2.3.2 Heuristics for EA-DFLP 19
2.3.2.1 Pairwise interchange heuristic 19
2.3.2.2 Other heuristics 21
2.3.3 Meta-heuristics for EA-DFLP 21
2.3.3.1 Genetic algorithms 21
2.3.3.2 Simulated annealing algorithms 23
2.3.3.3 Ant colony algorithms 24
2.3.3.4 Tabu search heuristics 25
2.3.4 Hybrid approaches for EA-DFLP 28
2.4 Unequal Area Problems 32
2.4.1 Slicing tree structure (STS) 32
2.4.2 Flexible bay structure (FBS) 33
CHAPTER THREE: EQUAL AREA DYNAMIC FACILITY LAYOUT PROBLEM 35
3.1 Equal Area Formulation for DFLP 36
3.2 The Genetic Algorithm 38
3.2.1 DFLP Encoding 39
3.2.2 Initialization 42
3.2.3 Fitness Function 42

3.2.4 Fitness Transferral and Selection 43
3.2.5 Crossover Operator 44
3.2.6 Mutation Operator 48
3.2.7 Replacement and Elitism 50
3.2.8 Algorithm Termination 50
3.3 Computational Study 52
3.3.1 Problem I 52
3.3.2 Problem II 53
3.3.3 Problem III 56
CHAPTER FOUR: UNEQUAL AREA FACILITY LAYOUT PROBLEM 59
4.1 Progressive Modeling Process 60
4.2 UA-FLP Progressive Model 64
4.2.1 Analytics, Component Model and FactDesign 64
4.2.2 Solution representation 64
4.2.3 The logic that govern 70
4.2.4 Progressive Algorithm 70
4.2.4.1 Initialize the search space 70
4.2.4.2 Selection process 71
4.2.4.3 Neighborhood Search 71
4.2.5 Numerical experiments and discussions 73

CHAPTER FIVE: CONCLUSION AND RECOMMENDATIONS 77

5.1 Conclusion 77

5.2 Recommendations 78

REFERENCES 79

APPENDICES 86

APPENDIX A 87

APPENDIX B 89

APPENDIX C 91

LIST OF TABLES

Table 2.1: Examples of methodologies used for equal-area DFLP.....31

Table 3.1: Parameter Settings 57

Table 3.2: Results and Comparison for Problem Data taken from Lacksonen and Enscore Jr (1993).....58

Table 4.1: The results for Van-Camp, BA 12 and BA 14 75

Table 4.2: AB20 Results.....76

LIST OF FIGURES

Figure 2.1: Classifications of the FLP and its formulation (Drira et al, 2007) 14

Figure 3.1: A sample grid-like layout for one time period and its coded chromosome 40

Figure 3.2: Sample grid-like layouts for a 3-period DFLP and its coded chromosome....41

Figure 3.3: Single point crossover operator for a 6-department and 2-period problem 47

Figure 3.4: Exchange mutation operator for a 6-department and 2-period problem.....49

Figure 3.5: The Genetic Algorithm Flow Chart.....51

Figure 3.6: Best Solution Obtained for Problem I.....54

Figure 3.7: Best Solution Obtained for Problem II 55

Figure 4.1: Progressive Modeling Process.....63

Figure 4.2: The Tree and the Flow/Distance Graph.....66

Figure 4.3: The Tree, the Layout, and the Flow/Distance Graph.....67

Figure 4.4: Component diagram for UA-FLP.....69

Figure 4.5: Tree rotation operator 72

Figure 4.6: Swapping 2 & 7 leaves (departments) 72

Figure 4.7: Flipping 2 & 4 parent node 73

Figure 4.8: Van Camp Solution, TMHC = 19655.3.....74

Figure A.1: Material flow matrix for Rosenblatt 6-department problem 87

Figure B.1: Material flow matrix for Conway and Venkataramanan problem 89

Figure C.1: Problem Number 1 from Lacksonen and Enscore Jr (1993). 91

LIST OF ABBREVIATIONS

ACO	Ant colony optimization algorithm
AI	Artificial intelligence
CRAFT	Computerized relative allocation of facilities technique
DFLP	Dynamic facility layout problem
DP	Dynamic programming
EA-DFLP	Equal area dynamic facility layout problem
FactDesign	Factory design software
FBS	Flexible bay structure
FLP	Facility layout problem
GA	Genetic Algorithm
HAS	Hybrid ant systems
MIP	Mixed integer programming
PA	Progressive Algorithm
PM	Progressive modeling
QAP	Quadratic Assignment Problem
RWS	Roulette wheel selection
SA	Simulated Annealing
SFLP	Static facility layout problem

STS	Slicing tree structure
TSH	Tabu search heuristics
UA-FLP	Unequal area facility layout problem
UML	Unified modeling language

CHAPTER ONE: INTRODUCTION

1.1 Background Information

In today's dynamic global market, facility planning is a strategy that organizations use to stay competitive. Several needs arise with the competitive nature of the market; they include a lean approach to operations to reduce cost, the need to offer competitive price, and reducing lead times to improve customer satisfaction. A well-planned facility is a crucial step to help an organization achieve these goals. To ensure that the supply chain of any industry or organization is a success, the facilities have to be planned (either that of the factory, the distribution centers, stores or service centers) so that the objectives of flexibility, upgradability and adaptability can be achieved. Facility planning determines how an activity's tangible fixed assets best support achieving the activity's objectives (Tompkins et al., 2010).

The importance of a well-planned facility cannot be overemphasized. Some of the benefits of facility planning are improved customer satisfaction through prompt response to customer needs; increased return on assets by maximizing the inventory turnover; and reducing cost due to storage of built-up inventory, inventory obsolescence and distribution cost. Other benefits include increased supply chain profitability, improved maintenance, improved and safer work environment, enhanced material flow and handling (Tompkins et al., 2010).

The physical arrangement of the departments (production areas, production-related and support areas, personnel areas) in a facility is known as the facility layout (Tompkins et al., 2010). Determining this physical arrangement of the departments of a production

facility or a service centre is known as the facility layout problem (FLP) (Singh and Sharma, 2006). Singh and Sharma (2006) further explained FLP as finding the most efficient arrangement of the departments within a facility. The layout of a facility has an important role to play in the effectiveness of a production system, profitability and supply chain excellence in general. As a result, this is an important area of research in production economics and advanced manufacturing.

Facility layout decisions have direct implications on the operational efficiency in both short and long-term. Studies show that in manufacturing, between 20 and 50% of the total operating expenses can be assigned to material handling and this cost can be reduced by 10 to 30% if the facilities are effectively planned (Tompkins et al., 2010). Singh and Sharma (2006) reported that reduced material movement in a facility lowers the throughput time, work-in-progress inventory, and the operation congestion. As a result, minimizing the material handling cost is a common objective considered in the facility layout problem. The material handling cost can be estimated by multiplying the material flow between interacting departments and the associated distance between them (McKendall and Shang, 2006). Travelled distance may be estimated between input/output locations or departments centroids. Both rectilinear and Euclidean measures are widely accepted distance metrics. The associated data of material flow and distance data is usually organized in a matrix format called from-to charts.

The facility layout problem could be classified as either single objective or multi-objective. Most researchers and practitioners have agreed that to be more realistic, more than a single objective is to be considered when planning a facility. Common objectives considered in literature are minimizing material handling cost and maximizing the total

closeness ratings. The importance of minimizing the material handling cost has been iterated first while the closeness rating is very useful when there is a need to keep specific departments separate for safety or noise reasons even if there is high traffic between them. These objectives are conflicting and several researchers reached the consensus that a better layout should leverage the benefits of the objectives and eliminate some of their pitfalls. The Pareto approach has been effective in generating non-dominated solutions that integrate this leverage (Deb et al., 2002).

Based on the material flow, the facility layout problem can be classified into two major categories: the static facility layout problem (SFLP) and the dynamic facility layout problem (DFLP) (Drira et al., 2007). The static layout problem arises when the flow of materials between the departments is constant during a planning horizon (Kuppusamy, 2001). The SFLP is sufficient if the material flow would not change during the planning horizon (Balakrishnan and Cheng, 1998). However, in today's competitive global market, a major feature of a manufacturing environment is volatility. In the modern day market, product mix and demand are continuously changing. Therefore, the material flow changes from time to time and thus affect the layout. For this reason, static layout designs may not meet the needs of modern day organisations. Departments of the facility have to be re-arranged so as to respond to changes in demand and product mix. This gives rise to the dynamic facility layout problem (DFLP).

Both Shore and Tompkins (1980) and Baykasoğlu and Gindy (2001) identified the following as some of the factors that are associated with changing material flow between the departments in a facility:

1. Changes in the quantities of production and associated production schedule

2. Change in the design of existing products

3. Addition or removal of a product

4. Change in the processing sequence for existing products

5. Short life cycle of products

6. Change or replacement of production equipment

An assumption that the material flow changes over time during the planning horizon complicates the facility planning beyond SFLP. A layout plan that minimizes the material handling cost during a time period might result in higher material handling cost for the next. The DFLP is an extension of the SFLP, layout for each time period can be obtained by using the SFLP assumption (Balakrishnan and Cheng, 1998). The time period could be weeks, months or even years and material flow during a period is assumed to be constant. DFLP involves selecting a static layout for a period in a multi-period planning horizon and deciding whether to change the arrangement of the facility in the coming period (Balakrishnan et al., 2003). A trade-off exists between the extra material handling cost of retaining a current layout in the next period and the cost that would have been incurred in rearranging the current layout. The rearrangement cost includes both the cost of moving the departments and the cost of lost production during the process of rearrangement. If the extra material handling cost is low compared to the rearrangement cost, then there is no justification for rearrangement and vice versa (Balakrishnan and Cheng, 1998). Hence, DFLP involves efficient arrangement of the departments of a facility such that the sum of material handling and rearrangement cost is minimized during a multi-period planning horizon (McKendall and Shang, 2006).

The dynamic facility layout problem (DFLP) is a computationally complex combinatorial optimization problem for which exact solution have only been found for small sized problems (Baykasoglu et al., 2006). The problem is classified as NP-hard; complexity increases exponentially with the number of departments (Mihajlović et al., 2007). With its NP-hard nature, the DFLP can only be solved to optimal when the number of departments is small. For large sized problems, the solution approach becomes intractable. Rosenblatt (1986) made the first major attempt to solve the dynamic facility layout problem using the dynamic programming method (DP) and optimal solutions were obtained for only small sized problems. In the past three decades, heuristic approaches have been developed by researchers to provide a near optimum solution for the DFLP in considerable time. Review of some of the heuristics was completed by Balakrishnan and Cheng (1998).

Several models have been used in the literature to represent the facility layout problem (FLP), some of them are the quadratic assignment problem (QAP), graph theory model, slicing tree structure (STS) and mixed integer programming model (MIP). The most commonly used formulation for the dynamic facility layout problem is the QAP (Singh and Sharma, 2006). The QAP is a discrete formulation for the DFLP (Drira et al., 2007). In this approach, the facility is divided into equal size rectangular boxes and a department may be assigned to just one or more of these locations. The equal area DFLP has been formulated into QAP and solved using different heuristic and meta-heuristic approaches by different researchers in the past, but this assumption is not practical.

A more practical approach to this problem is the unequal-area DFLP, where departments are considered to have different sizes. Some researchers used the discrete

equal sized boxes, where a department may occupy one or more of these boxes. Other researchers used the continuous approach of mixed integer programming where departments can be placed anywhere within the floor space, provided that they do not overlap. However, only small-sized problems have been solved to optimal using this approach. An emerging and more promising approach is the slicing tree structure (STS). This object-oriented binary tree representation has leaves representing individual departments and internal nodes represent cascaded layout placeholders.

1.2 Thesis Statement

Facility layout problems become computationally intractable as the number of departments increase. Therefore, there is need to develop smart and informed algorithms to solve large-sized problems. This thesis presents a Genetic Algorithm and Progressive Modeling, a new modeling approach to solve a couple of variants of the problem, namely the dynamic layout problem and unequal areas problem.

1.3 Research Objectives

The objectives of this research are as follows:

1. To develop a facility layout design framework
2. To develop a genetic algorithm for the equal area dynamic facility layout problem (EA-DFLP).
3. Introduce a new data model and a generalized framework for the unequal area facility layout problem (UA-FLP).
4. Introduce a new solution approach and algorithm for the unequal area facility layout problem (UA-FLP): the doubly linked graph and slicing trees, and its progressive algorithm.

1.4 Thesis Outline

Chapter 1: Introduction

The background knowledge of the topic includes facility layout and planning. This chapter also discusses the research scope and objectives, alongside with an insight to the solution approaches presented in this thesis.

Chapter 2: Literature Review

The existing and emerging trends in the literature of both equal-area dynamic facility layout problem (EA-DFLP) and unequal-area facility layout problem facility layout (UA-FLP) are presented.

Chapter 3: Equal-Area Dynamic Facility Layout Problem

This chapter discusses the equal area dynamic facility layout problem (EA-DFLP). A genetic algorithm using elitism as its replacement criteria to solve this problem is presented. A set of well-known problems were taken from the literature to prove the effectiveness of the developed algorithm.

Chapter 4: Unequal Area Facility Layout Problem

Chapter four presents a new modeling framework called Progressive Modeling. This chapter demonstrates the approach by solving the UA-FLP. A handful of well-known benchmark problems were solved to prove that this approach is promising in solving complex real-world problems.

Chapter 5: Conclusion and Recommendations

This chapter summarizes the thesis and presents the contributions of the research. It also recommends future research directions that we consider typical extensions of this thesis.

CHAPTER TWO: LITERATURE REVIEW

2.1 Facility Layout Problem

The solution methodologies for the facility layout problem can be divided into three major categories: (1) the exact procedures, (2) heuristic and (3) meta-heuristic approaches. The exact procedures give optimal solutions for the FLP while the heuristic approaches give suboptimal solution (Balakrishnan and Cheng, 1998; Kuppusamy, 2001; Singh and Sharma, 2006). Some other approaches also exist in literature. Artificial intelligence approaches include neural network, fuzzy logic and expert system (Singh and Sharma, 2006). The DFLP, just like the SFLP are both computationally intractable when department size increases (Kuppusamy, 2001). Optimum solutions can only be obtained for small sized problems as presented by Rosenblatt (1986). As a result of the intractable nature of these large sized problems, most researchers propose heuristic and meta-heuristic approaches to solve these problems. Some of these approaches are pair-wise exchange heuristic, genetic algorithms (GA), simulated annealing algorithms (SA), ant colony algorithms (ACO) and tabu-search heuristics.

2.1.1 Static facility layout problem

With large sized problems, the SFLP is computationally intractable. Several heuristics have been used to solve the SFLP. These heuristic algorithms can be classified based on their primary function into construction algorithms and improvement algorithms. The construction algorithms develop a layout from scratch while the improvement algorithms improve the objective function of an initial layout that is provided by the analyst (Tompkins et al., 2010). The quality of solution produced by the improvement algorithms is more satisfactory when compared to the construction algorithms; this is because they

start from a feasible solution and then improve on them (Singh and Sharma, 2006). Armour and Buffa (1963) developed a Computerized Relative Allocation of Facilities Technique (CRAFT). CRAFT is an improvement heuristic algorithm which has been used to solve the static facility layout problem.

The CRAFT starts with an initial layout (actual layout of an existing facility or a prospective layout developed by another algorithm (Tompkins et al., 2010)) and improves it. The CRAFT is based on the pairwise-interchange method; it is the most common improvement heuristic algorithm (Singh and Sharma, 2006). Kuppusamy (2001) reported some of the construction algorithms that have been used in the literature to solve the SFLP. Some of them are ALDEP, CORELAP, MAT, PLANET, FATE, COFAD and SHAPE. These algorithms are used to develop an initial layout for the improvement algorithms. A detailed review on the solution methodologies for SFLP can be found in (Singh and Sharma, 2006).

2.1.2 Dynamic facility layout problem

The research on the solution approaches for the DFLP is recent when compared to the SFLP. Balakrishnan and Cheng (1998) reviewed the literature on DFLP, a more recent and detailed review of the literature can be found in Drira et al (2007). Rosenblatt (1986) presented the first major solution methodology for the DFLP. He proposed a dynamic programming algorithm to solve the dynamic facility layout problem with equal size departments. This approach solved the DFLP optimally for only small problems in reasonable time. For large sized problems, it becomes computationally intractable.

As a result of the computational complexity of the large-sized dynamic facility layout problems, many researchers have proposed several approximate solutions. Drira et

al. (2007) classified the meta-heuristics algorithms for DFLP into two: the evolutionary algorithms (genetic algorithms and ant colony algorithms) and the global search algorithms (tabu search algorithms and simulated annealing algorithms). Conway and Venkataramanan (1994) examined the suitability of the genetic algorithms for the dynamic facility layout problem with a budget constraint on the rearrangement of the departments. Balakrishnan and Cheng (2000) developed an improved genetic algorithm for the dynamic facility layout problem. This algorithm differs from that proposed by Conway and Venkataramanan (1994) in three ways: the crossover operator that was employed, the introduction of genetic mutation and generational replacement approach. Both mutation and generational replacement approach were introduced to increase the population diversity and avoid the termination of the algorithm to local optima.

Baykasoglu et al. (2006) proposed an ant colony optimization algorithm for solving budget constrained and unconstrained dynamic facility layout problems. This work was the first approach to solve a budget-constrained dynamic facility layout problem using the ant colony optimization algorithm. McKendall et al. (2006) proposed two simulated annealing algorithms for the dynamic facility layout problem. The first simulated annealing algorithm (SA I) is a direct adaptation of the original SA algorithms which is similar to that proposed by Baykasoğlu and Gindy (2001) while the second simulated annealing algorithm (SA II) is just like SA I but the look-ahead/look-back strategy was incorporated.

Another meta-heuristic approach for the DFLP is the tabu-search heuristic (TSH). Kaku and Mazzola (1997) proposed a TSH for the dynamic facility layout problem. This heuristic employed several memory features and strategies for search diversification and intensification. A similar tabu-search heuristic was proposed later to obtain an efficient

layout for the DFLP using data envelopment analysis (DEA). The DEA considered cost, adjacency and distance requested (Bozorgi et al., 2015). The reverse strategy was also introduced in addition to the pairwise-interchange for neighbourhood search of the initial layout.

The pairwise-interchange is the most common heuristic algorithm that has been applied to solve the DFLP. Urban (1993) proposed a steepest descent pairwise-interchange for the DFLP. This heuristic is similar to CRAFT (Armour and Buffa, 1963) except for the rearrangement cost that was incorporated in the formulation. This heuristic also considered the forecast windows, *m*, to find different sets of good layout plans for the planning horizon. Balakrishnan et al. (2000) later developed an improved pairwise exchange heuristic for the dynamic facility layout problem. This heuristic was an improvement of the work done by Urban. The improvements were backward pass of the solution obtained from Urban's pairwise-interchange heuristic and combining the heuristic developed by Urban with dynamic programming.

Hybridization of different heuristics and meta-heuristics is also another common approach used to solve the dynamic facility layout problem. Some of these hybrid heuristic and meta-heuristic algorithms found in the literature are hybrid genetic algorithm proposed by Balakrishnan et al. (2003); hybrid ant systems proposed by McKendall and Shang (2006); hybrid multi-population genetic algorithm for the dynamic facility layout problem (Pourvaziri and Naderi, 2014); and the development of a new data structure for solution representation in a hybrid ant colony optimization algorithm (Chen, 2013). A detailed review of these solution methodologies are described in section 2.3 of this chapter.

2.2 Models and Formulations for Facility Layout Problem

In solving the facility layout problem, they were being formulated based on several models. These models are used to express the complex relationship between the variables in the problem (Drira et al., 2007). These models and formulations (Drira et al., 2007; Singh and Sharma, 2006) were studied extensively in this work to generate a classification for the facility layout problem. These classifications are as follows:

1. Based on the principles that these formulations rely on: graph theory, mathematical model and neural network.
2. Based on the objective of the formulation: single and multi-objectives.
3. Based on the manner in which the problem is formulated: discrete and continuous.
4. Based on the area of the facility departments: equal area and unequal area.
5. Based on the material flow: static and dynamic.
6. Fuzzy formulation

The classification of the facility layout problem and its formulations generated from the extensive study of the FLP literature is shown in Figure 2.1 below.

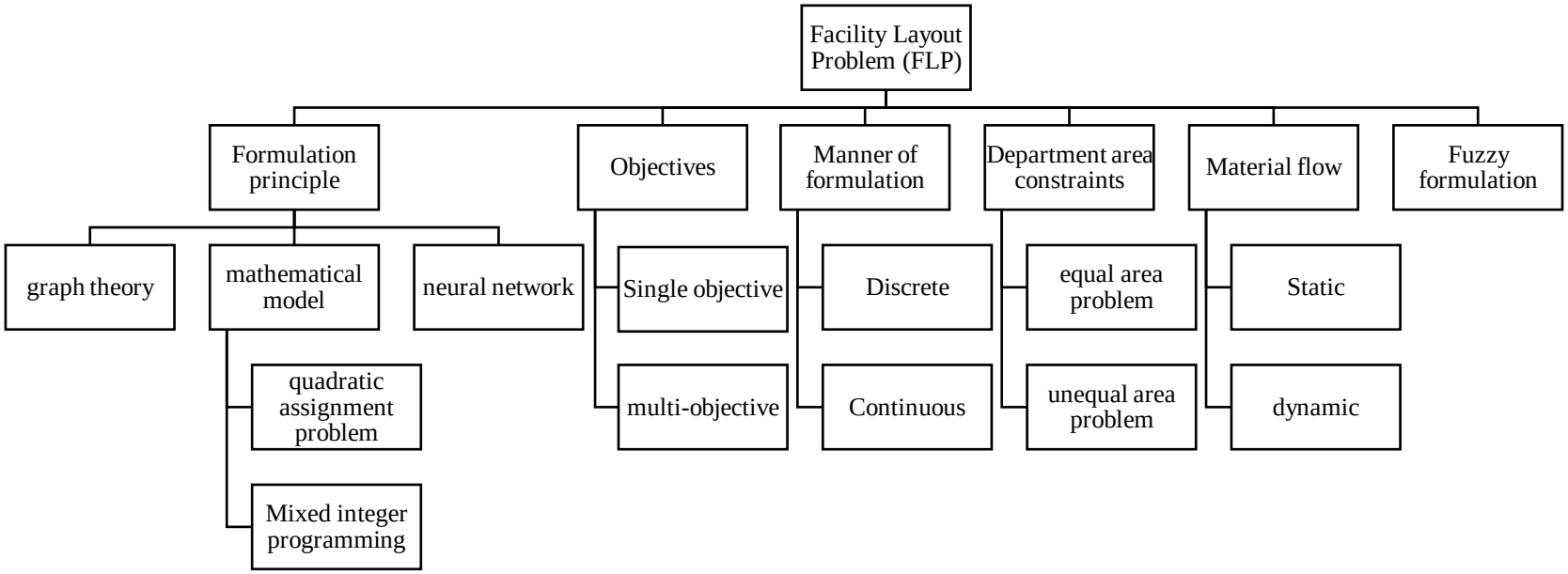


Figure 2.1: Classifications of the FLP and its formulation (Drira et al, 2007)

2.2.1 Quadratic assignment problem

When a discrete assumption is made for the facility layout problem, the quadratic assignment problem (QAP) is the model adopted. It is the most common formulation used by researchers for the layout problem (Singh and Sharma, 2006). To mention a few Balakrishnan et al. (2003), Baykasoğlu and Gindy (2001), Baykasoglu et al. (2006), McKendall et al. (2006), Kaku and Mazzola (1997), Bozorgi et al. (2015) adopted the QAP formulation (discrete formulation) for DFLP. The QAP Formulation for the facility layout problem was first used by (Koopmans and Beckmann, 1957). For the QAP formulation, the facility space is divided into equal rectangular blocks. These blocks are the locations and each department is assigned to one location (Fruggiero et al., 2006). Balakrishnan et al. (1992) proposed a QAP formulation for the constrained dynamic facility layout problem (CDFLP) where the amount of funds available for layout rearrangement is limited by the predetermined budget. The formulation below is for Balakrishnan et al. (2003) and it was adapted from the CDFLP (Balakrishnan et al., 1992):

$$Min\ Z = \sum_{t=1}^{T}\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{N}\sum_{l=1}^{N} f_{tik}\,\mathrm{d_{tjl}X_{tij}X_{tkl}} + \sum_{t=2}^{T}\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{N}\sum_{l=1}^{N}\mathrm{A_{tijl}\,Y_{tijl}} \quad (1)$$

Subject to

$$\sum_{i=1}^{N} X_{tij} = 1, \quad j = 1, \dots, N, \quad t = 1, \dots, T \quad (2)$$

$$\sum_{j=1}^{N} X_{tij} = 1, \quad i = 1, \dots, N, \quad t = 1, \dots, T \quad (3)$$

$$Y_{tijl} = X_{(t-1)ij}X_{til}, \quad i, j, l = 1, \dots, N, \quad t = 2, \dots, N \quad (4)$$

Where

$N =$ number of departments and locations

$T =$ number of time periods in the planning horizon

$i, k =$ departments in the layout

$j, l =$ locations in the layout

$f_{tik} =$ flow cost from department i to k

$d_{tjl} =$ distance from location j to l

$Y_{tijl} =$ the 0,1 variable for shifting i from j to l in period t

$X_{tij} =$ the 0,1 variable for locating department i at location j in period t

$A_{tijl} =$ the cost of shifting from j to l in period t

Equation (1) is the objective function and it is used to minimize the sum of the material flow and rearrangement cost between the departments. Equation (2) and (3) states that each department must be located and each location must be occupied in every period. Equation (4) adds the rearrangement cost to the material flow cost if a department is shifted between locations in consecutive periods.

2.2.2 Mixed integer programming

Unlike the discrete formulation for the facility layout problem, the continuous formulation does not require the facility space to be divided into equal rectangular blocks. The departments are placed anywhere within the space available without overlapping (Dunker et al., 2005). According to Tompkins et al. (2010), for the continuous formulation; just the coordinates of the centroid and sides are needed to define a department's location

and shape. Hence, the facility layout problem may be formulated as a mixed integer programming (MIP) problem. The MIP formulation can be found in Lacksonen (1994) and Dunker et al. (2005).

2.3 Methodologies for Equal Area Dynamic Facility Layout Problem (EA-DFLP)

2.3.1 Exact approaches

Rosenblatt (1986) proposed a dynamic programming (DP) approach to solve the dynamic facility layout problem. Using the terminologies in dynamic programming, a stage corresponds to a period and a state corresponds to a specific layout. In his work, a deterministic environment was assumed where the orders for a given finite horizon is known. The representation for the number of departments in a period is 'N' and the possible layouts is given by (N!). For a given number of periods 'T', the total number of possible layouts for the planning horizon is given by $(N!)^T$. He proposed the recursive formula given below to solve the DFLP:

$$L_{tm}^{*} = \min_{k}\{L_{t-1,k}^{*} + C_{km}\} + Z_{tm}^{*}, \quad t = 1, \dots, n \tag{5}$$

$and\ L_{01}^{*} = 0, assuming\ there\ is\ a\ single\ initial\ layout.$

$Where$

$C_{km} = rearrangement\ costs\ from\ layout\ A_k\ to\ layout\ A_m,\ where\ C_{A_kA_k} = 0$

$Z_{tk} = material\ handling\ costs\ for\ layout\ A_k\ in\ period\ t$

$L_{tk}^{*} = minimum\ total\ costs\ for\ all\ periods\ up\ to\ t,$

$where\ layout\ A_k\ is\ being\ used\ in\ period\ t$

The following assumptions were made in this formulation: rearrangement cost is fixed and it is independent of both the distance between the locations and the period which it occurs; the assignment cost of a department to a location is negligible. As the value of 'N' increases, the solution becomes computationally intractable. Therefore, he proposed simplifying procedures. The recursive formula (5) above can be used to generate both optimal or a heuristic solution depending on the number of layouts used. If all the possible layouts for each period are considered to solve the problem, an optimal solution is obtained. However, the problem is computationally complex with large values of 'N'; consequently he suggested two heuristic approaches.

The first approach is similar to that proposed in Ballou (1968) heuristics for the warehouse location problem. Here, the best solutions in all periods are considered for each period to solve the dynamic programming problem. Therefore, it involves solving the SFLP optimally for all periods. For example, in a five-period problem; four best layouts in each period were considered as the states of the dynamic programming, resulting in 20 states in total that were considered for each period to solve the problem.

For the second approach, layouts were randomly generated for each period. All randomly generated layouts were considered for each period to solve the problem. For example, in a 5-period problem; 10 layouts were randomly generated for each period, and then the dynamic programming problem was solved considering all 50 states for each period. The first approach gave better results when the heuristics were tested using a 6 department and 5 period problem. Rosenblatt suggested that CRAFT or COFAD could be used to generate the layouts in place of the randomly generated layouts.

2.3.2 Heuristics for EA-DFLP

2.3.2.1 Pairwise interchange heuristic

Urban (1993) proposed a steepest descent pairwise interchange for the DFLP. This heuristic approach for DFLP is similar to CRAFT (Armour and Buffa, 1963) except for the rearrangement cost that was incorporated in the formulation. The rearrangement cost was incorporated to accommodate the multi-period planning horizon. This heuristic is an improvement type, which starts with an initial solution and improves on it. The initial layout could be an existing one or a suggested layout. This approach also considered the forecast windows, *m*, to find different sets of good layout plans for the planning horizon.

The forecast window represents the number of periods considered when the pairwise interchange is done. The material flows of one or more periods (depending on the forecast window) are considered to determine the current layout. For example, if the forecast window is 2, the material flow for period 1 and 2 are used to perform the pairwise exchange for period 1. The best layout for period 1 forms the initial layout for period 2 and so on. This is known as the look-ahead strategy. The number of forecast windows is equal to the total number of periods 'T'. Hence, the best of 'T' multi-period layout plan which minimizes the total cost (material handling and rearrangement cost) is chosen as the final layout. The pairwise interchange is performed by exchanging a pair of departments in the layout. The total number of possible pairwise interchanges for a given single layout is given by $\binom{N}{2}$. Where 'N' is the total number of departments in the layout.

Balakrishnan et al. (2000) proposed an improved pairwise exchange heuristic for the dynamic plant layout problem. This heuristic was an improvement of the pairwise exchange proposed for DFLP by Urban (1993). Two improvements were proposed to

Urban's work: backward pass for the solutions generated with Urban's heuristics and combining Urban's heuristics with dynamic programming. For the first improvement, the Urban's method was used to solve the problem; then a backward pass was done for the solutions. This method generates 'M' backward pass solutions and the best of these 'M' solutions is selected as the final solution.

The second approach combines the heuristics proposed by Urban with (Rosenblatt, 1986) dynamic programming. The multi-period layout solutions generated from Urban's heuristic is equal to the total number of forecast windows 'M'. Since the number of forecast windows 'M' is equal to the total number of periods 'T', there are $M \times T = T^2$ possible single layouts. These single layouts are used as the states in the Rosenblatt's dynamic programming. The proposed heuristic was tested with problems that have 6, 15, and 30 departments in a 5 and 10 period planning horizon. The results show that the suggested heuristics both performed better than Urban (1993) steepest pairwise exchange.

The improved pairwise interchange heuristic (Balakrishnan et al., 2000) was applied in Balakrishnan and Cheng (2009) while incorporating rolling horizons and forecast uncertainty. To investigate the effect of the forecast uncertainty, the shifting cost and material flow were varied by subjecting them to negative, zero and positive forecast error. The rolling horizon experiments were simulated by rolling a 5 period problem within a 10 period problem. For example, periods 1-5, 2-6, 3-7 and so on to represent a five-period problem without a fixed planning horizon. The heuristic was tested with problems in the literature and it was concluded that the rolling horizon had an effect on the performance of algorithms.

2.3.2.2 Other heuristics

Erel et al. (2003) proposed a heuristic that is in 3 phases. Phase 1- each period is solved as a static facility layout problem optimally and the best layouts were considered as states for dynamic programming. This was suggested in the first approach of Rosenblatt (1986). Phase 2- the authors cast the dynamic facility layout problem as a shortest path problem (SP) on a network and the SP was solved with dynamic programming technique. Phase 3- local improvement of the solution obtained in phase 2 was done using pairwise exchange. Lacksonen and Enscore Jr (1993) modified five heuristics to solve the dynamic facility layout problem. The heuristics modified are: dynamic programming presented by Rosenblatt (1986), branch and bound algorithm for the QAP by Pardalos and Crouse (1989), cutting Plane algorithm for the QAP developed by Burkard and Bönniger (1983), cut trees by Gomory and Hu (1961) and CRAFT by Armour and Buffa (1963).

2.3.3 Meta-heuristics for EA-DFLP

2.3.3.1 Genetic algorithms

Conway and Venkataramanan (1994) proposed a genetic search technique (CONGA) to solve the dynamic facility layout problem. This research demonstrates how the genetic search technique can be used to solve the constrained dynamic plant layout problem (CDPLP) proposed by Balakrishnan et al. (1992). The CDPLP formulation is a QAP for DFLP with a budget constraint on the shifting of departments. The amount of funds available for layout rearrangement is limited by the predetermined budget.

For crossbreeding, two strings (2 multi-period layouts) are selected based on their strength in the population. A random splicing position is selected and the strings are split. The substrings to the right of the split are swapped. If the random split position falls in the

middle of a period, then the stronger string retains its partial layout while the substring to the right of the weaker string fills the unassigned facilities that correspond and are feasible. The substring to the right of the stronger string fills the unassigned facility of the weaker string that corresponds; with the partial layout of the weaker string subjected to change for feasibility.

The string with lower cost is allowed to survive to the next generation. The algorithm was tested with problems from literature and it performed better than Rosenblatt's dynamic programming for 6 and 9 department problems. A major disadvantage of this genetic search is the infeasible layouts that it generates. A department may occur more than once in a layout, requiring further swaps to be made to generate a feasible solution (Balakrishnan and Cheng, 2000). As a result of these demerits, Balakrishnan and Cheng (2000) developed a nested-loop genetic algorithm (NLGA) to solve the problem of infeasible solutions. The genetic algorithm they proposed differs from the existing algorithms in the following ways: they adopted a point to point crossover operator; they introduced mutation and a new generational replacement strategy to increase the diversity of the population.

A point to point crossover was used in place of the random splitting. After every crossover, a feasibility check is done to eliminate infeasible layouts. The best child replaces the worst parent in the population. That is, the child with the minimum cost replaces the parent that has the maximum cost in the population. The mutation was also done on the best child to ensure population diversity. The mutation operator employed involves randomly choosing a period, and then two departments in the period are randomly selected for swapping. The process is terminated when the difference between the best children in

two successive generations is less than a threshold value. The crossover operator and the mutation make up the inner loop. The outer loop consists of parents from the old population, best children that replaced worst parent and some randomly generated layouts. The outer loop ensures that the inner loop works on different population for different generations. The results from the tests showed that the proposed algorithm performed better than CONGA.

2.3.3.2 Simulated annealing algorithms

Simulated annealing is a stochastic neighborhood search technique that may accept non-improving solutions with a probability to avoid the algorithm from terminating at local optima (Baykasoğlu and Gindy, 2001). The acceptance of the non-improving solution is dependent on the temperature. Temperature is the control parameter and as it decreases the probability of acceptance reduces (Baykasoğlu and Gindy, 2001). Baykasoğlu and Gindy (2001) developed the first simulated algorithm approach to solve the dynamic facility layout problem. In this technique, the neighborhood search is carried out by randomly selecting a period of the initial solution. Then two locations of the selected period are randomly selected for swapping. If the total cost (sum of material handling and shifting cost) decreases, then the new solution is accepted. If not, the non-improving solution may be accepted with a probability depending on the temperature. This increases the search space to ensure a global optimisation.

After a predetermined number of moves at a temperature level, the temperature is lowered, and then further iterations are performed. The algorithm terminates when the predetermined objective or parameters are reached. These parameters could be no improvement between successive iterations, the maximum number of iterations, or a final

temperature. The parameters that control the acceptability of a non-improving solution are the cooling schedule, the initial temperature, and the rate of cooling. The proposed algorithm was tested with problems from Balakrishnan and Cheng (2000) and it performed considerably well. Problems from literature were also tested and compared to known solutions of Rosenblatt (1986) dynamic programming and genetic search of Conway and Venkataramanan (1994). This algorithm performed better than both of them.

McKendall et al. (2006) proposed two simulated annealing heuristics to solve the dynamic facility layout problem. The first simulated annealing (SA I) is a direct adaptation of the simulated annealing to DFLP, just like that proposed by Baykasoğlu and Gindy (2001). The second simulated annealing incorporates a look-ahead and a look-back technique. If the objectives function (total cost) is improved after carrying out the neighborhood search, the improved solution is updated and the same departments are swapped in the preceding and succeeding period. The look-ahead/look-back strategy continues till there are no more periods to consider or when the solution is rejected. The temperature is adjusted, and the procedure is repeated all over again. The proposed algorithms were tested with problems from Balakrishnan and Cheng (2000). The solutions were compared to that from the simulated annealing proposed by Baykasoğlu and Gindy (2001); hybrid genetic algorithm proposed by Balakrishnan et al. (2003); hybrid ant system by McKendall and Shang (2006); and dynamic programming by Erel et al. (2003). The SA II heuristics performed considerably well and better than the other heuristics.

2.3.3.3 Ant colony algorithms

Baykasoglu et al. (2006) proposed the first ant colony optimisation algorithms (ACO) to solve the DFLP with budget constraint. They adopted the formulation for the

CDPLP proposed by Balakrishnan et al. (1992). The facility is only shifted when the budget has accumulated the necessary resources. The ACO proposed starts with an initial solution and the quality of this initial solution has a great effect on the computational time. A random solution is generated for the first period and this is used for all periods to ensure that rearrangement cost at the start of the procedure is zero.

An improvement procedure based on pairwise exchange technique is performed on the initial solution. If the objective function improves, the new solution is accepted. The new solution is updated as the best solution and the pheromone trail is deposited. For subsequent iterations, the solution is generated randomly based on the pheromones. If the best solution cannot be improved after a predetermined number of iterations R, all pheromone trails are erased and the heuristic is started all over again with the best solution alone. The algorithm is terminated when the max iteration is reached or the target objective function is met. The algorithm was tested with problems from Balakrishnan and Cheng (2000) and the results were compared to known results from literature: Balakrishnan and Cheng (2000), Conway and Venkataramanan (1994) and Erel et al. (2003). The algorithm performed better than all except the dynamic programming and simulated annealing heuristics proposed by Erel et al. (2003).

2.3.3.4 Tabu search heuristics

The tabu search heuristic starts with an initial solution and the neighbourhood is being explored using the steepest descent pairwise exchange heuristic. For the pairwise exchange, total number of possible exchange of department pairs is given by $\frac{N(N-1)}{2} \times T$. Where N is the number of departments in the facility and T is the total number of periods

in the planning horizon (Bozorgi et al., 2015). For each exchange done, the objective function (total cost; sum of shifting and material flow cost) is determined and the solution is updated if the total cost reduces. The interchange continues until a non-improving solution is obtained. This heuristic may lead to local optima (McKendall Jr and Liu, 2012).

To avoid the algorithm from terminating at local optima, tabu search does not terminate once a non-improving solution is found. This heuristic iterates through non-improving solutions. The only restriction here is that recent moves are forbidden with help of the tabu list. Tabu list contains the status of recent moves. The tabu list has specified length; so a move can only be tabu for only a number of iterations. The tabu restriction is overridden if a recent move improves the objective function (McKendall Jr and Liu, 2012). This is called the aspiration criterion (Kaku and Mazzola, 1997).

McKendall Jr and Liu (2012) proposed three tabu search heuristics: TS1, TS2, and TS3. TS1 is a basic tabu-search heuristic, just like the approach explained above. TS2 uses the diversification and intensification strategies to obtain a near optimum solution for the DFLP. The diversification strategies used include dynamic tabu length, frequency-based memory and the penalty function. To ensure that this heuristic explores different regions of the search space, the tabu tenure length varies between an upper and a lower boundary depending on the percentage improvement of the objective function (percentage reduction in total cost). The frequency-based memory traces and records repeated movements while the penalty function penalizes non-improving moves. The penalty value is equal to the product of the penalty parameter ‘a’ and the frequency of movement. If the solution is improving, the penalty value is zero. Otherwise, the total cost of the move is a sum of the actual total cost and the penalty value. Decreasing tabu length and fixing pairs of

departments are the two intensification strategies used in this approach. The third heuristic proposed TS3 is a probabilistic tabu search heuristic. The moves are evaluated and ranked; then the top moves are considered and accepted with a probability in the order of their ranking.

The tabu search heuristic proposed by Bozorgi et al. (2015) is similar to the TS2 by McKendall Jr and Liu (2012). The focus of this work was to develop an efficient layout considering a multi-objective function. The criteria were total cost, adjacency and distance requested. This approach incorporated the reverse strategy to the pairwise exchange and also used the data envelopment analysis (DEA) to obtain the most efficient layout. Total cost was used as the input while adjacency and distance requested were the output. If more than one efficient layout is obtained, the one with the lowest cost is chosen.

Tabu search heuristic proposed earlier by Kaku and Mazzola (1997) also makes use of the diversification and intensification strategies. These strategies differ from those proposed by McKendall Jr and Liu (2012). For the diversification strategy, this heuristic uses multiple starting solutions that were specially constructed such that important difference is obtained. More than one tabu length is used for an initial solution to intensify the search. Here, a tabu occurs when two departments are returned to a location that they previously occupied, while in McKendall Jr and Liu (2012), a move is tabu is when it has just been recently done. The termination criterion also differs; the former terminates when the maximum iterations and or maximum number of consecutive non-improving moves is reached while the latter terminates after a specified run time. Results from test carried out on the problem from literature showed that the TS2 outperformed the other heuristics on

DFLP. The limitation of the TS2 is its computational time and the difficulty in writing the codes.

2.3.4 Hybrid approaches for EA-DFLP

Hybridization of heuristics and meta-heuristics has been reported to generate better solutions for the DFLP. Balakrishnan et al. (2003) proposed a hybrid genetic algorithm (HGA) to solve the dynamic facility layout problem. This heuristic uses dynamic programming as its crossover operator. Unlike the GAs discussed earlier, this heuristic uses many strings and many crossover points. CRAFT was used for mutation and the initial solutions were generated using Urban's pairwise exchange (Urban, 1993). The heuristic performed better than the existing GAs at the time: Balakrishnan and Cheng (2000); Conway and Venkataramanan (1994).

A unique crossover operator was applied by Pourvaziri and Naderi (2014) in the hybrid multi-population genetic algorithm they proposed for DFLP. In this operator, the departments to the left of the crossover point are applied to the offspring while the rest are filled in the order they occur in the second parent. The mutation is also a little different; two departments are randomly selected and swapped. They used simulated annealing heuristic as a local search when the promising space is found. For better results, the parameter (mutation rate, crossover, population size) tuning was done using the Taguchi method. The heuristic performed relatively well but the performance reduces with increasing problem size.

Three hybrid ant colony heuristics were developed by McKendall and Shang (2006) for the dynamic facility layout problem. The first heuristic (HAS I) is a modification of the hybrid ant system for quadratic assignment problem (HAS-QAP) proposed by

Gambardella et al. (1999). This heuristic makes use of the random descent pairwise exchange to improve the initial solution. The new solution is updated and pheromone trail matrix is initialized if the solution improves the objective function. The algorithm terminates when some predetermined targets are met. The intensification and diversification strategies were employed to explore a wider search space.

The second hybrid ant system (HAS II) is similar to the first one, except that it uses simulated annealing heuristics in place of the random descent pairwise exchange to improve the initial solution. The third heuristic (HAS III) proposed made use of the random descent pairwise exchange just like HAS I but the look-ahead/look-back strategy was added. These strategies looks into the changes in the objective function if the pairwise exchange was done in both the preceding and succeeding periods. The look-ahead/look-back strategy is continued till the exchange no longer improves the solution or when there are no more periods to consider.

Most of the approaches available in literature focus on the search aspect for the DFLP. A different objective was the focus in Chen's hybrid ant system (Chen, 2013). Large DFLPs require longer computational time. Hence, Chen proposed a new data structure for solution representation to improve the computational efficiency. The research looked into encoding and decoding schemes for solution representation such that more solution representations are packed into fewer bytes. Consequently, during the solution swapping and storage, fewer bytes are being worked on resulting in shorter computational time. For example, the 6 department problem was encoded as an integer with a department represented by 4 bits (1 integer is equivalent to 32 bits).

Two of the hybrid ant systems proposed by McKendall and Shang (2006) were modified to propose the binary coded hybrid ant system (BC-HAS) (Chen, 2013). HAS I and HAS III (McKendall and Shang, 2006) were modified to develop the BC-HAS I and BC-HAS II respectively (Chen, 2013). In addition to the pairwise exchange heuristic used for local search in BC-HAS I, a look ahead/look-back strategy was incorporated in the BC-HAS II. The solutions obtained from this research were compared with those from literature and the algorithms outperformed them all. The computational speed is more pronounced for the large DFLPs (30 departments and 10 periods), with BC-HAS being 3 to 8 times faster than those available in literature. Other hybrid heuristics found in literature are the GA and DP proposed by Dunker et al. (2005), the hybrid dynamic programming in Erel et al. (2003). For easy reference, the methodologies for equal-area DFLP surveyed in this work are summarized in Table 2.1.

Table 2.1: Examples of methodologies used for equal-area DFLP

No.	Author	Year	Methodology	Technique
1	Rosenblatt	1986	Exact procedure	Dynamic Programming (DP)
2	Urban	1993	Heuristics	Pairwise Exchange Heuristic
3	Balakrishnan et al.	2000	Hybrid heuristics	Pairwise Ex. and DP
4	Conway and Venkataramanan	1994	Metaheuristics	Genetic Algorithm
5	Balakrishnan and Cheng	2000	Metaheuristics	Genetic Algorithm
6	Baykasoglu and Gindy	2001	Metaheuristics	Simulated Annealing
7	Mckendall et al.	2006	Metaheuristics	Simulated Annealing
8	Baykasoglu et al	2006	Metaheuristics	Ant Colony Algorithm
9	Mckendall and Liu	2012	Metaheuristics	Tabu Search heuristic
10	Bozorgi et al.	2015	Metaheuristics	Tabu Search heuristic
11	Kaku and Mazzola	1997	Metaheuristics	Tabu Search heuristic
12	Mckendall and Shang	2006	Hybrid heuristics	Hybrid Ant System
13	Chen(Chen, 2013)	2013	Hybrid heuristics	Hybrid Ant Colony
14	Balakrishnan et al.	2003	Hybrid heuristics	Hybrid Genetic Algorithm
15	Pourvaziri and Naderi	2014	Hybrid heuristics	Hybrid Genetic Algorithm
16	Erel et al.	2003	Hybrid heuristics	Hybrid DP
17	Lacksonen and Enscore Jr	1993	Heuristics	DP, Branch & Bound, Cut Trees, Cutting Plane, CRAFT

2.4 Unequal Area Problems

More often than not, the departments in manufacturing and service providing facilities are not equal in sizes. The floor space allocated to each department depends on its space requirement. Consequently, the unequal area assumption is a more practical approach to solving the facility layout problem. Since each department has its space requirement, there is a need to arrange a set of departments within a designated area without violating area and shape constraints. This is the unequal area facility layout problem (UA-FLP). Just like the equal area problem, the main objective considered in UA-FLP is to minimize material handling cost which is usually calculated as the multiplication of the flow between department pairs with the rectilinear distance between the department centroids. Other objectives may include safety, closeness ratings and rearrangement cost (in the situation where the problem is dynamic).

The first computerized heuristic to solve the UA-FLP was CRAFT developed by Armour and Buffa (1963). It is an improvement heuristic algorithm that starts with an initial layout (actual layout of an existing facility or a prospective layout developed by another algorithm (Tompkins et al., 2010)) and improves it. The CRAFT is based on the pairwise-interchange method. The objective was to minimize the material handling cost given an initial layout. Many heuristics followed CRAFT in the following three decades. The classical approaches used the grid-based layout or a block arrangement algorithm to create the final layout.

2.4.1 Slicing tree structure (STS)

In the early 1990s, better approaches that commonly use strict rectangular shapes for departments with geometric constraints such as min side length and aspect ratios

dominated the UA-FLP literature. One of such heuristic is the recursive slicing tree algorithm of Tam (1992). Each slicing tree corresponds to a particular layout, and the nodes of the tree contain either a department label in case of a leaf node or a cut node in case of an internal node. Shayan and Chittilappilly (2004) identified that the efficiency of the genetic algorithm in layout problems is highly dependent on the method of coding the relevant features of a layout as a chromosome. They proposed a genetic algorithm named GA.FLP.STS, an implementation of the slicing tree structure that would not require repairing procedures to ensure that the chromosomes represent legal layouts after application of genetic operators. Scholz et al. (2009) associated each node in the tree with a bounding curve to handle department area, shape, and orientation constraints. Wong (2010a) incorporated an adaptive penalty function in the objective function to guide the solutions towards the feasible region.

2.4.2 Flexible bay structure (FBS)

Similar to slicing tree structure but with a smaller search space is the Flexible Bay Structure (FBS) (Konak et al., 2006; Tate and Smith, 1995). In FBS formulation, the placement of departments in a facility layout problem generates bays. The rectangular area of the facility floor is divided into bays that has varying width. Each bay is further divided into rectangular departments that has different length but same width. The term flexible bay comes from the fact that the width of a bay is automatically adjusted by the number of departments contained. Tate and Smith (1995) encoded a solution in two different chromosomes: the first chromosome shows the sequence of the departments bay by bay, read from top to bottom, left to right; the second chromosome contains the number of bays and the break points between bays. Wong (2010b) proposed an ant system for the unequal

area facility layout problem. However, two modifications were made to the FBS when solving problems that has empty spaces: feasible solution space is extended using empty space to fulfill department constraint and the objective function was improved by recursively filling the bay with empty space. FBS is simpler, easier to solve, and bays may be easily become candidates for aisle structures (Konak et al., 2006). However, that comes at the cost of a limited search space.

In the last two decades, these two representations (STS and FBS) became a subject of many research papers and a good tool to demonstrate many modern metaheuristics such as genetic algorithms, simulated annealing, ant systems, tabu search, etc. Despite the dearth number of publications compared to meta-heuristics, exact methods have witnessed significant progress in the last two decades. In 1990, Montreuil developed the first mixed-integer programming (MIP) model for UA-FLP (Montreuil, 1991). Better models have been followed since then, and a 7- (Meller et al., 1998) followed by a 9- (Sherali et al., 2003) and then an 11-department (Meller et al., 2007) problems have been solved to optimality. In practice, we still need more realistic problems of the size of 15, 30, or more. Therefore, computerized heuristics and meta-heuristics are still the preferred solution approach adopted by both researchers and practitioners.

CHAPTER THREE: EQUAL AREA DYNAMIC FACILITY LAYOUT PROBLEM

The facility layout problem has to do with assigning a number of departments in a facility to a number of locations such that some certain objectives are optimised. The dynamic nature of this problem arises when material flow changes from one time period to the other during a planning horizon. An example of a DFLP material flow data is a 5-period and 6-department problem (2×3 facility layout problem) of Rosenblatt (1986) shown in Figure A.1. As shown in Figure A.1, the material flow from department 2 to 1 is 63 units in period 1 while it became 168 units in period 3. Some of the reasons for changing material flow in a facility were described in section 1.1. Also from Figure A.1, it can be stated that there is relatively high material flow into departments 3 and 4 in period 1 while department 5 and 6 experience the highest material inflow in period 4. With changing material flow during the planning period, the facility needs to be reassigned such that the objectives still remain optimized.

In the course of changing the facility arrangement, some shifting costs (rearrangement costs) are incurred (either due to moving the departments or lost production). If these rearrangement costs are relatively low, we tend to change the arrangement of the facility such that it suits the new material flow. If otherwise, the rearrangements are ignored as they would not provide any cost reduction but rather increase the total cost. The bottom row of material flow data in Figure A.1 shows the fixed cost of shifting each department. For example, a cost of 367 units is incurred whenever department 4 is shifted. So if the material flow data for each time period within a planning horizon and the rearrangement cost data are available or we can possibly forecast this data, a multi-

period layout solution that optimizes our objective can be generated using various optimization techniques. In this study, the minimization of the sum of material handling and rearrangement cost is considered.

3.1 Equal Area Formulation for DFLP

The dynamic facility layout problem has attracted the attention of several researchers in the past two decades, with a large proportion of the publications focused on solving the equal area problem. For the equal area assumption, the facility is divided into equal sized locations and each department of the facility can be assigned to one of these locations. The most commonly used formulation for equal area facility layout problems is the quadratic assignment problem (QAP). The QAP is a discrete formulation of the facility layout problem. An actual facility layout is represented in a grid-like structure organised in rows and columns. To represent the dynamic layout, a number of these structures are used, with each structure representing the layout of a time period within the planning horizon. The number of structures used to represent a dynamic layout is given by the number of time periods 'T' in the problem. The QAP formulation below is a modified form of Balakrishnan et al. (1992) proposed by Balakrishnan et al. (2003). This formulation is used in this study to solve the dynamic facility layout problem.

$$Min\ Z = \sum_{t=1}^{T}\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{N}\sum_{l=1}^{N} f_{tik}\,\mathrm{d_{tjl}}\mathrm{X_{tij}}\mathrm{X_{tkl}} + \sum_{t=2}^{T}\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{N}\sum_{l=1}^{N}\mathrm{A_{tijl}}\,\mathrm{Y_{tijl}} \quad (6)$$

Subject to

$$\sum_{i=1}^{N} X_{tij} = 1, \quad j = 1, \dots, N, \quad t = 1, \dots, T \tag{7}$$

$$\sum_{j=1}^{N} X_{tij} = 1, \quad i = 1, \dots, N, \quad t = 1, \dots, T \tag{8}$$

$$Y_{tijl} = X_{(t-1)ij} X_{til}, \quad i, j, l = 1, \dots, N, \quad t = 2, \dots, N \tag{9}$$

Where

$N =$ number of departments and locations

$T =$ number of time periods in the planning horizon

$i, k =$ departments in the layout

$j, l =$ locations in the layout

$f_{tik} =$ flow cost from department i to k

$d_{tjl} =$ distance from location j to l

$Y_{tijl} =$ the 0,1 variable for shifting i from j to l in period t

$X_{tij} =$ the 0,1 variable for locating department i at location j in period t

$A_{tijl} =$ the cost of shifting from j to l in period t

Equation (6) is the objective function and it is used to minimize the sum of the material flow and rearrangement cost between the departments. Equation (7) and (8) states that each department must be located and each location must be occupied in every period. Equation (9) adds the rearrangement cost to the material flow cost if a department is shifted between locations in consecutive periods.

The flow cost is a product of the material flow between each pair of department and the unit cost of material flow between these departments. The rectilinear distance between the department centroids is the distance measure used in this research. Data associated with the material flow between department pairs and the rectilinear distance between the departments centroids were organised in a matrix format called from-to-chart. A typical example is shown in Figure A.1. The following assumptions were made in this study:

1) The area of the locations are equal and each department fit into a single location.
2) The unit cost of material flow between departments is fixed for all department pairs irrespective of the type, weight, shape or size of material flowing between them.
3) The cost of shifting departments is fixed irrespective of the time period the shifting takes place. The shifting cost also does not depend on the distance between the present and former locations of the department.
4) The time value of money is not taken into consideration when estimating the material handling and the shifting (rearrangement) cost.

3.2 The Genetic Algorithm

Genetic algorithms (GA) belong to a class of evolutionary algorithms that mimics natural genetics and uses the principle of "survival of the fittest" to evolve a set of initial solutions into an optimal or a near-optimal solution (Yu and Gen, 2010). GAs are population-based algorithms; initial solutions are essential to kick-start the search process. To use the GA, an individual (a solution) is encoded as a chromosome string and the performance of each individual can be evaluated by an objective function also known as the fitness function. The GA is such that fitter individuals have a better chance of surviving to the next generation. This is called the selection process and it is fundamental to

optimization and convergence of the algorithm. GAs use different variation operators such as crossover (breeding) and mutation to search the solution space. This mimics the genetic changes in living organisms. They use systematic random search approach which makes them efficient in generating sub-optimal solutions in situations where the search space is very large and the exact procedures are inadequate. The equal area DFLP is a complex combinatorial optimization problem with a solution space given as $(N!)^T$. Its complexity increases exponentially with the number of departments (Baykasoglu et al., 2006). The proposed algorithm to solve this problem was developed using GA procedures found in Yu and Gen (2010). These procedures were adopted for DFLP and the steps are explained in the following sections.

3.2.1 DFLP Encoding

To use the genetic algorithms for optimization of this problem, the layout of an individual (a solution) was coded by representing the grid-like structure for each time period with a chromosome string of a length equal to the number of departments 'N'. This is done in a sweeping order; row by row as shown in Figure 3.1. The chromosome string is made up of small units called allele and each allele holds a positive integer that represent the department number. These strings were then joined together to form the dynamic layout chromosome string (See Figure 3.2).

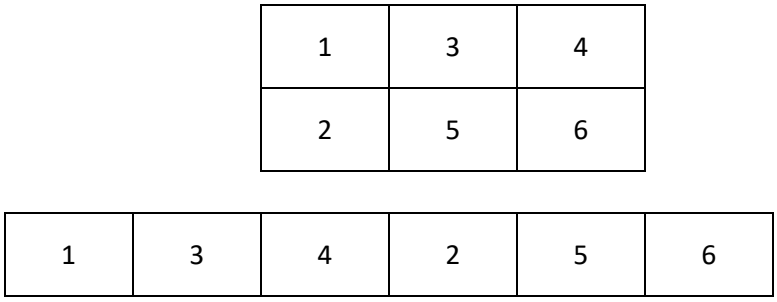


Figure 3.1: A sample grid-like layout for one time period and its coded chromosome

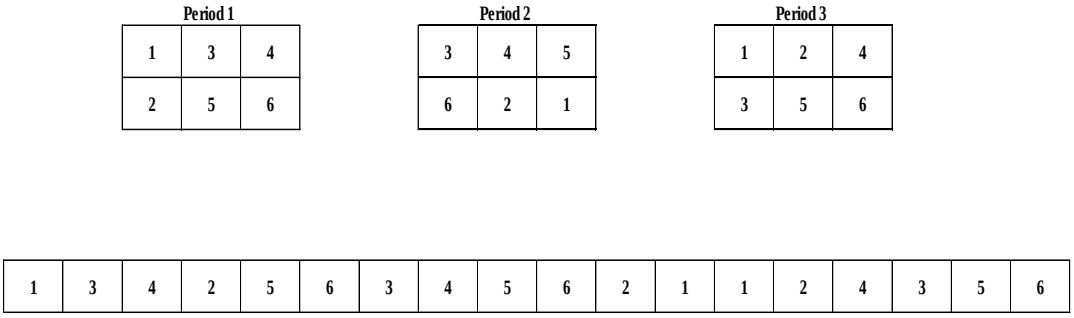


Figure 3.2: Sample grid-like layouts for a 3-period DFLP and its coded chromosome

3.2.2 Initialization

The GA is a population-based algorithm; it carries out its operations on a group of initial solutions. To kick-start this evolving process, initial solutions were randomly generated. A feasible solution is such that a department number is not repeated within a time period. The random generation was used to avoid clustering of solutions around a region of the search space and encourage a global search. Chromosome strings were generated by random permutation of the alleles. These randomly generated individuals make a population and the number of individuals in this population is represented by pop_size. Each individual in the population is a parent and the population is a parent pool. The pop_size for this genetic algorithm is problem dependent as shown in section 3.3.1, 3.3.2 and 3.3.3. Tests carried out in this study show that a pop_size of 500 individuals was efficient for problem I (see section 3.3.1). Therefore, a crucial step in the initialization stage of this algorithm is parameter setting. The following parameters were assigned in the initialization stage: population size 'pop_size', crossover probability 'P_c', mutation probability 'P_m' and maximum number of generations 'max_gen'.

3.2.3 Fitness Function

After the initialization stage, the fitness of each parent in the parent pool were evaluated using the objective function given by equation (6) and (9) while obeying the constraints given by equation (7) and (8). The sequence of numbers on the dynamic layout chromosome string gives the relative location of the departments when they are decoded into the grid-like facility layout. The material handling cost was evaluated using the data provided in the material flow and distance from-to-matrices while the shift cost data was used to evaluate the rearrangement cost. The fitness (evaluated with the objective function-

equation 6) is the sum of material handling and rearrangement costs for a dynamic layout. The DFLP is a minimization problem, therefore, lower total cost implies better fitness. At the point when the objective value of each individual in the initial population is evaluated, a number of generations counter 'gen' is set as zero.

3.2.4 Fitness Transferral and Selection

The main loop of this algorithm starts from the selection stage. For optimization and convergence, the selection process is fundamental. The roulette wheel selection is employed (RWS) in this algorithm. This selection process is such that fitter individuals have higher probability of being selected. The probability of an individual to be selected was represented by its relative fitness. Equation (10) shows how the relative fitness was calculated. If the fitness value of an individual was used directly to obtain its relative fitness, the objective of minimization of total cost will not be met with this algorithm. Individuals with higher total costs would have better chance of being selected using the actual fitness value. In a bid to ensure the minimization objective is met, a fitness transferral was done. In other words, the minimization problem was converted to a maximization problem to use the RWS. The fitness transferral was achieved by subtracting all fitness values from a value higher than the maximum fitness value. Consequently, the individuals with lower total cost now have higher fitness value and selection probability. At this point, it is important to note that an individual solution is described by its chromosome string and fitness value.

$$P_s = \frac{F_k}{\sum_{k=1}^{pop\_size} F_k} \tag{10}$$

Where

P_s *is the probability of selection*

F_k *is the fitness value after transferral*

pop_size is the populations size and k is the individual's number in the population

For crossover (recombination), a mating pool was generated using the roulette wheel selection. The steps for the RWS used are:

Step 1: Generate a cumulative distribution using the relative fitness of the individuals in the order which they were stored in the programming environment.

Step 2: Indicate the upper and lower boundary of each individual in the distribution.

Step 3: Generate a random number R ($0 < R < 1$).

Step 4: Select the individual that this random number falls within its boundaries.

This was repeated pop_size times so that pop_size individuals were selected to the mating pool. With this method, some of the individuals were selected more than once while some were not selected at all. The probability of being selected is an individual's relative fitness.

3.2.5 Crossover Operator

Single point crossover was employed in this algorithm to explore the search landscape. For single point crossover, two individuals are selected randomly from the pool for mating. To pair individuals for crossover in a programming environment, the mating pool created with the RWS was reshuffled. The steps used for the reshuffling are;

Step 1: Generate a random integer permutation between [1, pop_size] and store them as 'randperm' in the programming environment.

Step 2: Then, j = randperm (i). Where 'i' is a counter from 1 to pop_size. The i^{th} element in the randperm array is equal to j.

Step 3: Apply this to the mating pool, such that the i^{th} element in 'randperm' is the j^{th} individual in the mating pool. Select this individual and store according to step 4. Then repeat step 2 and 3 till i = pop_size.

Step 4: Store the individuals in a new variable 'mating_pairs' in the order which they were selected using the counter 'i'. Now, individual 1 and 2, 3 and 4…pop_size -1 and pop_size in the 'mating_pairs' are the crossover pairs.

In this algorithm, not all individuals in the 'mating_pairs' undergo recombination. The single point crossover operator was applied with probability 'P_c'. The crossover probability enables the GA to carry on the good traits in the parent pool to the next generation and ensure convergence. This probability is problem dependent, just like the pop_size. The crossover probability applied to each problem set tested with this algorithm are described in section 3.3.1, 3.3.2 and 3.3.3. The procedure for the single point crossover used in this algorithm are listed below:

Step 1: Generate 'n' random integer permutation between [1, pop_size/2]. Where n = pop_size/2 x P_c. This is the implementation of the crossover probability that was proposed.

Step 2: Select a mating pair in the location given by an element of the randomly permutated array in step 1 and store it in a temporary variable 'crossover_individuals'. Note: do not repeat any element in the permutated array.

Step 3: Generate a random integer between [1, L-1]. Where 'L' is the length of the dynamic layout chromosome string.

Step 4: Slice both mating individuals at this point and swap the alleles to the right of the crossover point to produce two offspring.

Step 5: Repeat step 2, 3 and 4 'n' times to perform the crossover operation on all individuals selected for crossover based on crossover probability.

Figure 3.3 shows the implementation of a single point crossover on a 6-department and 2-period problem. The random splitting point in Figure 3.3 is at the seventh point. The departments to the right of this point were swapped between the two parents. This crossover operator generates infeasible solutions (see Figure 3.3). A solution is infeasible if a department occurs more than once in a time period. As shown in Figure 3.3, department number 3 occurred twice in the second period of first infeasible offspring while department 1 occurred twice in the second period of the second infeasible offspring. A repairing algorithm was developed to make these infeasible solutions feasible by scanning through the offspring generated and replacing departments that has earlier occurred with those that are yet to occur in that time period. The feasible offspring are carried over for mutation.

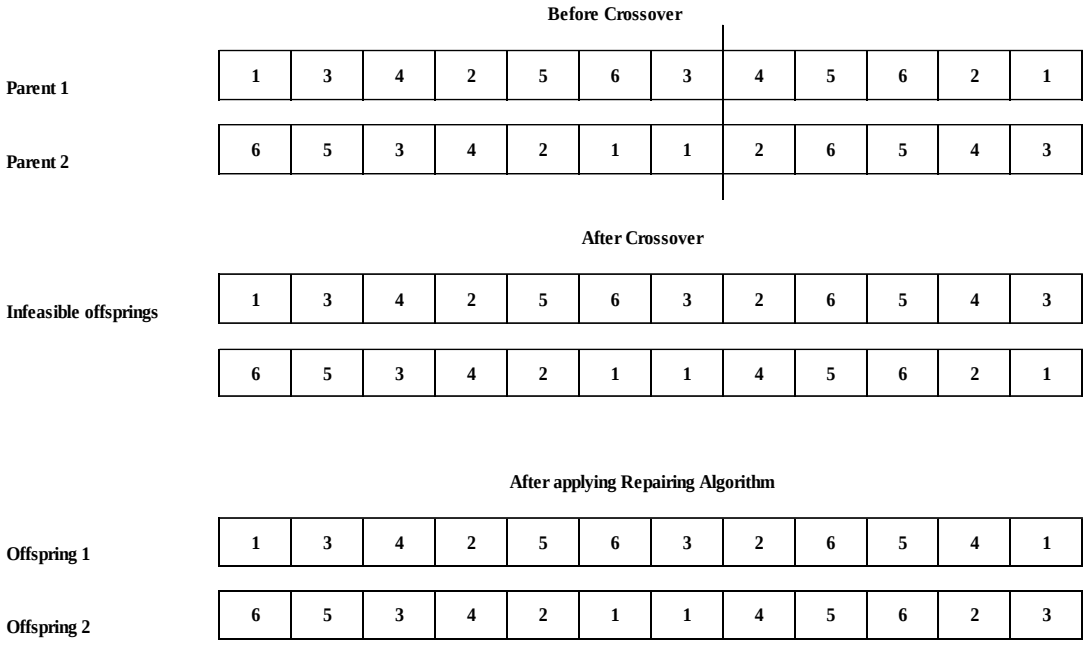


Figure 3.3: Single point crossover operator for a 6-department and 2-period problem

3.2.6 Mutation Operator

Mutation operators, just like the crossover operators were applied in this algorithm to create new individuals. They are mechanisms used for exploring the solution search space. Mutation was applied to create random changes to the population for the purpose of diversity. This prevents the algorithm from terminating at local optima. The allele exchange mutation was used in this study. A time period was selected randomly and two departments in this time period were randomly exchanged. It is important to note that the mutation operator is performed on a single dynamic chromosome string. Figure 3.4 shows an exchange mutation operation carried out on an offspring (offspring is a dynamic chromosome string of a 6-department and 2-period problem). Department 1 and 5 of the first period were randomly exchanged to create the mutant. The mutant replaces the offspring irrespective of its fitness value. The mutation operator was applied to the offspring pool at a probability 'P_m'. The mutation probability applied to each problem set tested with this algorithm are described in section 3.3.1, 3.3.2 and 3.3.3.

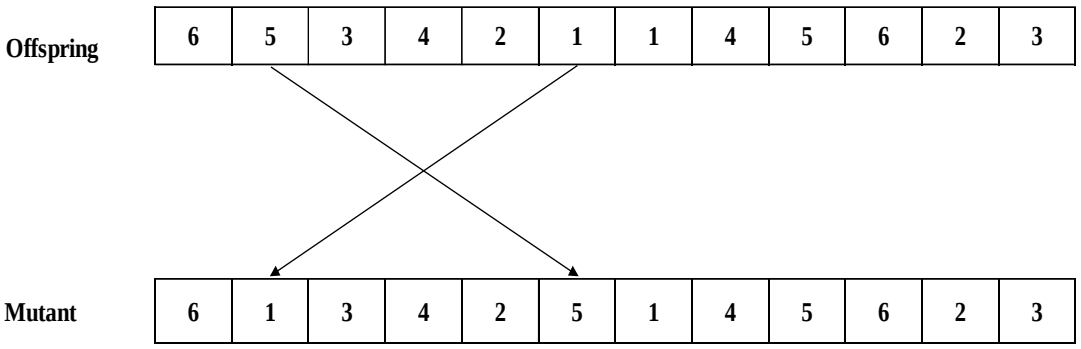


Figure 3.4: Exchange mutation operator for a 6-department and 2-period problem

3.2.7 Replacement and Elitism

Replacement is a type of selection process. The replacement stage was incorporated as a mechanism to decide the individuals that would be selected to the next generation. The concept of elitism in (Deb et al., 2002) was applied in this algorithm. The parent and children population were combined to form a unified population. Pop_size individuals with the best fitness values were then selected to survive to the next generation. The number of generations counter updates with the function 'gen = gen + 1' after replacement. The elitism is fundamental to preserving the best individuals for each generation.

3.2.8 Algorithm Termination

To stop the algorithm, a termination criteria had to be set. The maximum generation criteria was implemented in this algorithm. At 'max_gen' iterations, the algorithm terminates and returns the final pop_size individuals as the results. In other words, all stages in the main loop were iterated until this stopping criteria is met. The algorithm termination point is also problem dependent just like every other parameter for this algorithm. This stopping criteria proved to be efficient for convergence in all problem sets tested. The flow chart shown in Figure 3.5 further describes this algorithm. Details of the parameters used for each problem set are discussed in chapter 5.

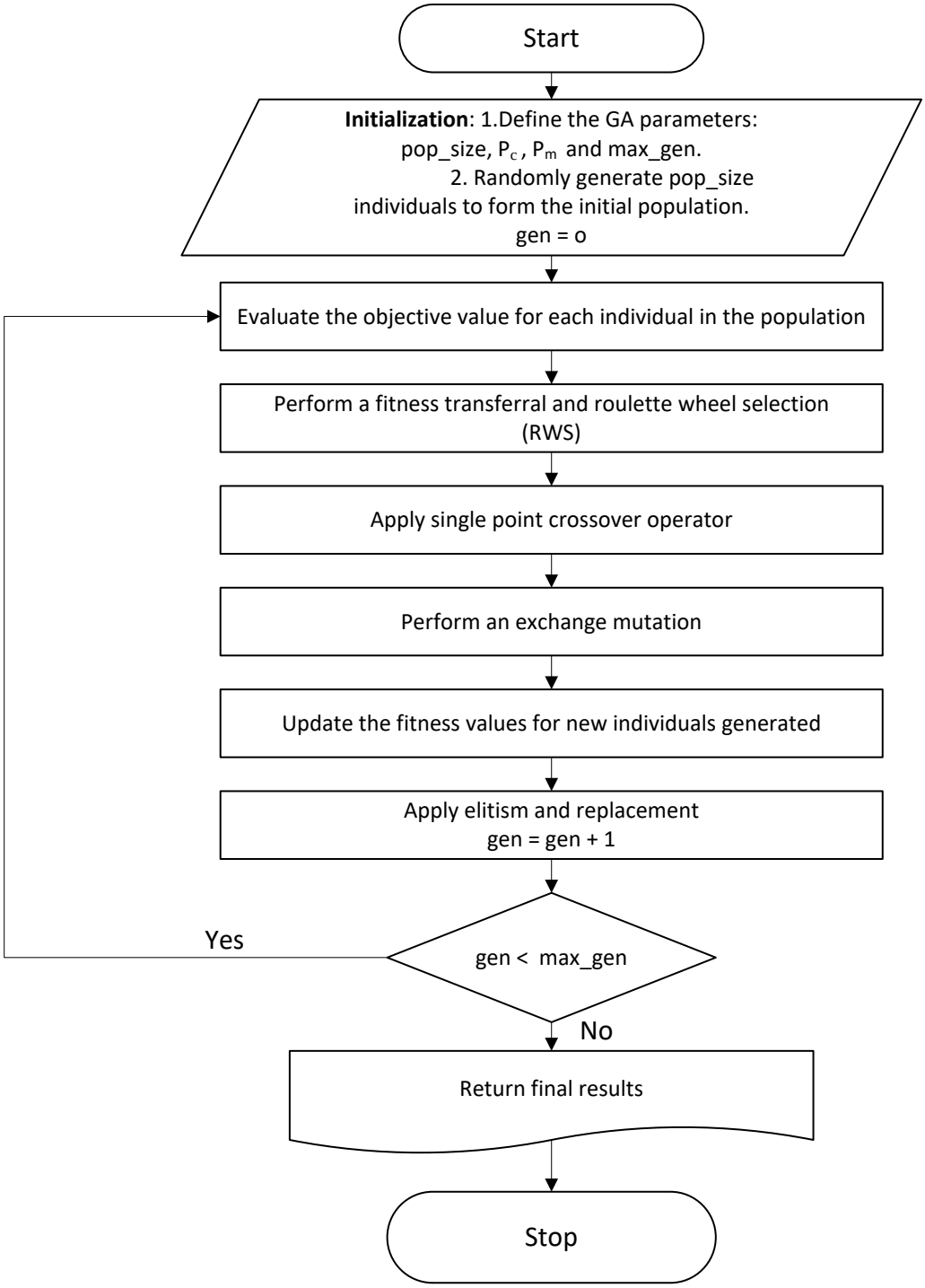


Figure 3.5: The Genetic Algorithm Flow Chart

3.3 Computational Study

The genetic algorithm developed in this study (see section 3.2) was tested on some benchmark problems from the equal area dynamic facility layout problem literature. These benchmark problems were chosen because they are well known and have been used by several researchers for comparison and validation of their algorithms. Two sample problems and one problem set were used to test the efficiency of the genetic algorithm proposed in this thesis (section 3.2). The first problem was taken from Rosenblatt (1986). It is a 6-department and 5-period problem with a 2×3 facility layout. The second problem is a 9-department and 5-period problem with a 3×3 facility layout taken from Conway and Venkataramanan (1994). Problem I and II are given in Appendix A and B respectively. The third data set, which was taken from Lacksonen and Enscore Jr (1993) contains test problems that has 6 departments with 3 and 5 time periods. For each time period (T = 3, 5), there are four problem instances. So there are 8 problems in total for this problem set.

3.3.1 Problem I

This 2×3 facility layout problem with N = 6 and T = 5 taken from Rosenblatt (1986) is given in Appendix A. This problem data contains the material flow, the shift cost and the distance matrix. For this problem, different population sizes 'pop_size' were tested, but the best result was obtained with 500 individuals. The termination criterion of maximum generations 'max_gen' was applied to this problem and 500 generations was sufficient to generate a good solution. Crossover and mutation probability used were Pc = 0.4 and Pm = 0.2 respectively. The best result obtained for this problem using the genetic algorithm is 71,494. This is better than the 71,984 obtained with the dynamic programming algorithm that was proposed by Rosenblatt (1986). The solution obtained is only 0.43% above the

optimal solution reported in Baykasoğlu and Gindy (2001). This proves that the genetic algorithm developed is comparatively efficient for this problem size. The best solution obtained is shown in Figure 3.6.

3.3.2 Problem II

This problem has 9 departments and 5 time periods with a 3×3 facility layout. It was taken from Conway and Venkataramanan (1994). For this problem, the parameter settings that gave the best result are: pop_size = 800 individuals, max_gen = 500, Pc = 0.5 and Pc = 0.2. The problem data is given in Appendix B. The best result found with this genetic algorithm is 636,346. The optimal solution for this problem is not known, but the solution obtained with this algorithm is 4.8% above the best known solution which can be found in Baykasoğlu and Gindy (2001). The best layout obtained for this problem using the genetic algorithm proposed in this thesis is shown in Figure 3.7.

Period 1

2	4	5
1	3	6

Period 2

2	4	5
1	3	6

Period 3

2	4	6
1	5	3

Period 4

2	6	4
1	5	3

Period 5

5	1	4
6	2	3

Total Cost = 71494

Figure 3.6: Best Solution Obtained for Problem I

Period 1

4	1	3
5	2	9
6	7	8

Period 2

4	1	7
5	3	9
8	6	2

Period 3

1	3	6
7	4	9
8	2	5

Period 4

1	3	7
4	5	6
8	2	9

Period 5

5	8	1
4	6	2
3	7	9

Total Cost = 636 346

Figure 3.7: Best Solution Obtained for Problem II

3.3.3 Problem III

This data set contains 8 different problems taken from Lacksonen and Enscore Jr (1993). The problems have 6 departments with either 3 or 5 time periods. Each time period (T = 3, 5) has four problems. Different time periods were chosen to test the effect of changing time periods on the proposed algorithm. The summary of the parameter settings for each problem is shown in Table 3.1. Appendix C shows the data for problem number 1 and 5 with 3 and 5 time periods respectively. The shifting cost is the same for all departments. Other problems have their data in the same format as those presented in Appendix C. A complete data for these problems were obtained from Bozorgi et al. (2015) and they can also be obtained from the author.

To further evaluate the validity of the proposed genetic algorithm, the results obtained from this experimental study were compared to those obtained from cutting plane algorithm (CP) of Lacksonen and Enscore Jr (1993), tabu search heuristic (TS) of Kaku and Mazzola (1997), hybrid ant systems (HAS) of Mckendall and Shang (2006), simulated annealing heuristics (SA) of McKendall et al. (2006), probabilistic tabu search (PTS) of McKendall Jr and Liu (2012) and Tabu search heuristic (TS-NM) of Bozorgi et al. (2015). These heuristics were chosen for comparison for the following reasons: they have the best-known solutions for this problem set and for their high number of citations. In my opinion, these heuristics are the best in the literature of equal area DFLP. Table 3.2 shows the results of the proposed genetic algorithm (GA*) and the aforementioned heuristics. The results for all the benchmark heuristics were taken from Bozorgi et al. (2015).

Table 3.1: Parameter Settings

Problem Size		Problem No.	Pop_size	max_gen	Pc	Pm
No. of Dept. (N)	Time Period (T)					
6	3	1	500	600	0.8	0.2
		2	800	800	1.0	0.2
		3	1000	800	1.0	0.2
		4	1000	1000	1.0	0.4
	5	5	800	250	1.0	0.2
		6	800	400	1.0	0.2
		7	800	800	1.0	0.2
		8	800	250	1.0	0.2

Table 3.2: Results and Comparison for Problem Data taken from Lacksonen and Enscore Jr (1993)

Problem Size		Prob. No.	Best Known	CP	TS	HAS	SA	PTS	TS-NM	GA*	% Impr.
N	T										
6	3	1	267	267	267	267	267	267	267	267	0.00
		2	260	260	260	260	260	260	260	260	0.00
		3	313	363	363	363	363	363	313	313	0.00
		4	251	299	299	299	299	299	251	249	0.80
	5	5	402	442	442	442	442	442	402	402	0.00
		6	556	586	586	586	586	586	556	556	0.00
		7	424	424	424	424	424	424	424	424	0.00
		8	428	428	428	428	428	428	428	428	0.00

CHAPTER FOUR: UNEQUAL AREA FACILITY LAYOUT PROBLEM

In this chapter, we analyze, model, and solve the unequal area facility layout problem (UA-FLP) introduced in section 2.4 from the novel Progressive Modeling perspective. Inspired by the desire to develop a pragmatic approach to system modeling, Progressive Modeling (PM) has been developed as an initiative to address many of the challenges faced in modeling and optimizing complex real-life systems. Progressive Modeling was developed by Mohamed Ismail (Ismail, 2011) while working on his Ph.D. research. It had been first introduced to solve the aggregate production planning problem in (Ismail and ElMaraghy, 2009). PM is a new multidisciplinary modeling paradigm that adopts systems thinking and introduces some new concepts related to problem analytics, modeling, and solution algorithms.

The UA-FLP is concerned with arranging a set of departments within a designated area without violating area and shape constraints. The main objective is to minimize material handling cost which is usually calculated as the multiplication of the flow between department pairs with the rectilinear distance between the department centroids. In Progressive Modeling approach to UA-FLP, a solution is represented using dual graphs: a binary tree and a network graph. A framework composed of several interacting components has been developed to encapsulate the model and its solution algorithm. The framework's modular architecture enables incremental updates of both model and its solution algorithm on an ongoing basis; a progressive one. A layout solution is represented as an object-oriented binary tree where leaves represent individual departments, and internal nodes represent cascaded layouts.

4.1 Progressive Modeling Process

Progressive modeling is a new modeling approach that adopts the systems approach in a novel way for analyzing, modeling, and solving large-scale complex industrial problems. PM draws several principles from Operations Research, Optimization Metaheuristics, Software Engineering, and Object-oriented Systems Analysis and Design.

Step 1: Systemize, componentize, and analyze

The Progressive Modeling process starts by analyzing the problem at hand from a system perspective and decomposing it into several interacting components (Ismail, 2013). This step ends by developing a loosely-coupled component-based model that can be easily developed and maintained. UAFLP component model is illustrated in Figure 4.4. Component models demonstrate the separation of concerns of the interacting components in a black-box communication fashion (Ismail, 2013).

Step 2: Develop the logic that governs

In the second step, and inside these components, the logic that govern the problem formulation, a mathematical model, or a logical one; expert system, fuzzy logic or any other artificial intelligence (AI) model, is implemented. Operations Research defines decision variables, constraints, and objectives as the basic building blocks of mathematical models (Ismail, 2013). PM separates the decision space, from the objective space and defines a new term called model space. Decision variables or search neighborhood define(s) the search space; solution objective or objectives define the objective space; and a problem model that can be progressively updated define the “model space”. With time, models could be further developed to reflect a better understanding of the underlying

problem or underlying system changes or adaptations. Step 2 is problem specific and is concerned with real-world large-scope complex systems.

Step 3: Search for the best alternative

This step is concerned with solution algorithm development. A progressive algorithm compiles algorithms that explore both the search space where different alternatives are found and the objective space where all these alternatives are evaluated according to a certain selection criteria (Ismail, 2013). Progressive Algorithms (PAs) abide by the best practices of component-based software engineering and object-oriented design that create more robust, more efficient, and easily extensible solution algorithms. Solutions are represented using object-oriented design principles. Problem data are easily accessed by soliton objects. Unlike traditional metaheuristics, e.g. GAs, SA, and others, the search process is an informed one. Several moves are created and only the best one is selected. Encoding and decoding operations are either minimized or eliminated entirely. Objective updating may happen just locally. The last two strategies can drastically affect the solution time needed to get similar quality solutions obtained by other metaheuristics.

Step 4: Deliver and Adjust

This step include implementation and presenting a user-friendly interface. All the complexities of the modeling process are hidden from the end solution users. FactDesign is a modular software platform that is being built to demonstrate the progressive models and their solutions. FactDesign is a shorthand for Factory Design. FactDesign has several modules for assembly systems, flexible and reconfigurable manufacturing systems, cellular manufacturing, layout and material handling systems planning, and manufacturing

planning and control. FactDesign is a software propriety of Mohamed Ismail. FactDesign has two major versions: R and I. The R-version is dedicated for research purposes while the I-Version targets the industrial or the non-academic audience.

Step 5: Review and Adapt

The fifth step is to verify and validate the model and solution results. PM allows room for changes, adaptation and coevolution of the results with underlying systems; the objective is not only to optimize the underlying system behavior, but also our understanding or modeling of that system. It involves developing better analytics, better logic, and better algorithms is the constant goal of PM (Ismail, 2013).

The process is a 5-step double looped continually improving process as depicted in Figure 4.1. The five steps are iterated on an ongoing basis not only to get better results, but also to get better models, and better solution algorithms and that's why it was called progressive modeling at the outset.

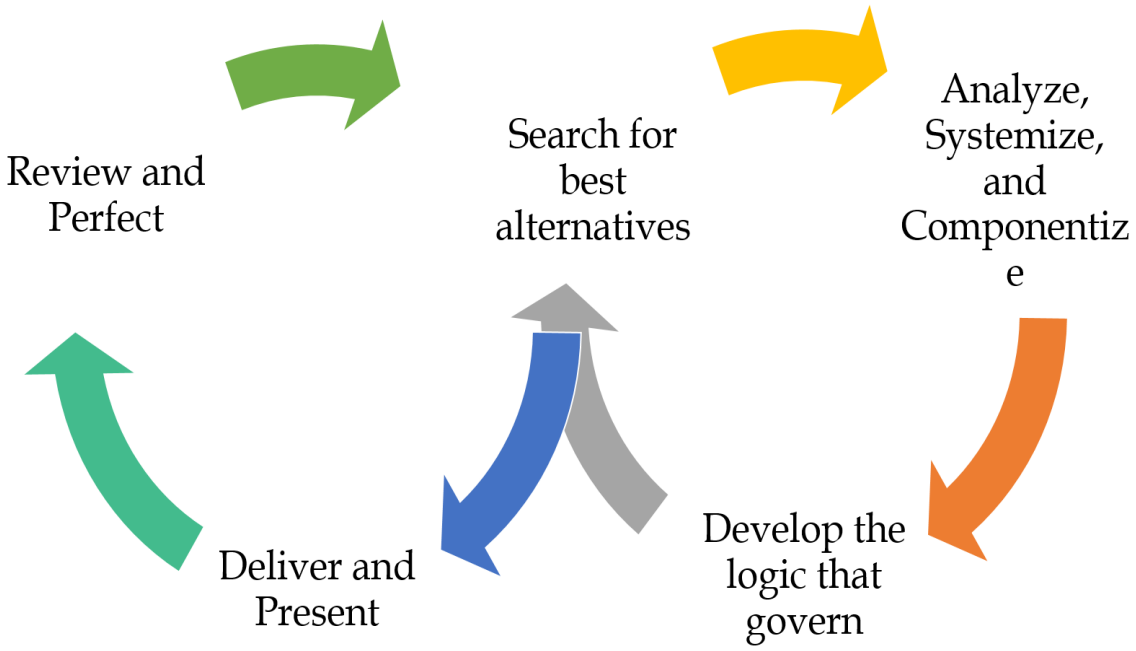


Figure 4.1: Progressive Modeling Process

4.2 UA-FLP Progressive Model

4.2.1 Analytics, Component Model and FactDesign

PM models are implemented as software frameworks. The problem data, the model logic, the search space and the objective space controllers are the main constituents of that framework. The model is defined through the graphical user interface. All progressive models developed are hosted by one unified software platform called FactDesign. FactDesign is a shorthand for factory Design. Many models related to manufacturing systems analysis and design are hosted by the same platform. Figure 4.4 shows the component model for UA-FLP. The component-based design facilitates faster and easier updates, better maintenance, and high reusability of these components. Such architecture will enable us to extend developed model to more generic variants of this problem and integrate it in a wider framework for factory analysis and design.

4.2.2 Solution representation

Progressive Modeling introduces a component model to deploy the problem logic and its solution algorithm over several interacting components. A layout solution is represented by two coupled graphs of a binary tree and a network diagram. A layout graph, a typical problem solution, is represented with object-oriented binary tree T (N, L), whose nodes encapsulate all data needed to define a candidate solution. Its internal nodes define cascaded sub-layouts, and its leaves represent department's layout data: centroids, dimensions and geometrical attributes (min side lengths and aspect ratios). In addition to layout data, the nodes have additional data called the slicing operator which determines the geometry of the sub-layouts. Such novel representation will eliminate encoding and

decoding operations used in many modern optimization metaheuristics such as genetic algorithms, ant systems and the likes.

A network diagram represents the flow and the distance travel ends between departments. The network is a graph G (V, E) with node set V and edge set E, whose vertices represent individual departments and its edges represent the flow and the distance among those departments. The network diagram facilitates evaluation of problem material handling cost. Figure 4.2 shows the tree and the flow/distance graph. Progressive Modeling distinguishes the search space which holds a population of binary trees, and the objective space in which a layout solution is nothing but a figure (a number) to be maximized or minimized. Such a kind of black boxing enables us to use several algorithms for neighborhood movement. This research uses the random wheel selection of genetic algorithms to do this part for single objective problems. If we have a multiple objective problem, algorithms such as NSGA II or SPEA 2 can be utilized. Switching from a single objective problem to a multiple one could be done with a minimal effort.

Since the problem at hand is highly constrained, the search space will be divided into two different spaces: feasible and infeasible ones. Every space will have its own operators which optimizes its performance. So two different types of problems will be solved simultaneously as will be described later in section 4.2.3. The infeasible one will be a subject of some search operators which aim at turning the problem into feasible one, while the feasible space will be governed by a controller which considers the optimization process of feasible ones. The entire model has a graphical user interface that is utilized to define the problem at hand and its design spaces. Figure 4.3 shows the tree, the layout, and the flow/distance graph as displayed on the graphical user interface.

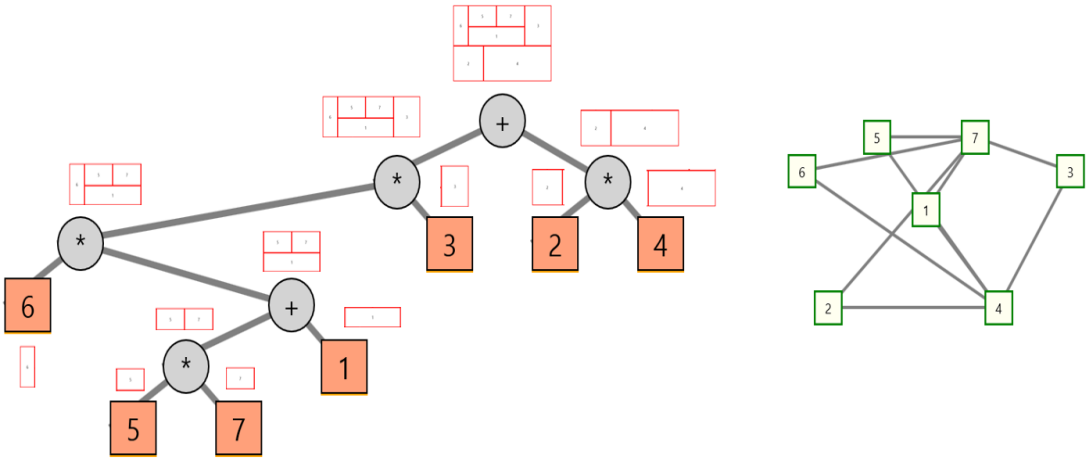


Figure 4.2: The Tree and the Flow/Distance Graph

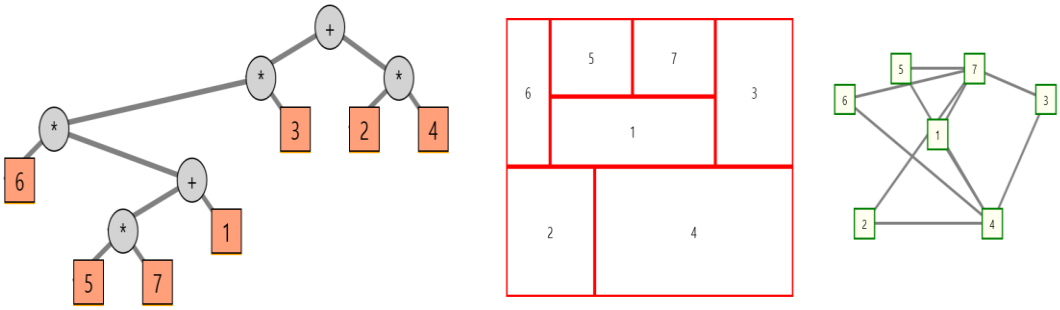


Figure 4.3: The Tree, the Layout, and the Flow/Distance Graph

Figure 4.4 shows the unified modeling language (UML) component diagram for the unequal area facility layout problem (UA-FLP). The model is made up of 6 components. The problem is analyzed and deployed over these modular components. The component-based design facilitates faster and easier updates, better maintenance, and high reusability of these components. Such an architecture will enable us to extend developed model to more generic variants of this problem and integrate it in a wider framework for factory analysis and design.

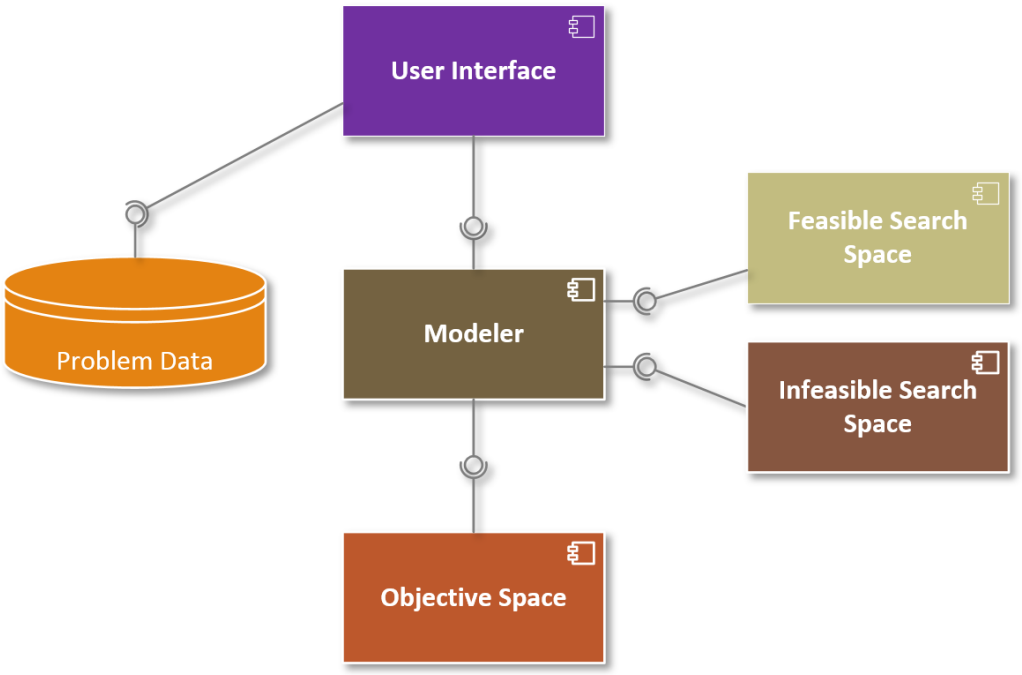


Figure 4.4: Component diagram for UA-FLP

4.2.3 The logic that govern

At center of every component model, there is a ‘modeler’ that connects all the pieces of the puzzle and determines how both the search and selection spaces interact with each other and how the solution algorithm is managed. The modeler receives a copy of problem data, department areas, shape constraints, objectives, etc. and initiate the solution process. Solution candidates are created and evaluated in the search space components. While feasible solution is evaluated based on the material handling criterion, the infeasible ones are evaluated based on a shape violation criterion. Progressive Modeling never ships infeasible solution(s) to the selection space.

4.2.4 Progressive Algorithm

Progressive algorithms generally comprises of three stages:

4.2.4.1 Initialize the search space

Unlike genetic algorithms and the likes, no string representation will be used, no encoding or decoding is required. A solution is a composite of data objects that represent a meaningful problem solution. A solution is a self-contained one in the sense that all the problem data are easily accessible. Simply put, a problem solution is a specialized data structure that has a complete description of what a solution means and has an impact on the way that this solution could be improved. Solutions have names that reflects the problem domain. Solutions are not ants, bees, or individuals as they are called by ant colony optimization, bee colony and genetic algorithms respectively. In this problem, a solution is a layout described by its binary tree and evaluated by its flow network. This philosophy of constructing solutions have important implications on how the search space is structured and how the search process is controlled. Progressive Algorithm are population-based.

The Initialization Algorithm

The solution starts by reading the problem data and initializing the solution process by creating a random list of departments. The tree object is created from the root and keeps expanding randomly until all the departments are assigned. Every node has slicing operator, a list of descendants (labels of node descendants), and a layout placeholder, see Figure 4.2. Such data affluent representation creates what is called informed search space. Solutions generated are classified as feasible or infeasible ones, simply flagged as red or green. Solutions are in a state of continuous movement once they change from one state to the other.

4.2.4.2 Selection process

Solutions are improved via progressive updating where better solutions have high chances of being selected to stay in the updated search pool while weaker ones are supposed to leave the pool. In order to avoid premature search convergence; getting entrapped in a certain local search area, the updating process is probabilistic and is equipped with mechanism that guarantees fair search process. PM imports well-known metaheuristics neighborhood updating or regeneration algorithms such as GAs selection, Simulated Annealing energy function/cooling schedule. However, by using the modular architecture, we can switch among many of the corresponding neighborhood algorithms and choose the best performing one.

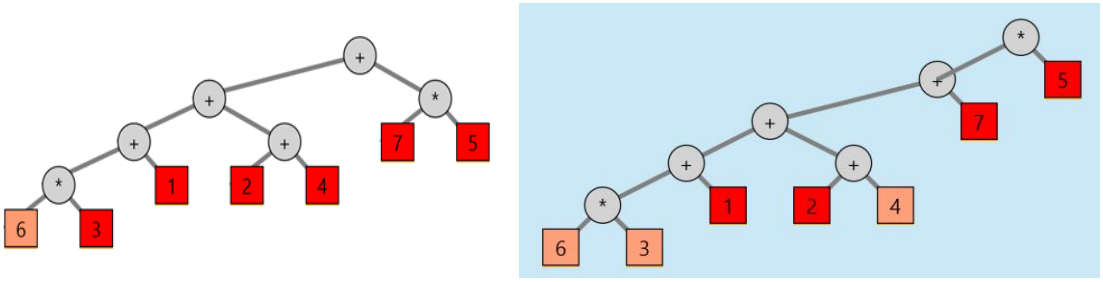
4.2.4.3 Neighborhood Search

The search space is explored using data informed operators. Those operators could be unary operators, binary where two solutions are involved, or n-ary operator where multiple operators can be utilized. Those operators are search space specific. Operators for

feasible search space are designed mainly to create feasible solutions while the others undergo a selection process (see section 4.2.2). Operators for unfeasible solution are made mainly to repair the solutions.

Neighborhood operators (search space moves):

Tree rotations: tree can be rotated to the left or the right to change the layout structure. The next figures shows an example.



Before rotation after rotation

Figure 4.5: Tree rotation operator

Branch Swapping: two branches could be swapped, two departments could be swapped, and a branch could be swapped with a leave. All branch-swapping operations could be done in an informed fashion; two equal or almost equal departments are swapped.

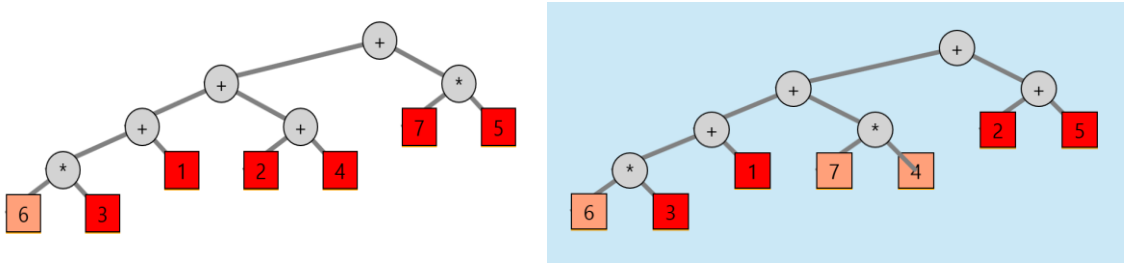
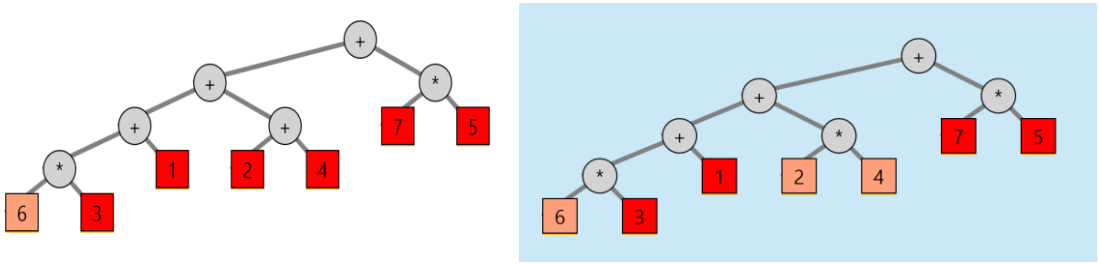


Figure 4.6: Swapping 2 & 7 leaves (departments)

Slicing flipping: slicing operators could be flipped from vertical to horizontal and vice versa.



Before flipping 2 and 4 parent node after flipping 2 and 4 parent node

Figure 4.7: Flipping 2 & 4 parent node

Local updating: using the network graph and after any move, only affected departments are updated. Depending on the depth of the affected nodes, the computational time needed for updating objective function could be saved.

4.2.5 Numerical experiments and discussions

The proposed algorithm was tested using various problem sets taken from the literature. They are O7, O8 and O9 from Meller et al. (1999), vC10 from Van Camp et al. (1992), Ba12 and Ba14 from van Camp (1989), AB20 from Armor and Buffa (1963). The latest has been tested accordingly using several values of aspect ratios. For small size problems, all optimum or best-known solutions have been obtained. For this experiment, we chose the number of iterations to be 10,000 and the population size to be 50. If the best-found solution is not getting any better after 100 iterations, the solution terminates as well.

The results for mid-size problems and AB20 are reported in Table 4.1 and Table 4.2. Figure 4.8 shows the van camp solution obtained.

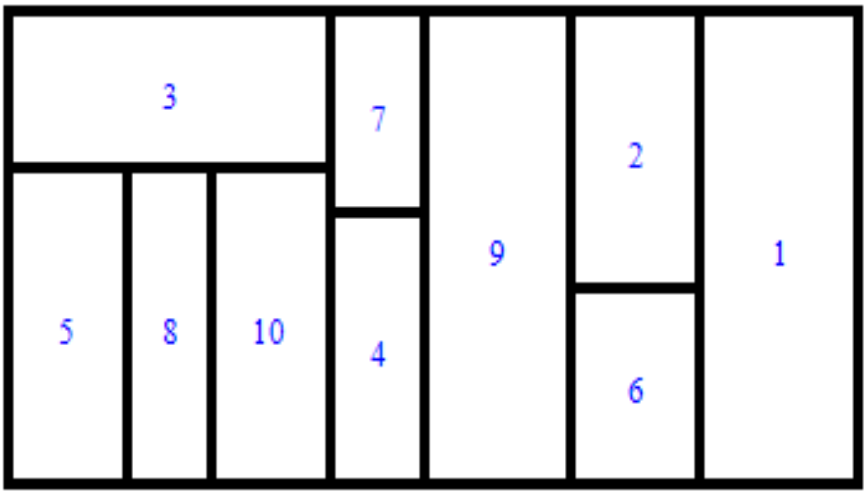


Figure 4.8: Van Camp Solution, TMHC = 19655.3

Table 4.1: The results for Van-Camp, BA 12 and BA 14

Author	**Problem**		
	Van Camp	**BA12**	**BA14**
Bazaraa	24475	24475	8170.5
Van Camp NLT	20472	20472	6875
Hassan's SHAPE		10578	6399
Tate and Smith		8861	5080.1
FactDesign	19812.1**	8350**	4919**

Table 4.2: AB20 Results

Author	**α_{max}**							
	5	**4**	**3**	**2**	**1.75**	**1.707**	**1.5**	**1.4**
Tate & Smith (FLEXBAY) 1995	5524.7	5743	5832.6	6171.1	7205	6662.9	-	-
Kim & Kim (SPM) 1998	-	5465	5646	**5818.1**	5872.7	6144.9	6542	6824
FactDesign	**5261**	**5261**	**5571**	5835	**5835**	**5835**	**6490.4**	**6743.4**

CHAPTER FIVE: CONCLUSION AND RECOMMENDATIONS

5.1 Conclusion

In this thesis, two forms of the facility layout problem were explored: the equal area dynamic facility layout problem and the unequal area facility layout problem. The EA-DFLP is concerned with planning a multi-period facility layout that minimizes the sum of both the material handling and rearrangement cost. For this problem, the assumption that all the departments of a facility have equal sizes holds. A genetic algorithm was developed for this problem. This algorithm is a direct implementation of the genetic algorithm with the incorporation of elitism as a replacement criteria. Some sets of well-known problems were tested with this algorithm. The algorithm generated a new best solution for one of the problems and it obtained the best-found results for most of the rest.

This research also explored a more practical assumption of unequal department areas. More often than not, this is the case in real world systems. A new modeling approach called progressive modeling was presented and applied to the unequal areas layout problem. A software framework was developed to capture the data, implement the proposed solution algorithm, and display the results. A component model that demonstrates the problem definition, modeling, and solution was presented.

A novel solution representation made up of object-oriented binary tree and a double linked network graph was used in this framework. The new solution representation, unlike in other meta-heuristics, eliminated the encoding and decoding operations and accelerated the solution evaluation using the local updating. The developed model and framework can both be easily maintained and updated. Some sets of well-known problems were also tested

with this solution approach. The new modeling approach is significantly faster and more promising in solving complex real life problems.

5.2 Recommendations

The progressive modeling approach, although new, is showing a lot of promise in solving large sized facility layout problems. This could be ground-breaking, especially with the increasing need for smart and informed algorithms to handle complex real world problems. Areas of future research include:

1. A model to include the dynamic layout and stochastic versions of the unequal areas facility layout problem. This should include development of a model that captures the probabilistic nature of the material flow in facilities.
2. Extension of the facility layout problem to include multi-objective models under both the equal and unequal area assumptions.
3. A pilot study on an actual manufacturing or service system to show the cost savings that could be generated from effective facility using the progressive modeling approach. This would also show the effectiveness of the solution approach in handling large sized problems.
4. A model to include variable department rearrangement cost. The time value of money should also be factored into both the material handling and rearrangement costs.
5. The unit cost of material flow between departments should be a function of the type, weight, shape or size of material flowing.

REFERENCES

Armour, G. C., Buffa, E. S., 1963. A heuristic algorithm and simulation approach to relative location of facilities, Management Science 9(2), 294-309.

Balakrishnan, J., Cheng, C. H., 2009. The dynamic plant layout problem: Incorporating rolling horizons and forecast uncertainty, Omega 37(1), 165-177.

Balakrishnan, J., Cheng, C. H., 2000. Genetic search and the dynamic layout problem, Computers & Operations Research 27(6), 587-593.

Balakrishnan, J., Cheng, C. H., 1998. Dynamic layout algorithms: a state-of-the-art survey, Omega 26(4), 507-521.

Balakrishnan, J., Cheng, C. H., Conway, D. G., 2000. An improved pair-wise exchange heuristic for the dynamic plant layout problem, International Journal of Production Research 38(13), 3067-3077.

Balakrishnan, J., Cheng, C. H., Conway, D. G., Lau, C. M., 2003. A hybrid genetic algorithm for the dynamic plant layout problem, International Journal of Production Economics 86(2), 107-120.

Balakrishnan, J., Jacobs, F. R., Venkataramanan, M. A., 1992. Solutions for the constrained dynamic facility layout problem, European Journal of Operational Research 57(2), 280-286.

Ballou, R. H., 1968. Dynamic warehouse location analysis, Journal of Marketing Research , 271-276.

Baykasoglu, A., Dereli, T., Sabuncu, I., 2006. An ant colony algorithm for solving budget constrained and unconstrained dynamic facility layout problems, Omega 34(4), 385-396.

Baykasoğlu, A., Gindy, N. N., 2001. A simulated annealing algorithm for dynamic layout problem, Computers & Operations Research 28(14), 1403-1426.

Bozorgi, N., Abedzadeh, M., Zeinali, M., 2015. Tabu search heuristic for efficiency of dynamic facility layout problem, The International Journal of Advanced Manufacturing Technology 77(1-4), 689-703.

Burkard, R. E., Bönniger, T., 1983. A heuristic for quadratic boolean programs with applications to quadratic assignment problems, European Journal of Operational Research 13(4), 374-386.

Chen, G. Y., 2013. A new data structure of solution representation in hybrid ant colony optimization for large dynamic facility layout problems, International Journal of Production Economics 142(2), 362-371.

Conway, D. G., Venkataramanan, M., 1994. Genetic search and the dynamic facility layout problem, Computers & Operations Research 21(8), 955-960.

Deb, K., Pratap, A., Agarwal, S., Meyarivan, T., 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE transactions on evolutionary computation 6(2), 182-197.

Drira, A., Pierreval, H., Hajri-Gabouj, S., 2007. Facility layout problems: A survey, Annual Reviews in Control 31(2), 255-267.

Dunker, T., Radons, G., Westkämper, E., 2005. Combining evolutionary computation and dynamic programming for solving a dynamic facility layout problem, European Journal of Operational Research 165(1), 55-69.

Erel, E., Ghosh, J., Simon, J., 2003. New heuristic for the dynamic layout problem, Journal of the Operational Research Society 54(12), 1275-1282.

Fruggiero, F., Lambiase, A., Negri, F., 2006. Design and optimization of a facility layout problem in virtual environment, 13-16.

Gambardella, L. M., Taillard, É D., Dorigo, M., 1999. Ant colonies for the quadratic assignment problem, Journal of the operational research society 50(2), 167-176.

Gomory, R. E., Hu, T. C., 1961. Multi-terminal network flows, Journal of the Society for Industrial and Applied Mathematics 9(4), 551-570.

Ismail, M., 2013. Progressive Modeling: The Process, the Principles, and the Applications, Procedia Computer Science 16, 39-48.

Ismail, M., 2011. Change-ready mpc systems and progressive modeling: vision, principles, and applications,.

Ismail, M. A., ElMaraghy, H., 2009. Progressive modeling—An enabler of dynamic changes in production planning, CIRP Annals-Manufacturing Technology 58(1), 407-412.

Kaku, B. K., Mazzola, J. B., 1997. A tabu-search heuristic for the dynamic plant layout problem, INFORMS Journal on Computing 9(4), 374-384.

Konak, A., Kulturel-Konak, S., Norman, B. A., Smith, A. E., 2006. A new mixed integer programming formulation for facility layout design using flexible bays, Operations Research Letters 34(6), 660-672.

Koopmans, T. C., Beckmann, M., 1957. Assignment problems and the location of economic activities, Econometrica: journal of the Econometric Society , 53-76.

Kuppusamy, S., 2001. Simulated annealing heuristics for the dynamic facility layout problem .

Lacksonen, T., Enscore Jr, E. E., 1993. Quadratic assignment algorithms for the dynamic layout problem, The International Journal of Production Research 31(3), 503-517.

Lacksonen, T. A., 1994. Static and dynamic layout problems with varying areas, Journal of the Operational Research Society 45(1), 59-69.

McKendall Jr, A. R., Liu, W., 2012. New Tabu search heuristics for the dynamic facility layout problem, International Journal of Production Research 50(3), 867-878.

McKendall, A. R., Shang, J., 2006. Hybrid ant systems for the dynamic facility layout problem, Computers & Operations Research 33(3), 790-803.

McKendall, A. R., Shang, J., Kuppusamy, S., 2006. Simulated annealing heuristics for the dynamic facility layout problem, Computers & Operations Research 33(8), 2431-2444.

Meller, R. D., Chen, W., Sherali, H. D., 2007. Applying the sequence-pair representation to optimal facility layout designs, Operations Research Letters 35(5), 651-659.

Meller, R. D., Narayanan, V., Vance, P. H., 1998. Optimal facility layout design, Operations Research Letters 23(3), 117-127.

Mihajlović, I., Živković, Đ, Štrbac, N., 2007. Using genetic algorithms to resolve facility layout problem, Serbian Journal of Management 2(1), 35-46.

Montreuil, B., 1991. A modelling framework for integrating layout design and flow network design, in: Anonymous Material Handling'90, Springer, pp. 95-115.

Pardalos, P. M., Crouse, J. V., 1989. A parallel algorithm for the quadratic assignment problem, 351-360.

Pourvaziri, H., Naderi, B., 2014. A hybrid multi-population genetic algorithm for the dynamic facility layout problem, Applied Soft Computing 24, 457-469.

Rosenblatt, M. J., 1986. The dynamics of plant layout, Management Science 32(1), 76-86.

Scholz, D., Petrick, A., Domschke, W., 2009. STaTS: a slicing tree and tabu search based heuristic for the unequal area facility layout problem, European Journal of Operational Research 197(1), 166-178.

Shayan*, E., Chittilappilly, A., 2004. Genetic algorithm for facilities layout problems based on slicing tree structure, International Journal of Production Research 42(19), 4055-4067.

Sherali, H. D., Fraticelli, B. M., Meller, R. D., 2003. Enhanced model formulations for optimal facility layout, Operations research 51(4), 629-644.

Singh, S. P., Sharma, R. R., 2006. A review of different approaches to the facility layout problems, The International Journal of Advanced Manufacturing Technology 30(5-6), 425-433.

Tam, K. Y., 1992. Genetic algorithms, function optimization, and facility layout design, European Journal of Operational Research 63(2), 322-346.

Tate∗, D. M., Smith, A. E., 1995. Unequal-area facility layout by genetic search, IIE transactions 27(4), 465-472.

Tompkins, J. A., White, J. A., Bozer, Y. A., Tanchoco, J. M. A., 2010. Facilities Planning. : John Wiley & Sons.

Urban, T. L., 1993. A heuristic for the dynamic facility layout problem, IIE transactions 25(4), 57-63.

Van Camp, D. J., Carter, M. W., Vannelli, A., 1992. A nonlinear optimization approach for solving facility layout problems, European Journal of Operational Research 57(2), 174-189.

Wong, K. Y., 2010a. Applying ant system for solving unequal area facility layout problems, European Journal of Operational Research 202(3), 730-746.

Wong, K. Y., 2010b. Solving facility layout problems using Flexible Bay Structure representation and Ant System algorithm, Expert Systems with Applications 37(7), 5523-5527.

Yu, X., Gen, M., 2010. Introduction to Evolutionary Algorithms. : Springer Science & Business Media.

APPENDICES

APPENDIX A

The following shows the problem data of a 5-period and 6-department problem (2×3 facility layout problem) of (Rosenblatt, 1986). The problem data includes the material flow data, the shifting costs and the distance data.

Figure A.1: Material flow matrix for Rosenblatt 6-department problem

Period 1	0	63	605	551	116	136
	63	0	635	941	50	191
	104	71	0	569	136	55
	65	193	622	0	77	90
	162	174	607	591	0	179
	156	13	667	611	175	0
Period 2	0	175	804	904	56	176
	63	0	743	936	45	177
	168	85	0	918	138	134
	51	94	962	0	173	39
	97	104	730	634	0	144
	95	115	983	597	24	0
Period 3	0	90	77	553	769	139
	168	0	114	653	525	185
	32	35	0	664	898	87
	27	166	42	0	960	179
	185	56	44	926	0	104
	72	128	173	634	687	0
Period 4	0	112	15	199	665	649
	153	0	116	173	912	671
	10	28	0	182	855	542
	29	69	15	0	552	751
	198	71	42	24	0	758
	62	109	170	90	973	0
Period 5	0	663	23	128	119	50
	820	0	5	98	141	66
	822	650	0	137	78	91
	826	570	149	0	93	151

	915	515	53	35	0	177
	614	729	178	10	99	0
Shifting Costs	887	964	213	367	289	477

Distance Matrix (2×3 facility layout)

0	1	2	1	2	3
1	0	1	2	1	2
2	1	0	3	2	1
1	2	3	0	1	2
2	1	2	1	0	1
3	2	1	2	1	0

APPENDIX B

The following data are for problem II (section 3.3.2). It is a 9-department and 5-period problem with a 3×3 facility layout taken from Conway and Venkataramanan (1994).

Figure B.1: Material flow matrix for Conway and Venkataramanan problem

Period 1

0	3622	258	493	697	296	627	552	287
991	0	316	443	570	684	334	283	1043
673	6522	0	484	114	324	611	762	762
791	4369	203	0	170	1031	598	923	788
867	5146	56	203	0	1121	309	154	361
894	3264	71	62	769	0	664	343	282
714	3113	240	506	831	1183	0	1144	311
588	1319	319	161	826	1194	744	0	773
1096	6521	335	317	459	439	416	1222	0

Period 2

0	136	6371	886	1596	213	499	1378	476
657	0	3461	1275	567	254	405	263	449
590	528	0	488	498	273	311	1277	486
179	684	1305	0	1748	101	462	1008	559
772	550	6113	478	0	261	53	1134	1285
511	822	2046	1105	1404	0	384	405	875
577	690	2362	925	944	139	0	847	312
300	461	3343	514	676	128	487	0	214
291	560	6306	397	235	243	466	963	0

Period 3

0	265	720	3275	361	230	580	221	1433
695	0	816	5276	636	683	637	1877	203
901	1535	0	2322	323	592	129	857	979
1138	298	987	0	400	1051	163	238	924
619	478	856	4205	0	615	81	991	990
647	1373	441	722	608	0	128	603	1040
1008	1383	772	3552	497	836	0	1795	211
1348	682	233	892	206	600	448	0	679

1291	2281	595	3972	89	840	257	348	0

Period 4

0	753	632	1686	722	241	192	510	63
840	0	897	795	3331	1274	426	611	442
2138	895	0	1277	3019	693	88	470	514
561	445	1444	0	1123	385	523	2015	428
335	421	1549	560	0	820	251	1480	455
636	515	776	1590	5257	0	781	504	416
571	625	765	1304	5312	954	0	647	82
1675	297	176	1137	1240	1313	715	0	321
1187	1550	751	441	840	336	252	1695	0

Period 5

0	1017	663	1460	1118	804	256	1291	246
854	0	1102	1476	1109	2931	975	1032	403
850	1017	0	1503	412	4102	613	1083	140
525	205	792	0	1060	3647	196	591	981
1653	113	1133	1501	0	2160	203	706	695
981	686	184	852	450	0	155	560	962
781	1010	353	319	648	2043	0	914	185
2031	701	930	755	1113	1883	772	0	175
867	580	377	478	284	4879	106	325	0

Shifting Costs

802	985	517	500	736	910	768	564	923

Distance Matrix (3×3 Facility layout)

0	1	2	1	2	3	2	3	4
1	0	1	2	1	2	3	2	3
2	1	0	3	2	1	4	3	2
1	2	3	0	1	2	1	2	3
2	1	2	1	0	1	2	1	2
3	2	1	2	1	0	3	2	1
2	3	4	1	2	3	0	1	2
3	2	3	2	1	2	1	0	1
4	3	2	3	2	1	2	1	0

APPENDIX C

The data for problem number 1 and 5 taken from Lacksonen and Enscore Jr (1993) are given below:

Figure C.1: Problem Number 1 from Lacksonen and Enscore Jr (1993).

Period 1

0	0	0	0	0	0
8	0	0	0	0	0
4	2	0	0	0	0
8	1	4	0	0	0
6	0	8	1	0	0
7	3	4	7	5	0

Period 2

0	0	0	0	0	0
7	0	0	0	0	0
8	1	0	0	0	0
7	1	1	0	0	0
6	0	7	1	0	0
7	5	1	3	6	0

Period 3

0	0	0	0	0	0
10	0	0	0	0	0
10	1	0	0	0	0
8	1	1	0	0	0
5	0	4	1	0	0
10	1	1	6	5	0

Shifting Cost = 50, for all departments.

Distance Matrix

0	1	2	1	2	3
1	0	1	2	1	2
2	1	0	3	2	1
1	2	3	0	1	2
2	1	2	1	0	1
3	2	1	2	1	0

Problem Number 5 from Lacksonen and Enscore Jr (1993). This is a 6-department and 5-period problem. N = 6, T = 5.

	Period 1				
0	0	0	0	0	0
2	0	0	0	0	0
8	1	0	0	0	0
4	6	0	0	0	0
8	1	7	3	0	0
4	7	5	10	5	0
	Period 2				
0	0	0	0	0	0
1	0	0	0	0	0
8	5	0	0	0	0
4	0	0	0	0	0
6	6	9	5	0	0
4	4	4	9	3	0
	Period 3				
0	0	0	0	0	0
1	0	0	0	0	0
10	3	0	0	0	0
3	0	0	0	0	0
0	3	9	6	0	0
2	2	2	9	1	0
	Period 4				
0	0	0	0	0	0
1	0	0	0	0	0
10	3	0	0	0	0
6	7	0	0	0	0
0	2	7	0	0	0
2	1	2	0	1	0
	Period 5				
0	0	0	0	0	0

0	0	0	0	0	0
10	10	0	0	0	0
7	1	0	0	0	0
0	0	6	0	0	0
3	10	4	0	3	0

Shifting Cost = 10, for all departments

Distance Matrix

0	1	2	1	2	3
1	0	1	2	1	2
2	1	0	3	2	1
1	2	3	0	1	2
2	1	2	1	0	1
3	2	1	2	1	0