MODELING AND ANALYSIS OF THE FACILITY LAYOUT PROBLEM

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

ZEYNEP KİRKİZOĞLU

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
INDUSTRIAL ENGINEERING

JULY 2006

Approval of the Graduate School of Natural and Applied Sciences

_____

Prof. Canan Özgen
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

_____

Prof. Çağlar Güven
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

_____

Assoc. Prof. Haldun Süral
Supervisor

Examining Committee Members

Assoc. Prof. Canan Sepil	(METU, IE)	_____
Assoc. Prof. Haldun Süral	(METU, IE)	_____
Asst. Prof. Osman Alp	(Bilkent Uni., IE)	_____
Asst. Prof. Pelin Bayındır	(METU, IE)	_____
Asst. Prof. Ayten Türkcan	(METU, IE)	_____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name : Zeynep KİRKİZOĞLU

Signature :

ABSTRACT

MODELING AND ANALYSIS OF THE FACILITY LAYOUT PROBLEM

KİRKİZOĞLU, Zeynep

M. Sc. Thesis, Department of Industrial Engineering
Supervisor: Assoc. Prof. Haldun Süral

July 2006, 162 pages

The facilities layout problem, which is an integral part of facilities design, aims to spatially locate the production units within a facility subject to some design criteria and area limitations, with one or multiple objectives. In this study, the layout problem is reviewed in detail, with an emphasis on the dynamic environment it operates in. Despite the fact that layouts within the context of changing manufacturing requirements represent the problem better, the single period block layout problem is observed to have remained worth analyzing.

In this thesis, a hybrid model that combines the strong aspects of the available models in the literature is constructed for the single period block layout problem. The LP relaxation of this model and the effect of adding valid inequalities to the model are studied. A rounding heuristic based on the LP relaxation of the problem is proposed and computational experimentation is made. Also, an evolutionary algorithm scheme that uses the sequence pair representation is proposed. Three mutation operators are developed to be used in this scheme. Preliminary test are made for implementations of these operators and results are given.

Keywords: Facilities Layout, Heuristics, Evolutionary Algorithms

ÖZ

TESİS YERLEŞİM PROBLEMİNİN MODELLENMESİ VE ANALİZİ

KİRKİZOĞLU, Zeynep

Yüksek Lisans Tezi, Endüstri Mühendisliği Bölümü
Tez Yöneticisi: Doç. Dr. Haldun SÜRAL

Temmuz 2006, 162 sayfa

Tesis tasarım probleminin oldukça önemli bir parçası olan tesis yerleşim problemi, bir tesis içindeki üretim birimlerini bazı tasarım ve alan kriterleri altında, bir ya da birçok amaç doğrultusunda uzaysal bir düzleme yerleştirmeyi hedefler. Bu çalışmada, yerleşim problemleri, içinde faaliyet gösterdikleri çevrenin dinamikliği vurgulanacak şekilde incelenmiştir. Her ne kadar tesis yerleşim problemleri değişen üretim ihtiyaçları kapsamında ele alındıklarında problemi daha iyi temsil ediyor olsalar da, tek dönemlik yerleştirme taslağı probleminin incelenmeye değer bir problem olarak kaldığı görülmüştür.

Bu tezde, tek dönemlik yerleştirme taslağı problemi için yazında mevcut olan modellerin güçlü yönlerini birleştiren melez bir model kurulmuştur. Bu modelin doğrusal programlama gevşetmesi ve modele geçerli eşitsizlikler eklenmesinin etkileri çalışılmıştır. Problem için doğrusal programlama gevşetmesine dayanan bir yuvarlama sezgiseli geliştirilmiş ve sayısal deneylerle denenmiştir. Ayrıca, sıralama çifti gösterimini kullanan bir evrimsel algoritma tasarısı önerilmiştir. Bu tasarıda kullanılmak üzere üç mutasyon işleci geliştirilmiştir. Bu işleçlerin uygulamaları için ön testler yapılmış ve sonuçlar verilmiştir.

Anahtar Kelimeler: Tesis Yerleşimi, Sezgiseller, Evrimsel Algoritmalar

To Mom, Dad and Şılı

ACKNOWLEDGMENTS

First of all, I would like to express my gratitude to my supervisor, Assoc. Prof. Haldun Süral. During this study, I have learnt much from him, both in terms of research and life.

I also would like to thank all members of my examining committee for the very constructive comments on the organization and structure of the thesis. Especially, the contributions of Pelin Bayındır and Ayten Türkcan during the finalization of the manuscript are acknowledged.

I can neither put the endless love and support of my family into words, nor pay my debt to them by any means. All I am able to do is thank them.

I am greatly indebted to Sakine Batun and Melih Özlen for their invaluable friendship. They have taught me that people need not be born from the same parents in order to be brothers and sisters.

This thesis would never be complete without the inspiration, encouragement and friendliness of the department. I would like to thank all the professors, assistants, students and administrative staff of the department for this wonderful environment.

This study was supported through a graduate study scholarship by TÜBİTAK.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

Layout problem aims to spatially locate interrelated units such as departments, machines, chips, etc., subject to some design criteria and area limitations, with one or multiple objectives.

More specifically, the facilities layout problem, which is an integral part of facilities design, aims to locate the production units within a facility. Since facilities design is a very complex issue, which affects and is affected by almost all production related decisions, it is usually handled hierarchically rather than by an integrated fashion. Some examples of decisions related with facilities design are the product mix, the type and quantity of machinery, tooling and material handling equipment to use in production, the routes products will follow, method of planning and scheduling of jobs, control of raw material, work-in-process and finished good inventories, control of incoming and outgoing material to and from the facility, etc. The layout problem is one of the core subproblems of facilities design, related with almost all of the other decisions exemplified above, therefore having its share from the complexity of facilities design.

Despite the fact that it does not reflect the complexity of the problem, most of the studies on the layout problem focus on solving a “snapshot” of the manufacturing environment by minimizing the annual material handling cost. This is to achieve “solvability” by handling issues sequentially. Models constructed with an objective function of minimizing the annual material handling cost usually assume that there is dedicated material handling equipment between all department pairs. Additional assumptions are considerably stable flow amounts between departments for the time period planning is made for, no bottleneck in material handling resources, and setup

cost of material handling equipment being either negligible with respect to the total material handling cost or independent from the layout type.

The context in which a layout decision should be made is also an important aspect of the problem. Here, a distinction between the layout of machines and the layout of departments needs to be made. When the layout of machines is to be considered, the size of the machines are given and a more detailed layout is usually aimed, such as leaving a clearance between each machine pair, determining aisles and loading/unloading points, etc. For the case of laying out departments, most department shapes are not given as rigid or hard measures. Rather, the area each department requires is given and reasonable shapes are achieved by imposing an aspect ratio (α) – the ratio of a department's length over its width, or vice versa (whichever is maximum). It should be noted that although there is also research on layout with irregular shaped departments, most of the studied representations assume rectangular shaped departments. An exception for this assumption is seen in the method where the facility is divided into equal sized grids and departments are constructed by assigning them to several adjacent grids in the facility. In this method, a specific department shape is not aimed and the final layout can have very abnormal shaped departments.

Due to the fact that the models discussed above do not effectively represent the actual problem, they have been criticized and alternative modeling frameworks have been considered with increasing interest in recent years. Actually, using different types of models is a reflection of the slowly occurring paradigm shift regarding the perception of the layout problem. The steps towards this shift can be said to be formed by two factors: 1. Attempts to make the problem capture the practical issues more realistically; 2. Need for a different layout concept in order to cope with the new requirements that are caused by changes in manufacturing systems. Two examples for the first factor during preliminary stages of the layout research is the focus shift from modeling equal-area problems (using the quadratic assignment problem) to modeling unequal-area problems, and the consideration of alternative

scenarios for demand, which are used to analyze the robustness or flexibility of the layout.

The second factor is a recent issue, formally brought up by Benjaafar et al. (2002), although its signs were being observed in the layout literature for a while. It is discussed that product mixes and volumes are subject to frequent change, uncertainty in the parameters used for design is usually high, and "scope is more important than scale, responsiveness is more important than cost, and reconfigurability is more important than efficiency".

Some examples of active research topics associated with the above discussed two factors that lead us towards a new point of view to the layout problem are as follows:

- The integrated design of material handling system and layout (determination of aisle structures and input/output (I/O) points can also be included)
- The impact of layout on the total WIP inventory, congestion, cycle times and throughput rates and in the facility (see Benjaafar et al., 2002)
- The tradeoff between material handling and relayout costs, considering the level of uncertainty associated with each
- The level of robustness / flexibility / agility / reconfigurability of the layout with respect to known, partially known or unknown changes in product mixes, volumes, or routes
- The level of ease the layout can be reconfigured (physically or virtually)
- Utilization levels of the departments, machinery or material handling equipment

This study provides a detailed review of the layout problem in dynamic environments, which is given in Chapter 2, along with findings of some studies that can lead us to understanding the types, severity and frequency of the changes in the environment of a layout. Although the studies that consider the problem within a dynamic environment are better in terms of capturing the complexities associated with the problem, conducting these types of analyses requires being able to find

good solutions to the “classical” block (department) layout problem in short times. However, it is seen that there is still much to do with respect to the solution of this “simple” problem, where optimality can be achieved for up to only 11 department problem instances and heuristic procedures are designed for special layout types. Therefore, the focus of our study has been directed to develop a solution methodology for the conceptually simple, but the computationally hard single period block layout problem. We deal with rectangular shaped departments, where the distances between departments are measured from center to center and rectilinear distance metric is used. Departments, whose length and width values are decision variables that should satisfy a given limit of aspect ratio, are to be placed within a facility of rigid width and length. The objective function is to minimize the total material handling cost over a sufficiently long single period, where the flow costs and amounts between departments do not change.

In this thesis, a mixed integer programming model for the single period block layout problem is proposed, which brings the strong aspects of two alternative formulations in the literature together. LP relaxation of this model is analyzed and a heuristic based on rounding the value of solutions is developed. Performance of the heuristic is tested under several parameter settings.

In addition, an evolutionary algorithm based on a recently introduced representation scheme, the sequence pair representation, that does not impose certain specifications on the layout is proposed. Three mutation operators, based on two basic heuristic concepts are developed and preliminary experimentation is conducted to assess the qualities of the operators.

The contributions of this study can be summarized as follows:

- The layout problem is investigated within a broad context and the research in this area is presented in integrity.
- Two strong formulations with respect to different aspects, namely, the accuracy of the department area linearization and the tightness of the mixed integer programming (MIP) formulation are combined for the first time.

- The LP relaxation of the layout problem is investigated to develop insights about the problem. This study is the first in the literature to analyze the LP relaxation and develop an LP-based rounding heuristic for the single period block layout problem.
- The sequence pair representation is analyzed and two well-known basic local search moves are adapted to this representation in a more "intelligent" way, which are proposed as mutation operators for a possible evolutionary algorithm.

The content of the thesis is as follows. Related literature that focuses on the dynamic nature of the layout problem and the reasons that have led the problem to be considered in this context are given in detail and discussed in Chapter 2. Chapter 3 complements the discussion of the layout literature given in Chapter 2 by adding the single period layout problem into the picture. In Chapter 3, the alternative solution methods proposed for the single period layout problem are given, with a focus on the mathematical modeling aspects of the problem. The properties of the LP relaxation of the model, along with the proposed rounding heuristic complete the chapter. Chapter 4 is devoted to the evolutionary algorithm applications on the single period layout problem. After a brief review of the literature on this subject, an evolutionary algorithm scheme based on the sequence pair representation is proposed and mutation operators developed for this algorithm are introduced, together with the preliminary results obtained from a simple implementation of these operators. Finally, conclusive remarks are provided and future research directions are discussed in Chapter 5.

CHAPTER 2

A CRITICAL REVIEW OF THE LAYOUT LITERATURE

This chapter focuses on the layout problem with a new perception of the problem in layout community, which can partially be attributed to the impact of changing manufacturing requirements on the problem. The analysis on the single period layout problem is given in Chapter 3.

Today, the layout problem is considered in a dynamic nature. There may be internal or external reasons why a layout would aspire to change. External reasons are effects from the environment the layout operates. These reasons are either hard or impossible to control, for which the layout must act either proactively or reactively. For example, a sudden change in the raw material procurement due to some unexpected government regulation on imported materials is most probably to result in a temporary change in product mix that the layout will be required to produce, although is not specifically designed to do so.

Internal reasons, on the other hand, are due to design inefficiencies and have nothing to do with the external reasons. For example, if the material handling equipment is not suitable for turns, using a U-shaped or an L-shaped department is a design inefficiency, which is independent of external factors. Here, the layout is simply bad with respect to what it could have been and these factors can be intervened at any time to improve the efficiency.

Dynamic nature of the problem can be attributed to the existence of external factors. Therefore, the focus in this analysis is on external factors, their consequences and alternative ways to respond to them.

2.1 Review of Studies Assuming Dynamic Environments

Hitchings was the first to recognize and report on the dynamic nature of the layout design problem in 1970 (see Lacksonen, 1997). This dynamic nature adds the rearrangement costs to the usual "average material handling cost" in the classical objective function and introduces a tradeoff between the increasing material handling costs and the costs of relayout.

Time aspect is the most distinguishable property of the layout problem in dynamic environments. The planning horizon is usually divided into discrete time intervals of meaningful length and equal size, called "period"s. Lacksonen and Enscore (1993) discuss that the periods can be related to months, years or phases of a project; where most businesses plan for 3-5 periods.

Benjaafar et al. (2002) is the first to provide a structured analysis of this changing structure and its current and future reflections on the way the layout problem is handled. Some emerging trends in the industry that lead the researchers to this completely different perception of the problem are identified as contract manufacturing, delayed product differentiation, multichannel manufacturing, scalable machines and portable machines. The authors name the layouts that can cope with these changes as "next generation factory layouts" and discuss three of them, namely, distributed, modular and agile layouts, in more detail.

Benjaafar et al. (2002) also provide a nice classification of the layout problem within this framework of "everything changing". This classification, which is based on the production environment characteristics, is shown in Table 2.1. We also found it convenient to give the details of the studies we reviewed based on this classification.

Table 2.1 – Classification of the layout problem by Benjaafar et al. (2002)

	Uncertainty of future production requirements	
Cost of relayout	*Low*	*High*
Low	Dynamic layout	Reconfigurable layout
High	Robust layout	Distributed layout

2.1.1 Layout Planning in Environments of Low Uncertainty

Uncertainty of future production requirements being low implies a deterministic look to the time ahead and requires knowledge of several incoming periods so as to make plans accordingly. This knowledge of demand data can either be deterministic or stochastic. For deterministic case, there is a single flow matrix for each period. For stochastic case, alternative scenarios for product demands (implicitly, the flow data between departments) are given, although the exact probability of occurrence for each scenario may or may not be known. Stochastic case is also referred to as "uncertainty in demand", which is not used here to avoid confusion of terms with the titles under analysis.

If relayouts are possible, i.e., cost of relayout is low, then the type of layout in question is dynamic layouts. In the dynamic layout problems, the aim is to determine the points in time that the layout will change and which layouts to be used for each time without change. A review of dynamic layout problems (DLP) with deterministic flow data was conducted on a detailed basis with the expectation of findings that similar to the scheme of the next generation facility layout problems introduced in Benjaafar et al. (2002). However, it turned out that the literature on DLP is somewhat like a closed loop that only focuses on the defined problem and tries to develop computational results for this problem. Therefore, the detailed review is given in Appendix A for the interested reader.

Main criticism on these layout types is the assumption of having reliable flow data for several periods ahead. In many environments, these types of layouts would not provide the desired level of performance due to the diminishing reliability of data through distant futures. For instance, it could not be possible to name a group of products such that their future demand is known precisely. Another issue here is the underlying assumption of knowing all possible improvements in technology for the same future (so as to determine when and which departments will be removed and when and which will be added together with all associated technology that is being used and to be used). In addition, relayout costs might show nonlinear behavior, which yield extra computational difficulties, or may not be possible to estimate at all.

A very nice example of the DLP is observed in the layout of temporary facilities during a construction job. Throughout a construction project, several “departments” such as warehouses, job offices, workshops, equipment (like cranes), etc. need to be laid out and the requirements change as the project progresses. Mawdesley et al. (2002) nicely summarize the requirements, which are:

1. Access and traffic routes
2. Material storage and handling
3. Administration buildings and welfare facilities
4. Equipment, workshops and services

The problem of allocating space to resources governed by a construction schedule, and conversely, changing the schedule, i.e., changing activity sequences, resource selection and allocation or activity durations, when space availability is inadequate, is termed space scheduling (bidirectional interaction between scheduling and layout). Here, the project schedule and the layout are determined simultaneously.

Dynamic layout planning refers to the problem of constructing a sequence of layouts to span the duration of construction of a project subject to the activity schedule (unidirectional interaction from scheduling to layout). Here, the space scheduling problem is handled sequentially. After determining the schedule, the layout is designed subject to it.

The second scheme perfectly matches the deterministic structure of the DLP formulation, which is a well known and studied problem in construction engineering, mostly independent from the research that has been conducted within the IE area. Some examples of such studies are Tommelein and Zouein (1993), Zouein and Tommelein (1999), Ebeltagi et al. (2004) (with an emphasis on the safety issue), Chau (2004) and Ma et al. (2005). These papers focus on the applicability of the model, developing user interfaces and presenting case studies. Because of their focus on real life use, the requirements they are designed to handle can be put forward as a proof for the need to relayout (temporary) facilities during construction.

If relayouts are not possible (i.e. cost of relayout is high), the aim is to construct a layout that can be used forever without substantial physical changes. Therefore, the layout should perform "good" under all circumstances, although probably not "best" even for a single circumstance. These layouts are termed as "robust" or "flexible" layouts. Although their approaches are different, concepts of robustness and flexibility are close to each other in terms of their ability to respond to changes. Their common usage within the context is "robustness to uncertainty" and "flexibility for future changes" (Kulturel-Konak et al., 2004). More detailed definitions from the literature are as follows:

- "Robustness of a decision is characterized by its ability (flexibility) to handle changes in the environment." (see Rosenblatt and Lee, 1987)
- "A robust facility design means that it performs well over a variety of scenarios and outcomes. On the other hand, a flexible facility design means that it can readily adapt itself to changes without significantly affecting performance." (Kulturel-Konak et al, 2004)
- "Flexibility is the ability of a layout to respond to known and future product mixes." (Webster and Tyberghein, 1980)
- *Reactive flexibility:* It is the measure of insensitivity of a layout to changing material flow volumes between departments. Here, the random material flow costs are the major elements.

Adaptive flexibility: It is the measure of ability of a layout to be changed in the future. Here, the random relayout costs are the major elements (Savsar, 1991).

Using stochastic flow data for the single period layout problem and developing robust or flexible layouts have received much attention in the layout literature. Examples of such studies are Webster and Tyberghein (1980), Gupta (1986), Rosenblatt and Lee (1987), Savsar (1991), Rosenblatt and Kropp (1992), Enea et al. (2005), Aiello and Enea (2001), and Kulturel-Konak et al. (2004). Some related details are given below.

Webster and Tyberghein (1980) choose the measure of flexibility as the magnitude of material handling cost and use a real life situation to illustrate their approach, specifically, a job-shop environment with six products. They evaluate seven alternative layouts; the initial one plus six alternative layouts, each of which is designed for the production of a different single product. Then, they predict a future product mix and compute the corresponding total material handling and rearrangement costs for each layout alternative. The paper is realistic in the sense of calculation of material handling costs and reasoning why the rearrangement costs for certain alternatives are high. This is due to the fact that the problem is based on a real life example. The paper does not have very strong theoretical results or conclusions, however, it can be regarded as one of the papers that have the first essences of DLP before it was formally defined.

Gupta (1986) utilizes simulation to select a layout from a candidate set. He firstly finds the optimal layouts for a sufficient number of flow matrices generated from the probability distribution for individual flow volumes. He defines the ideal solution as the layout with each department pair being separated by the average distance of the generated layouts. Then the flexibility of the layout being analyzed is measured by evaluating its deviation from these ideal distances.

Rosenblatt and Lee (1987) consider the case where the demand for each product can be estimated at three levels: High, Medium and Low, corresponding to the highest, average and lowest possible values of demand, respectively. They assume that the route of each product is known and consider the objective of minimizing material handling costs. They measure the robustness of an alternative by the number of times that its solution lies within a pre-specified percentage of the optimal solution for different sets of scenarios. In the paper, a 3-product, 4-department example problem with equal areas is demonstrated. This paper gives a clear definition of robustness and compares it with two other approaches: Laplace (e.g. expected cost for each layout, assuming equal likelihood of each scenario) and maximum regret (e.g. difference between the best and worst costs). The authors also discuss the computational complexity issues and suggest some alternatives to overcome these, such as considering only those products highly affecting the material handling costs.

Savsar (1991) uses simulation as an analysis tool. First, he generates a number of random layouts. Then he evaluates them with respect to an objective function that combines material handling cost and closeness rating. Material handling costs are generated from specified distributions and closeness ratings are assumed to be given and not subject to changes. The best layouts with respect to the combined objective are saved and the average distance between each department pair is calculated by taking the average of the distances of these two departments in the saved layouts. Then, a penalty function for the deviation of an alternative layout from this "average" is defined as the 'reactive flexibility measure'. The 'adaptive flexibility measure' is calculated by finding the total expected relayout cost for each layout alternative. Finally, these two flexibility measures are again combined into a single objective function. The author provides a 6-department example with 2 time periods and concludes with parameter optimization concerns. The paper makes a valuable contribution by attempting to consider several objectives at once and incorporating flexibility concerns.

Rosenblatt and Kropp (1992) show that the optimal layout for a given stochastic flow pattern can be obtained by solving a single layout problem by using the expected flow matrix. They do not, however, discuss flexibility issues in detail.

Aiello and Enea (2001) and Enea et al. (2005) represent the uncertainty associated with product demands by fuzzy numbers and solve the single period facility layout problem with the objective function of minimizing total material handling cost.

Kulturel-Konak et al. (2004) consider the case where the demand data for products are stochastic and correlated each other. They also allow for the products to follow different routes in the facility, which they define as “routing flexibility”. Simulation is employed for parameter optimization and Tabu Search is used to solve the problem.

Studies concerning layout problems for multiple periods based on stochastic flow data (usually referred as dynamic layout problems) are less in number, examples of which are Kouvelis et al. (1992), Yang and Peters (1998), and Norman and Smith (2006).

Kouvelis et al. (1992) use the definition of robustness as given by Rosenblatt and Lee (1987). They both consider robustness in a single period with only demand uncertainty and robustness over multiple periods with demand uncertainty and changes in product mix. In handling the single period case, they use a modified branch and bound procedure to find the robust layouts, hence offer a better approach than total enumeration (which is used by Rosenblatt and Lee, 1987). For the multiple period case, they also consider relayout costs. However, they classify the departments (or machinery) into two groups as “monuments” with a very high (almost infinite) cost of relocation and “others” with affordable cost of relocation. The location of monuments are decided and fixed in the first period, and the algorithm is continued by finding robust layouts for each period and picking the one with minimal relocation cost. The authors modify the branch and bound procedure developed for the single period case and use it for finding these solutions. They

demonstrate their solution procedures on two example problems. It is stated that the branch and bound procedure can be used for problems up to 15 locations, but it becomes computationally intractable for larger problems. Therefore, use of heuristics is suggested and it is demonstrated that good results can still be obtained. The paper concentrates only on the QAP and the results do not seem to be valid for all similar type problems, especially as if the percentage of monuments increases. The paper makes a contribution by extending the definition of robustness into multiple periods but the related discussions are not well supported and seem to be superficial.

Norman and Smith (2006) define a robustness index, which is calculated by finding the partial expectation of material handling cost over a user specified interval of probabilities. They then minimize this index using an evolutionary algorithm.

2.1.2 Layout Planning in Environments of High Uncertainty

It is commonly agreed that today's manufacturing environments usually fall in this category. The research on the concepts of manufacturing flexibility and agile manufacturing supports this idea (see De Toni and Tonchia, 1998; Sanchez and Nagi, 2001).

The situation can be analyzed in two sub-categories: the cost of relayout being low and being high. Cost of relayout being low encourages changing the layout very frequently. The single example for this type of layouts, other than Kochhar and Heragu (1999), is given in Meng et al. (2004), who make a significant contribution by supporting the reasons behind the ability of quick reconfigurability and high adaptability. They point out that the layout problem can be turned into a tactical problem from a strategic problem with high levels of reconfigurability. Similar to Kochhar and Heragu (1999), they claim that changes in production data can only be known slightly ahead and the plans can be made for a single changeover. They propose a layout evaluation procedure, considering relocation and material handling costs, WIP inventory levels, and product lead times.

When the cost of relayout is high, the traditional point of view calls for a process (functional) layout, which had been considered for years as the most appropriate type of layout for the environments with high product variety and low production volumes. A new class of layouts is proposed as an alternative to process layouts, where the machines/departments of same type do not need to be adjacent to each other, but rather be dispersed throughout the facility (distributed layout, see Lahmar and Benjaafar, 2005). With this type of layouts, the "cells" or product routings are formed virtually rather than physically, according to the requirements available on hand. Although a functional layout can be argued to have this property, it has been shown that distributing machines/departments within the facility is more effective. Some alternatives proposed for this type of layouts are virtual cellular manufacturing systems (VCMS), fractal layouts that are composed of identical cells that can produce all types of products, and holographic layouts in which machines are distributed over the facility such that same types are away from each other as much as possible (see Benjaafar et al., 2002). Benjaafar and Sheikhzadeh (2000) also suggest a similar layout type, which is designed for multiple periods.

Although the underlying scheduling and routing complexity of these types of layouts should also be considered, these kinds of analyses are not available yet since the research in this area is limited and at the preliminary stages.

2.2 Analysis of Changes that Affect the Layout

Almost all researchers, especially the studies of those analyzed in Section 2.1, state that there are changes in the layout environment, which affect the efficiency of layout. Common reasons stated are changes in product mix (may include introduction of a new product), changes in product design, changes in product volume, changes in raw materials, changes in process and technology (Meyers and Stephens, 2005; Webster and Tyberghein, 1980; Savsar, 1991), shorter product life cycles, higher product variety, increasingly unpredictable demand, shorter delivery times (Benjaafar and Sheikhzadeh, 2000), and so on.

To the best of our knowledge, there does not exist a study that analyzes frequency and/or severity of effects of the above mentioned factors on the layout. Therefore, the answers for the questions "Why and how often a layout has to be changed?" are investigated through sources which might help one to gain insight. The sources utilized for this aim can be grouped into three categories based on their focus issues:

1. Studies concerning efficiency and effectiveness of cellular manufacturing systems throughout time.
2. Studies on manufacturing strategy and its implications on the manufacturing system
3. Research on the qualitative or quantitative features of the commonly cited reasons that result in the need to relayout

Most guiding research in this area is concerned with the life cycles of cellular manufacturing systems. Marsh et al. (1997) analyze the existence of a life cycle for manufacturing cells, which is described by the performance deterioration that leads to a necessity of change in the layout. The authors compile commonly referred reasons of these deteriorations and come up with the cycle circumscribed using dashed lines in Figure 2.1. The two propositions, P_1 and P_2, are described as follows:

P_1 : Over time, the dynamic forces of a manufacturing environment will lead to the deterioration of cell performance.

P_2 : A deterioration in cell performance will necessitate a change in the cell layout.

According to the field study authors have conducted, a total of 17.3% of the cells have experienced a life cycle termination, which were caused by changes in flow, productivity, technology, production volume and product design. The authors obtain little evidence on validity of the propositions P_1 and P_2. However, they find out other mechanisms that companies respond to these changes, named as "coping mechanisms". The complete cycle is shown in Figure 2.1.

Findings of Wemmerlöv and Hyer (1989) by a survey conducted among the companies that employ cellular manufacturing also concur with the above findings. They state that in about half of the companies, 11.3% of the products produced in a

cell (corresponding to 8.1% of the volume) were added to the cell later. This has two general implications: Requirement of additional tooling and fixtures and increase in the utility levels of cells. This finding is in line with the one by Marsh et al. (1997): The companies prefer using the existing cells with small operational modifications rather than costly relayouts, still managing to remain effective. The same situation is also observed by Molleman et al. (2002).

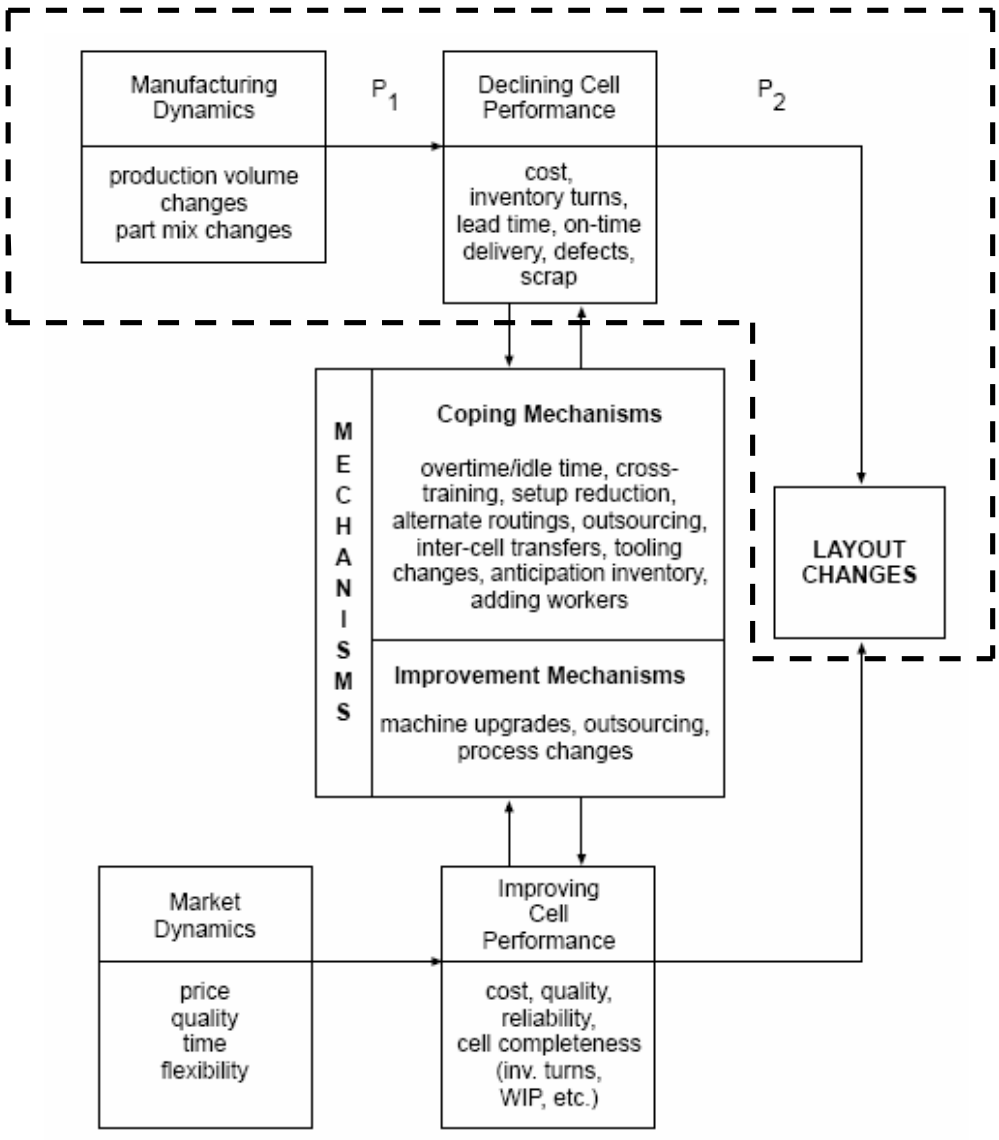


Figure 2.1 – Conceptual map of the cell life cycle given by Marsh et al. (1997)

Molleman et al. (2002) conduct a longitudinal case study and observed three different factors which may lead to a redesign of the cellular manufacturing system:

1. *Changes in strategy:* "The availability of new technology and changes in customer demand changed the focus of management from an internal towards a more external orientation. This change in orientation also stems from an environment that has become more competitive and hostile."
2. *Technological innovations:* "Important characteristics of the new machinery are its level of automation, integration of process capabilities, flexibility and cost. … Flexibility of the new machinery enables quick changeovers."
3. *The market:* "Integration of cells may be needed to avoid the emergence of overly small cells, which are more vulnerable to market changes and therefore less persistent in a dynamic environment."

The second group of studies that can be related with the posed question is on manufacturing strategy changes and flexibility. Although no direct relation has been shown until today, one might infer that changes in manufacturing strategy would have implications on the facility layout.

Cagliano et al. (2005) analyze the patterns of change in manufacturing strategy configurations of the firms. They define four different types of strategies; namely, market-based, product-based, capability-based and price-based. They conclude that firms change their manufacturing strategy relatively easily, although no trend is observed towards a certain type of strategy. So, as the authors state, "This confirms the relevance in emerging competitive paradigms of the ability of the companies to change focus of their manufacturing system."

Two other examples on the manufacturing strategy focused studies are by Ettlie and Penner-Hahn (1994) and Shewchuk and Moodie (2000), who investigate the relationship between manufacturing strategy and manufacturing system flexibility.

The third group of studies aims to analyze whether the commonly cited reasons of change have a basis. Based on empirical observations, Bayus (1994) shows there is

no solid evidence that product life cycles are getting shorter. He attributes this common illusion to the following factors:

1. New knowledge being applied faster
2. A greater number of new products being introduced over time
3. The decrease in time between innovations

Although the paper seems to falsify one of the commonly cited reasons of relayout (shorter product life cycles), it meanwhile supports the idea with the second factor it points out, a greater number of new products being introduced over time. The increasing variability in products is also supported by MacDuffie et al. (1996). Although the paper focuses on assembly facilities of automotive industry, its results can be generalized to others up to a certain extent.

2.3 Conclusions

Independent from whether a researcher restricts him/herself to solve a very restricted version of the layout problem or attempts to solve a number of intertwined problems simultaneously, the link with real life is very important for this problem, especially within the context the layout problem is recently perceived and handled. Therefore, the basis for the problem should clearly be established and supported with real life applications, if possible. The importance of founding bases and demonstrating the reality of the problems become clearer with the fact that manufacturing requirements and technologies are changing fast. A structured basis for the issue with proper classification, consistent terminology and applicability with respect to the available manufacturing practices would be of great value to both industry and academia.

Analysis of the layout problems with “everything is subject to change” point of view showed that this is a more “complete” way to treat the problem. On the other hand, it is seen that this kind of an analysis usually requires solving the single period facility layout problem for several times. Therefore, the prerequisite of handling the problem in a more general framework is being able to obtain good quality results for the

simpler problem in short times. Note that good quality results are required since the errors of the single period solutions will accumulate in dynamic models and short solution times are required to be able to solve the problem for several times. This is the point where the restricted problem (block layout) that hardly reflects the real cost of a layout becomes important. It turned out that this restricted problem is not easy to solve at all and the current methodologies are still not able to come up with a fast and reliable solution methodology, in spite of the extensive research in this area. Note that, the single period facility layout problem is discussed in more detail in Chapter 3.

In short, although the focus of the study was initially set on the dynamic nature of the problem, our findings later directed us to analyze the single period facility layout problem. Devising heuristics for the problem by exploiting its MIP model so as to obtain “good enough” results within reasonable solution times was chosen as the main issue to investigate.

CHAPTER 3

MIXED INTEGER PROGRAMMING MODELS FOR THE SINGLE PERIOD FACILITY LAYOUT PROBLEM

In this chapter, first, a brief overview of the single period facility layout problem is given. Next, the mixed integer programming (MIP) models in the literature that have been developed for the problem are reviewed. Then, a hybrid model is proposed, its details are given and it is tested. Finally, an LP based rounding heuristic is developed for the problem and computational experimentation results are given.

3.1 The Single Period Facility Layout Problem

The single period facility layout problem can simply be defined as the problem of locating departments in a facility. The most recent review of the layout problem can be found in Meller and Gau (1996a). Although mainly focused on the direction towards "next generation facility layouts", Benjaafar et al. (2002) also provide a comprehensive and recent review of the literature, which was discussed in more detail in Chapter 2.

Facilities layout problem is a hard problem to tackle optimally. Even the case where all relative locations of departments with respect to each other are given, department sizes are known, but orientations that are to be determined is shown to be NP-complete in the strong sense by Stockmeyer (1983).

The integer programming models constructed assume rectangular shaped departments with no restriction imposed on the structure in which the departments will be placed with respect to each other. However, they usually put a limit on the aspect ratio of each department. The associated objective function is to minimize the

total material handling costs. Although not used much nowadays, another common objective function was to maximize some closeness rating, based on adjacency scores for each department pair, provided by the decision maker. These integer programming models have a large set of binary variables for the disjunctive constraints (constraints that prevent the departments from overlapping each other), which can be said to define the general structure of the layout. A large set of binary variables makes even the simplest case of the problem hard. The layout problem can also be regarded as a design issue that affects and is affected by several other issues, where “expert opinion” might be needed in order to be able to make some decisions. These factors have led most of the research towards the direction of developing heuristics.

In general, the presence of several assumptions directly forms the definition of the problem itself, especially for the cases in which heuristic approaches are used. It is almost impossible to fairly compare two selected studies on the layout problem due to these different assumptions. Before getting into more detail in this issue, let us first give the list of differences observed in the literature:

1. Differences in the representation of a layout
2. Differences in objective functions
3. Differences in handling constraints
4. Differences regarding inclusion/exclusion of detailed design issues
5. Differences in the data sets used

3.1.1 Differences in the Representation of a Layout

The abstraction of a layout is not as straightforward as network structured problems, e.g., the traveling salesman problem. Because of this, several representation schemes have been developed and used. Here, we give the most used three representations and discuss their effects on the final layout.

Grid representation

This approach divides the available area of the facility into grids of equal size. Then, the departments are allocated to grids ensuring that they are contiguous and occupy the number of grids corresponding to the required area. This approach is actually an extension of the quadratic assignment problem (QAP). At early stages of the layout research, when QAP was the most common method to represent a layout problem, it was presumed that QAP can also handle unequal area layouts by using a grid representation with the inclusion of additional constraints. However, this approach has never been used due to the fact that QAP is a hard problem.

In heuristics, in order to maintain contiguity, the user is required to define a space filling curve (SFC), a line that traverses all grids by moving to the current grid's neighbor each time. The layout can then be represented by placing the departments along the defined SFC. Contiguity is guaranteed since an SFC enters and leaves a department exactly once. An example of a SFC is provided in Figure 3.1.

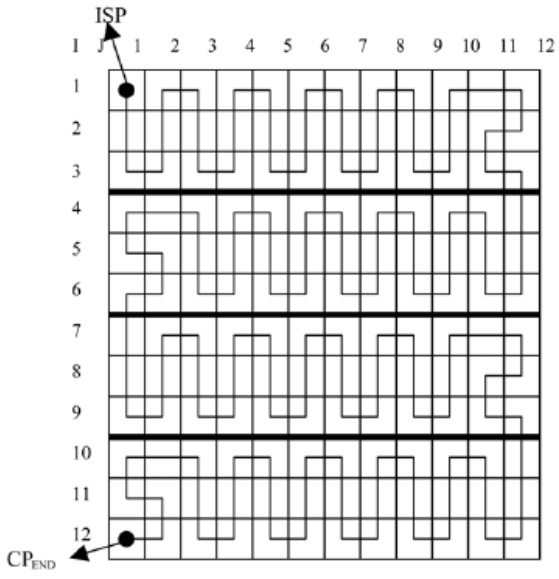


Figure 3.1 – Example of an SFC for facilities divided into grids (Hu and Wang, 2004)

A number of shortcomings for this approach are as follows. First, the final layout is highly dependent on the initial SFC drawn. This can be overcome by using several different SFCs and picking the best result. Next, irregularly shaped departments are produced. Since the generated layouts often need to be repaired, it can be said that this approach usually leads to inferior solutions with respect to other approaches.

Flexible bay structure

This structure was first proposed and used by Tate and Smith (1995). The main idea is to first divide the facility into vertical or horizontal columns, i.e., "bays", of different width, i.e., "flexible", and then locating the departments within these bays by considering the area requirements. An example layout formed by using flexible bay structure is given in Figure 3.2. This structure is usually argued to represent the actual layout problem, where the straight line between two adjacent bays will be used as aisles that material handling moves will take place.

Figure 3.2 – Example of a layout constructed by using flexible bay representation (Chen et al., 2002)

Slicing tree structure

This representation was first introduced to the layout literature by Tam (1992). A slicing structure for a layout is constructed by recursively partitioning the rectangular blocks (starting with the facility floor) vertically or horizontally. The slicing

structure is represented by a binary tree, where leaves represent the departments and nodes represent the direction of the cut and the relative position of the departments (the children of the node). An example tree and layout is given in Figure 3.3. In this figure, N(orth), E(ast), S(outh) and W(est) indicate the direction of the cut. For example, the right branch can be read as "5 is to the north of 4, which are together to the east of the rest of the departments"

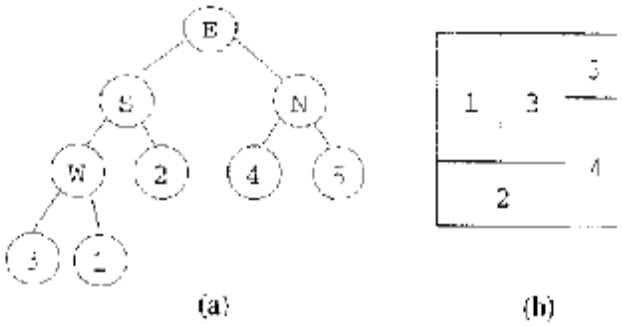


Figure 3.3 – Example of a slicing tree (a) and the corresponding layout (b) (Gau and Meller, 1999)

Similar to the flexible bay representation, this representation is also claimed to provide more suitable aisle shapes as opposed to a block layout.

Effect of the Selected Representation

The representation scheme chosen has a direct impact on the final shape of the layout. Because, the representation used restricts certain features of a layout, such as the shape or relative positioning of the departments, up to a certain extent. Examples of this situation are illustrated in Figures 3.4 and 3.5. Figure 3.4 shows a layout obtained by using grid representation. As it can be seen from the figure, there is no specific shape a department takes when grid representation is used. Layouts in Figure 3.5 are examples for the capabilities of the representations. While flexible-

bay can only represent (a), slicing tree is able to represent (a) and (b); whereas (c) can not be represented by either. So, it can be concluded that selection of the representation scheme is a very important step and it usually makes it meaningless to compare layouts constructed with different representations schemes.

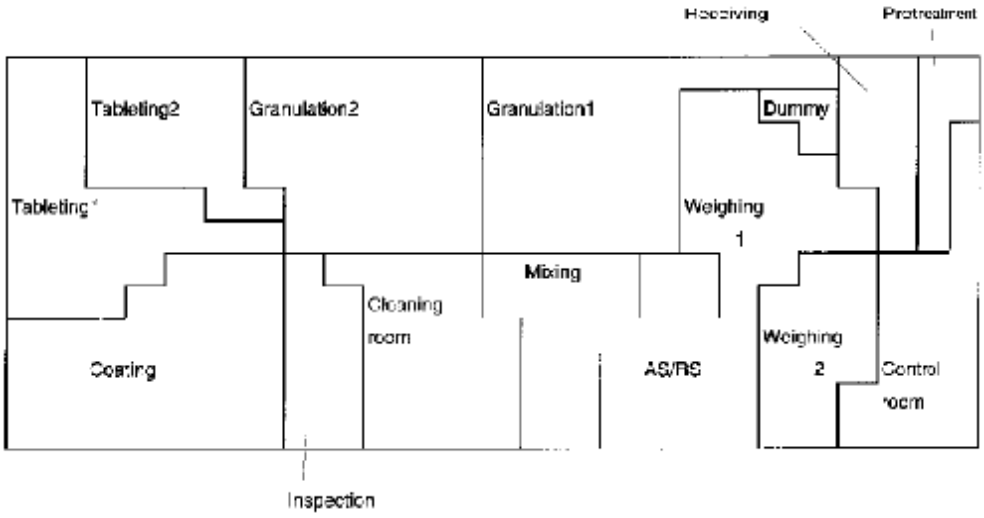


Figure 3.4 – An example layout obtained by using grid representation (Hamamoto et al., 1999)

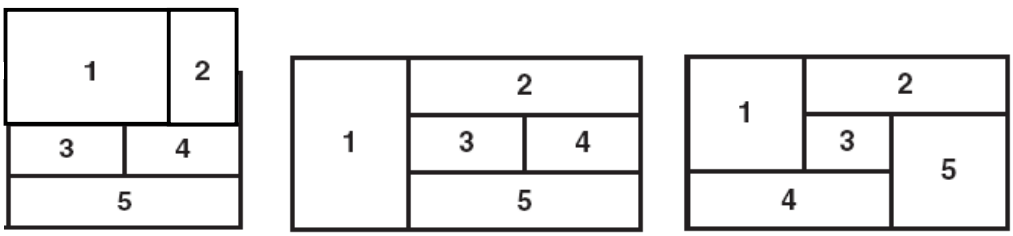


Figure 3.5 – Different layout types with rectangular shaped departments (Layouts (b) and (c) are taken from Kim and Goetschalckx, 2005)

3.1.2 Differences in the Objective Functions

Different objective functions are also one of the factors that complicate the comparability of different approaches. Several objective functions used in the literature were discussed before. Rosenblatt (1979) and Meller and Gau (1996b) are example studies which consider both the adjacency score and the material handling cost. Fu and Kaku (1997) and Benjaafar (2002) discuss the extent to which the objective functions of minimizing the material handling cost and minimizing the WIP inventory level (or congestion) in the facility are correlated with each other. Findings of Benjaafar (2002) are very convincing that these objective functions can lead one to completely different layouts.

3.1.3 Differences in Handling Constraints

One of the issues that cause a large variability is how the constraint of total area is represented for the departments whose length and width values are to be determined. This usually becomes a differentiating issue for mathematical models that are designed for solving the problem for optimality. This issue can be handled by linearizing these constraints in the integer programming models. Several linearizations that exist in the literature are discussed in the following section. Another approach is keeping these constraints as they are and using nonlinear programming approaches. Examples of nonlinear approaches are van Camp et al. (1991), Anjos and Vannelli (2006) and Castillo et al. (2005).

3.1.4 Differences Regarding Inclusion/Exclusion of Detailed Design Issues

Most commonly incorporated aspects into a layout problem are the simultaneous decision of I/O points and aisle shapes. I/O points, namely, input/output points, refer to the locations where the material loading and unloading from/to the department takes place, which are usually placed on department boundaries. Inclusion of I/O points and aisle shapes directly affects the distances used for cost calculation. However, it is not reasonable to compare the material handling costs calculated by

using center to center distances as opposed to the distances measured between I/O points that are traversed on the aisles even on the exactly same layout, yet alone two different layouts. Also, inclusion of the material handling method would alter the definition of distances (see Özdemir et al., 2003).

3.1.5 Differences in the Data Sets Used

Another reason which makes most of the studies in the literature incomparable is the lack of benchmark data sets. There are a few problems that are commonly used, though, such as the 10 department problem of van Camp et al. (1991), the 20 department problem of Armour and Buffa (1963), the problems up to 14 departments in Bazaraa (1975), and the problems ranging from size 3 to 10 in Sherali et al (2003). Also, the QAP instances of Nugent et al. (1968) are very commonly used, mostly for the flow and cost matrices by defining department/facility areas and shapes appropriate for the problem structure considered. It is of course not reasonable to expect each study to work with the same assumptions. On the other hand, data sets could be redefined so as to be consistent at least within each assumption.

In conclusion, it is very nice to have varieties of studies available on a broad range of assumptions, but the literature needs a more unifying point of view to be able to get the best from the available studies on hand.

In this study, we concentrate on the block layout problem without any inclusion of detailed design issues. The objective function is taken as the total material handling cost, where rectilinear distance metric is used and distances are measured between department centers. The area requirements of the departments are given, where their length and width are decision variables, subject to a limit on the aspect ratio. Hereafter, this problem will simply be referred as “the block layout problem”.

The problem has been modeled as an MIP by Montreiul for the first time in 1990 (see Meller and Gau, 1996a). However, developing exact solution methods has not been an active research subject until 1999 (see Meller et al., 1999), which is then

continued by Sherali et al. (2003) and Castillo and Wasterlund (2005). Although Lacksonen (1994) also provides a formulation for single and multiple periods, he solves the model in two stages, firstly by determining the relative locations of the departments and then solving the model. In the next section, we review MIP models in the literature.

3.2 MIP Models for the Block Layout Problem

A mathematical model for the block layout problem has the following constraint types:

- Departments should be placed within facility boundaries.
- Distance between departments should be measured.
- Aspect ratio requirements of departments should be satisfied.
- Area requirements of departments should be satisfied.
- Departments should not overlap (disjunctive constraints).

Ensuring that departments are within facility boundaries is quite straightforward to achieve by using a constraint that implies each department corner lies within the facility. Measuring the distance between departments is also a simple job if the rectilinear distance metric is used, in which the distance between two departments is projected separately onto each axis and they are summed up. Aspect ratio requirements are satisfied by specifying lower and upper bounds on the department width and length values. These three types of constraints are handled in the same way as described above, in all block layout problems. The last two are the types of constraints that differentiate developed models from each other. Below we consider them in detail.

3.2.1 Linearization Schemes

Note that the constraint for the area requirements of the departments to be satisfied is nonlinear, since area is the product of length and width. Several linearization schemes for this constraint have been proposed. Usually, all linearizations tend to underestimate area requirements. The reason is the objective function of all these

models being directly related with the distance between departments, which can be reduced by decreasing department length and widths.

The first linearization dates back to Montreiul, who is the first to model the layout problem as an MIP, in 1990 (see Meller and Gau, 1996a). He suggests giving bounds on the perimeters of the departments as a surrogate constraint on the area requirement. Letting w_i be the width, h_i the height, a_i the area requirement, α_i the limit on the aspect ratio, pl_i the lower bound on perimeter and pu_i the upper bound on perimeter of department i, the surrogate area constraint and the bounds on perimeters are given in (3.1)-(3.3).

$$pl_i \le 2(w_i + h_i) \le pu_i \tag{3.1}$$

where

$$pl_i = 4\sqrt{a_i} \tag{3.2}$$

$$pu_i = 2\sqrt{a_i}(1+\alpha_i)/\sqrt{\alpha_i} \tag{3.3}$$

Lacksonen (1994) suggests piecewise linearization using one breakpoint; however, his model requires two additional binary variables for each department. This linearization is given in detail in section A.3 of the review on dynamic layout problems provided in Appendix A.

Meller et al. (1999) use improved bounds on perimeters that require a single continuous variable for each department, wh_i, which denotes the length of the longer department side. In addition to the definitions given above, let f be a constant, whose value was determined as 0.95 by trial-and-error. Then the related constraints are given in (3.4)-(3.6).

$$wh_i \ge w_i \tag{3.4}$$

$$wh_i \ge h_i \tag{3.5}$$

$$2(w_i + h_i) \ge 3\sqrt{a_i} + f * wh_i \tag{3.6}$$

Sherali et al. (2003) propose using tangential supports. These supports are generated uniformly between the lower and upper bound values of the department width, where the accuracy of the linearization increases as the number of supports is increased. Letting Δ be the number of supports used, x_Δ the points (width values) these supports are generated, lbw_i the lower bound on the department width and ubw_i the upper bound on the department width, the tangential supports and the points they are generated are given in (3.7)-(3.8). Also, a graph that illustrates the linearization is provided in Figure 3.6.

$$x_\Delta = lbw_i + \frac{\lambda}{(\Delta - 1)}(ubw_i - lbw_i), \text{ where } \lambda = 0,1,\ldots,\Delta - 1 \text{ and } \Delta \geq 2 \qquad (3.7)$$

$$\frac{a_i w_i}{2} + \frac{(x_\Delta)^2 h_i}{2} \geq a_i x_\Delta \qquad (3.8)$$

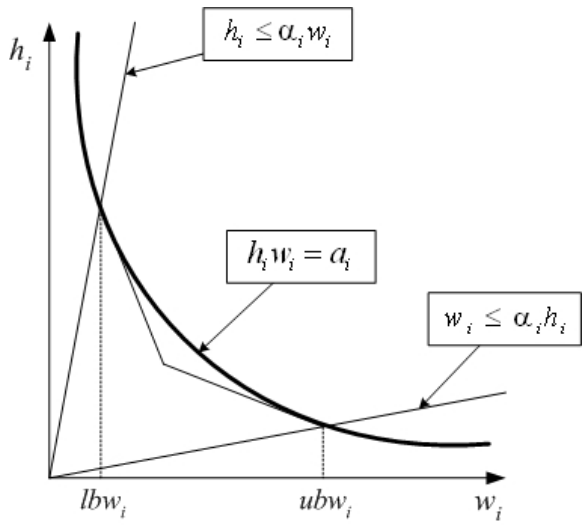


Figure 3.6 – A graphical example of the linearization by Sherali et al. (2003), two supports generated at lower and upper bound values of the department width

Finally, Castillo and Wasterlund (2005) propose a linearization that is designed to ensure areas within a specified range of maximum error. The concept the authors use is very compatible with facilities design: Area is a design parameter and its accuracy may not be 100%. It can be said that it is reasonable to let the user decide the level of accuracy to be used beforehand, instead of making him try-and-see. Another nice property of this linearization is being independent from the aspect ratio (α) used, where the aspect ratio refers to the ratio of a department's length over its width, or vice versa (whichever is maximum). Linearizations that bound the perimeter of a department are shown to become less accurate as α increases (see Meller et al., 1999).

Castillo and Wasterlund (2005) first relax the area constraints by converting them into an inequality, as given in (3.9), which is valid since the area requirements are always underestimated.

$$w_i h_i \geq a_i \tag{3.9}$$

Then, a cutting plane can be generated at point $\overline{w}_{i0}$ as given in (3.10).

$$-h_i - \frac{a_i}{\overline{w}_{i0}^2} w_i \leq -2\frac{a_i}{\overline{w}_{i0}} \tag{3.10}$$

The points at which cutting planes will be generated are selected such that the maximum area violation of a department is ε, where

$$1 - \frac{w_i h_i}{a_i} \leq \varepsilon \tag{3.11}$$

Two cuts at points $\overline{w}_{i0}$ and $\overline{w}_{i1}$ and the point corresponding to the maximum area deviation caused by these two cuts, $(\overline{w}_{i01}^{\max}, \overline{h}_{i01}^{\max})$ is illustrated in Figure 3.7, where P_1 and P_2 are the points that correspond to the lower and upper bound values of w_i, respectively.

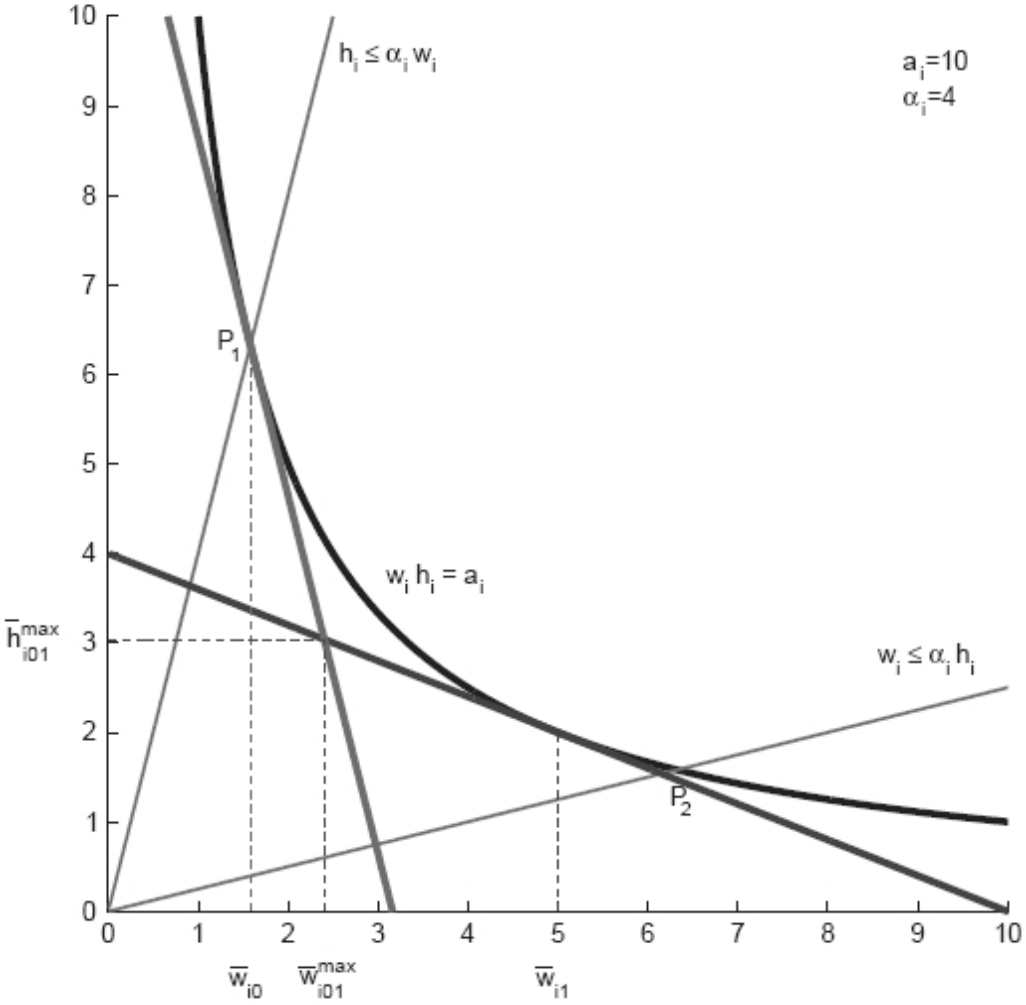


Figure 3.7 – An illustration of the ε-accurate area linearization

3.2.2 Handling Department Overlap Prevention

Note that the constraints for preventing department overlap are disjunctive constraints. Montreiul (see Meller and Gau, 1996a), Meller et al. (1999) and Sherali et al. (2003) use a pair of binary variables for department pairs, each variable indicating whether a department precedes the other on the axis the variable is defined for. So, the total number of binary variables used is 2*n*(*n*-1). Castillo and Wasterlund (2005) reduce the number of binary variables by half, to *n*(*n*-1), by defining the

binary variables only once for each department pair, i.e., defining only a variable x_{ij} instead of both x_{ij} and x_{ji}. However, the number of possible combinations of these binary variables that actually can correspond to a layout is $2^{n(n-1)}$ for both.

Recently, Meller et al. (2006) have developed a model for the block layout problem that benefits from the sequence-pair representation for preventing department overlaps. A sequence pair (Γ_+, Γ_-), introduced by Murata et al. (1995), is described as "a pair of entity sequences that can be used to determine the relative location of entities in the two-dimensional space." in Meller et al. (2006).

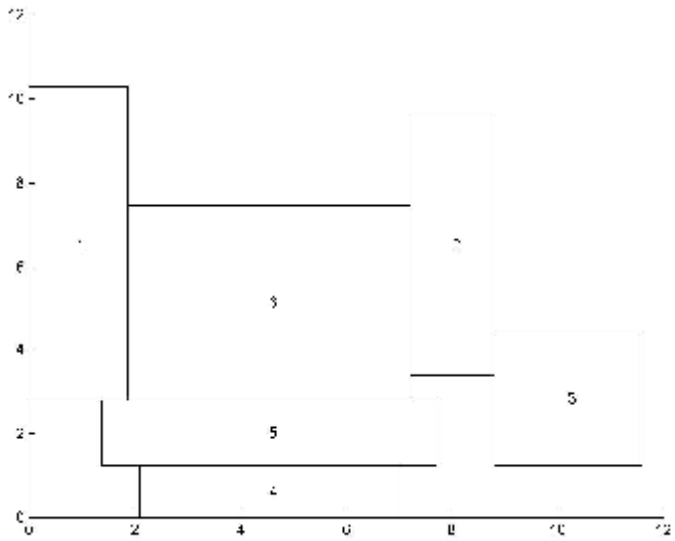


Figure 3.8 – The layout corresponding to a sequence pair of ((162534), (451623))

"Given a sequence-pair, (Γ_+, Γ_-), and two different departments *i* and *j* in (Γ_+, Γ_-), then *i* and *j* satisfy the following horizontal/vertical relationship in a facility layout:

- If *j* is after *i* in both Γ_+ and Γ_-, then department *j* is to the right of department *i*.

- If *j* is before *i* in both Γ_+ and Γ_-, then department *j* is to the left of department *i*.
- If *j* is before *i* in Γ_+ and after *i* in Γ_-, then department *j* is above department *i*.
- If *j* is after *i* in Γ_+ and before *i* in Γ_-, then department *j* is below department *i*."

For example, the sequence pair corresponding to the layout illustrated in Figure 3.8 is ((162534), (451623)).

An easier way of illustrating the relative positions of the departments is to draw an oblique grid, writing each sequence on an axis and marking the intersection points. The oblique grid representation of the example sequence pair in Figure 3.6 is shown in Figure 3.9.

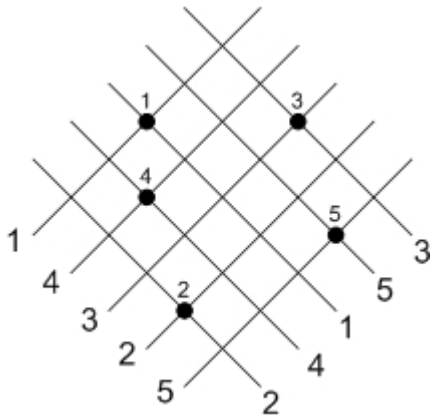


Figure 3.9 – The oblique grid representation corresponding to the sequence pair ((162534), (451623))

Sequence pair representation is shown to have the same number of binary variables ($2n(n-1)$) as Montreiul (see Meller and Gau, 1996a), Meller et al. (1999) and Sherali et al. (2003). However, the number of possible combinations of these binary

variables that actually can correspond to a layout is $(n!)^2$, which is much less than $2^{n(n-1)}$ and hence corresponds to a tighter formulation.

How a sequence pair representation is converted into a block layout

When length and width of each department along with its orientation is known, there are several algorithms for converting a sequence pair representation into a layout (i.e. specifying the coordinates of the department centers). The most efficient algorithm developed for this conversion is given by Kodama and Fujiyoshi (2003), which is of complexity O(n). They also give a brief chronology of such algorithms as follows.

> In Murata et al. (1995), a method to get the bottom left corner packing (there is no block that can shift left and down from its original position with other components fixed) from a given seq-pair of n rectangles in $O(n^2)$ time was proposed using horizontal/vertical constraint graphs. Later, another method to get packing in $O(n \log n)$ time (in 1996) was proposed. Recently a study (in 2001) showed that a packing can be obtained in $O(n \log \log n)$ time.

In conclusion, although the MIP models are hard to solve, they have been improved recently, with respect to the tightness of the formulations and the capability to represent the actual problem accurately. However, the models are still not able to solve problems larger than 11 departments (solved by Meller et al., 2006) to optimality.

In this thesis, the overlap prevention of the departments is achieved through the model given by Meller et al. (2006) and linearization of the area constraint is done as in Castillo and Wasterlund (2005). This way, we propose a new hybrid model that combines the strong properties of both studies.

3.3 Mathematical Model Proposed

In this section, we present a new model that is based on Meller et al. (2006) and Castillo and Wasterlund (2005). Below, we first give index sets, parameters, and decision variables. Then, the mixed integer programming model of the single period facility layout problem is given.

Parameters

i, j : Department indices ($i, j = 1, 2, \ldots, n$)

s : Dimension (axis) indices ($s = x, y$)

$\pm$: Sequence-pair indices ($\pm = +, -$)

f_{ij} : Total flow between departments i and j (multiplied with unit cost values if unit material handling cost differs among department pairs)

a_i : Area that should be occupied by department i, where $l_i^x l_i^y = a_i$ and l_i^s are decision variables explained below

L^s : Width ($s=x$) and length ($s=y$) of the facility in which the departments will be placed

α_i : Upper limit on the aspect ratio of a department, where $\alpha_i = Max\left\{\left(l_i^x / l_i^y\right), \left(l_i^y / l_i^x\right)\right\}$ and l_i^s is a decision variable explained below

ε : Desired level of accuracy for the area of departments

lb_i^s, ub_i^s : Lower and upper bound values for department half width and length. These values are calculated by using equations (3.12)-(3.14). Basic idea is based on the fact that the limiting conditions on the value a department width/length can take are dependent on α. At the limit value of α, one side is at its lower bound and the other is at its upper bound. So, the bounds are calculated by the two conditions that describe the situation: $2lb_i^s \; x \; 2ub_i^s = a_i$ and $ub_s^i / lb_s^i = \alpha_i$. The second condition on the upper bound states that width/length of a department cannot exceed the facility width/length.

$$ub_i^s = Min\left\{\frac{\sqrt{\alpha_i a_i}}{2}, \frac{L^s}{2}\right\} \qquad \forall\ i,s \qquad (3.12)$$

$$lb_i^x = \frac{a_i}{4ub_i^y} \qquad \forall\ i \qquad (3.13)$$

$$lb_i^y = \frac{a_i}{4ub_i^x} \qquad \forall\ i \qquad (3.14)$$

N_i : Number of constraints required for the linearization of area constraint for department i. Its value is given by equation (3.15), where $\lceil . \rceil$ corresponds to the smallest integer value that is greater than or equal to the function inside.

$$N_i = \left\lceil \frac{\ln\left(ub_i^x / lb_i^x\right)}{\ln\left(\left(1+\sqrt{\varepsilon}\right)/\left(1-\sqrt{\varepsilon}\right)\right)} \right\rceil \qquad \forall\ i \qquad (3.15)$$

$\bar{l}_{ik}$: Points at which cutting planes for the linearization of the area constraint are generated. Equation (3.16) gives how they are calculated, where $\bar{l}_{i0} = lb_i^x$.

$$\bar{l}_{ik} = \bar{l}_{i,k-1}\left(\frac{ub_i^x}{lb_i^x}\right)^{1/N_i} \qquad \forall\ i;\ k = 1,\ldots,N_i \qquad (3.16)$$

Decision variables

c_i^s : s-axis coordinate of the center of department i

d_{ij}^s : distance between the centers of departments i and j on s-axis

l_i^s : half length of department i on s-axis

$z_{ij}^{\pm}$: binary variable showing whether department i is before department j in the sequence ±

The mathematical model is given as follows.

SP-ε:

$$\text{Min} \sum_{\forall i} \sum_{\forall j>i} f_{ij}\left(d_{ij}^{x} + d_{ij}^{y}\right) \tag{3.17}$$

subject to

$$d_{ij}^{s} = \left| c_i^s - c_j^s \right| \qquad \forall\ i<j; \forall\ s \tag{3.18}$$

$$l_i^s \le c_i^s \le L^s\text{-}l_i^s \qquad \forall\ i; \forall\ s \tag{3.19}$$

$$lb_i^s \le 2l_i^s \le ub_i^s \qquad \forall\ i; \forall\ s \tag{3.20}$$

$$z_{ij}^{\pm} + z_{ji}^{\pm} = 1 \qquad \forall\ i<j; \forall \pm \tag{3.21}$$

$$z_{ik}^{\pm} \ge z_{ij}^{\pm} + z_{jk}^{\pm}\text{-}1 \qquad \forall\ (distinct)\ i, j, k; \forall \pm \tag{3.22}$$

$$c_i^x + l_i^x \le c_j^x - l_j^x + L^x\left(2 - z_{ij}^{+} - z_{ij}^{-}\right) \qquad \forall\ i \ne j \tag{3.23}$$

$$c_i^y + l_i^y \le c_j^y - l_j^y + L^y\left(1 + z_{ij}^{+} - z_{ij}^{-}\right) \qquad \forall\ i \ne j \tag{3.24}$$

$$l_i^y + \frac{a_i}{\left(\bar{l}_{ik}\right)^2} l_i^x \ge \frac{a_i}{\bar{l}_{ik}} \qquad \forall\ i;\ k = 1,\ldots,N_i \tag{3.25}$$

$$c_i^s, l_i^s, d_{ij}^s \ge 0 \qquad \forall\ i; \forall\ j; \forall\ s \tag{3.26}$$

$$z_{ij}^{\pm} \in \{0,1\} \qquad \forall\ i; \forall\ j; \forall\ \pm \tag{3.27}$$

Note that, expressions (3.12)-(3.14) and (3.17)-(3.20) are common in both models given by Meller et al. (2006) and Castillo and Wasterlund (2005). Constraints (3.21)-(3.24) are taken from Meller et al. (2006) and the rest, namely, (3.15), (3.16) and (3.25) are taken from Castillo and Wasterlund (2005).

In the model, (3.17) calculates the total material handling cost, based on the rectilinear distance and amount of flow between each department pair. Note that, here, a symmetrical flow matrix is assumed. (3.18) calculates the distance between the centroids of two departments on each axis. During implementation, this constraint was linearized as given by equations (3.28) and (3.29).

$$d_{ij}^s \ge c_i^s - c_j^s \qquad \forall\ i; \forall\ j; \forall\ s \tag{3.28}$$

$$d_{ij}^s \ge c_j^s - c_i^s \qquad \forall\ i; \forall\ j; \forall\ s \tag{3.29}$$

(3.19) ensures that departments lie within the boundaries of the facility floor plan. (3.20) imposes upper and lower bounds on the half width of a department. These bounds are computed by using equations (3.12)-(3.14). (3.21) ensures that if department *i* is placed before department *j*, then department *j* should be placed after department *i* in a sequence. (3.22) ensures that the relations between relative locations of the departments with respect to each other are transitive. Here, in a sequence ±, if department *i* is to the left of department *j*, and department *j* is to the left of department *k*, then department *i* should be to the left of department *k*. This relation is ensured by forcing $z_{ik}^{\pm}$ to be equal to 1 if both $z_{ij}^{\pm}$ and $z_{jk}^{\pm}$ are equal to 1. (3.23) ensures that departments *i* and *j* are separated on the x-axis if the sequence-pair implies so. The sequence pair implies department *i* to be located to the left of department *j* if *i* comes before *j* in both of the sequences. This situation occurs when both z_{ij}^{+} and z_{ij}^{-} are equal to 1. So, the constraint becomes binding when both are equal to 1 and non-binding if the condition does not hold. (3.24) is the same as (3.23), except that it is given for y-axis. Condition for the constraint to be binding, i.e. department *i* being located below department *j*, is z_{ij}^{+} being 0 and z_{ij}^{-} being 1. (3.25) is the linearization of department area requirements, which are calculated by using (3.15) and (3.16). Finally, (3.26) and (3.27) are nonnegativity and integrality constraints.

It is possible to tighten model SP-ε by adding valid inequalities. Several types of valid inequalities for the facility layout models were developed by Meller et al. (1999). However, the study by Sherali et al.(2003) reports that adding all types of valid inequalities lead to a drastic increase in the overall solution effort. Therefore, they identify three groups of inequalities which perform well and enhance the performance. Meller et al. (2006) also choose to include these three inequalities, and it is decided to use the same set in this study, too, details of which are given below.

1. *Symmetry breaking constraints*

   These constraints are used to reduce the solution space by eliminating symmetrical layouts that have the same material handling cost. The following

constraints select the departments, namely p and q, which have the largest flow amount between them and imply the centroid of department p to be placed to the southwest of the centroid of department q.

$$c_p^s \le c_q^s \qquad \forall s \qquad (3.30)$$

$$\sum_{s=x}^{y} \left(c_q^s - c_p^s\right) \ge \min\left\{\left(lb_p^x + lb_q^x\right), \left(lb_p^y + lb_q^y\right)\right\} \qquad (3.31)$$

$$z_{pq}^- = 1 \qquad (3.32)$$

2. Variable distance constraints

These constraints are added to make the distance variables take values different than 0. They ensure that two departments have a distance of at least the sum of their half lengths on an axis, if they are to be separated on that axis.

$$d_{ij}^x \ge l_i^x + l_j^x - \left(\min\left\{\left(ub_i^x + ub_j^x\right), L^x\right\}\right)\left(2 - z_{ij}^+ - z_{ij}^-\right) \qquad \forall i < j \qquad (3.33)$$

$$d_{ij}^x \ge l_i^x + l_j^x - \left(\min\left\{\left(ub_i^x + ub_j^x\right), L^x\right\}\right)\left(2 - z_{ji}^+ - z_{ji}^-\right) \qquad \forall i < j \qquad (3.34)$$

$$d_{ij}^y \ge l_i^y + l_j^y - \left(\min\left\{\left(ub_i^y + ub_j^y\right), L^y\right\}\right)\left(1 + z_{ij}^+ - z_{ij}^-\right) \qquad \forall i < j \qquad (3.35)$$

$$d_{ij}^y \ge l_i^y + l_j^y - \left(\min\left\{\left(ub_i^y + ub_j^y\right), L^y\right\}\right)\left(1 + z_{ji}^+ - z_{ji}^-\right) \qquad \forall i < j \qquad (3.36)$$

3. Distance bound constraints

This group of constraints is the same as the previous one, except that the lower bound values of the department half width and lengths are used instead of the actual half width and lengths.

$$d_{ij}^x \ge \left(lb_i^x + lb_j^x\right)\left(z_{ij}^+ + z_{ij}^- - 1\right) \qquad \forall i < j \qquad (3.37)$$

$$d_{ij}^x \ge \left(lb_i^x + lb_j^x\right)\left(z_{ji}^+ + z_{ji}^- - 1\right) \qquad \forall i < j \qquad (3.38)$$

$$d_{ij}^y \ge \left(lb_i^y + lb_j^y\right)\left(z_{ij}^- - z_{ij}^+\right) \qquad \forall i < j \qquad (3.39)$$

$$d_{ij}^y \ge \left(lb_i^y + lb_j^y\right)\left(z_{ji}^- - z_{ji}^+\right) \qquad \forall i < j \qquad (3.40)$$

Valid inequality set 1 requires 4 constraints, set 2 and 3 require $2n(n\text{-}1)$ constraints each; so the total number of constraints added to the model is $4n(n\text{-}1) + 4$ when valid inequalities are used. For more detailed information regarding the inequalities (3.30)-(3.40), the reader is referred to Sherali et al. (2003) and Meller et al. (1999).

3.4 Computational Results for the MIP and its LP Relaxation

In this section, we present our computational results for evaluating the model SP-ε and its LP relaxation performances under different measures. The test problem instances used in the experimentation are taken from the literature. Namely, m3, m4, m5, m6, m7, o7, o8, o9, fo7, fo8, fo9 from Sherali et al. (2003), fo10, fo11, o10 from Meller et al. (2006), vc10 from van Camp et al. (1991) and ab20 from Armour and Buffa (1963). Optimization models are generated and solved by using CPLEX 8.1, with keeping all default features. Instances were tested on an HP Compaq® PC, with an Intel Pentium® 4 processor of 2.80 GHz and 1.0 GB of RAM, with Microsoft Windows XP® Professional Version 2002 as the OS.

In the first part of our experimentation, the instances are tried to be solved to optimality to see whether using the ε-accurate representation of department areas together with the sequence pair leads us to the same conclusions with the original models by Meller et al. (2006) and Castillo and Wasterlund (2005).

Problems o7-o9 are solved by both with and without valid inequalities to observe the effect of adding valid inequalities and the results are reported in Table 3.1. In most cases and on average, a decrease in the total runtime and the number of nodes is achieved with the addition of valid inequalities.

The results with respect to solution quality and effect of changing epsilon concur with the findings of Castillo and Wasterlund (2005). The objective function value (total cost) increases or stays the same, as the accuracy of the representation is increased (i.e, epsilon is decreased), which is “due to the lesser underestimation of required areas with smaller values of epsilon”. Although not observed strictly for all

instances, it still can be concluded that the CPU time increases as epsilon is decreased, which is also in line with the findings of Castillo and Wasterlund (2005).

Another observation is that the CPU time increases more rapidly as epsilon is decreased when valid inequalities are not used. Using valid inequalities makes the CPU time more "stable".

Table 3.1 – Comparison of using and not using valid inequalities

P	ε	Without VI		With VI	
		#NODE	CPU	#NODE	CPU
o7	0.5%	2,562,600	1,679.8	1,725,400	1,470.3
	1.0%	3,391,600	2,129.1	1,920,500	1,540.7
	5.0%	2,094,500	1,108.8	1,933,500	1,778.6
	10.0%	1,482,300	768.6	1,166,700	897.4
o8	0.5%	24,638,900	21,606.5	22,279,900	14,403.7
	1.0%	25,352,800	21,267.6	8,959,600	12,975.0
	5.0%	23,044,300	16,815.1	12,326,300	13,506.1
	10.0%	25,189,500	19,163.8	13,679,700	14,402.5
o9	0.5%	36,328,300	40,805.6	12,784,000	14,403.6
	1.0%	NA*	NA*	12,910,300	14,404.1
	5.0%	31,459,600	31,927.7	16,371,300	14,403.7
	10.0%	12,723,000	12,446.1	6,123,500	11,752.4
AVG	0.5%	21,176,600	21,364.0	12,263,100	10,092.5
	1.0%	14,372,200	11,698.4	7,930,133	9,639.9
	5.0%	18,866,133	16,617.2	10,210,367	9,896.1
	10.0%	13,131,600	10,792.8	6,989,967	9,017.4

* could not be solved due to default memory limit of CPLEX

P Problem instance
ε Accuracy level of the area representation
VI Valid inequalities
#NODE Number of nodes analyzed
CPU Runtime in seconds of CPU

The LP relaxation solution values of the problem instances solved are all equal to zero when valid inequalities are not added to the model. This can be explained as

follows: Department overlaps are not prevented when binary variables are relaxed and since the objective function only depends on the distances between departments, the optimal solution turns out to be trivial, which is placing all department centroids on the same coordinate. The LP relaxation values of the problem instances that can be solved to optimality and related measures are reported on Table 3.2.

The average integrality gap (IGAP, the distance of the LP relaxation solution from the optimal solution) is large, which is around 78% for all values of ε. It is also noted that the LP relaxation value does not change for different ε values of the same problem. This results in a slightly decreasing pattern for IGAP as ε increases.

Table 3.2 – LP relaxation solution values of solvable problem instances

P	ε	With VI		LPREL	OPT	IGAP	IGAPLP
		#NODE	CPU				
m3	**0.5%**	6	0.0	14.3	37.80	62.3%	165.1%
	1.0%	6	0.0	14.3	37.80	62.3%	165.1%
	5.0%	6	0.0	14.3	37.80	62.3%	165.1%
	10.0%	6	0.2	14.3	37.80	62.3%	165.1%
m4	**0.5%**	20	0.0	0.0	51.09	100.0%	∞
	1.0%	20	0.0	0.0	51.09	100.0%	∞
	5.0%	19	0.0	0.0	51.09	100.0%	∞
	10.0%	20	0.0	0.0	51.09	100.0%	∞
m5	**0.5%**	100	0.1	0.0	61.74	100.0%	∞
	1.0%	100	0.1	0.0	61.65	100.0%	∞
	5.0%	100	0.1	0.0	61.47	100.0%	∞
	10.0%	30	0.1	0.0	61.02	100.0%	∞
m6	**0.5%**	1,780	0.2	22.3	82.26	72.8%	268.1%
	1.0%	136	0.2	22.3	82.26	72.8%	268.1%
	5.0%	370	0.3	22.3	82.21	72.8%	267.8%
	10.0%	200	0.2	22.3	82.15	72.8%	267.6%
m7	**0.5%**	2,600	2.0	22.3	106.76	79.1%	377.7%
	1.0%	1,043	0.8	22.3	106.76	79.1%	377.7%
	5.0%	1,100	0.8	22.3	106.76	79.1%	377.7%
	10.0%	800	0.7	22.3	106.76	79.1%	377.7%

Table 3.2 (continued)

P	ε	With VI		LPREL	OPT	IGAP	IGAPLP
		#NODE	CPU				
fo7	**0.5%**	54,700	43.3	2.5	20.71	87.9%	728.5%
	1.0%	45,000	33.6	2.5	20.53	87.8%	721.1%
	5.0%	57,800	43.2	2.5	20.40	87.7%	716.1%
	10.0%	57,600	40.2	2.5	19.57	87.2%	682.6%
fo8	**0.5%**	176,100	189.6	3.0	22.29	86.5%	642.9%
	1.0%	177,800	178.4	3.0	22.25	86.5%	641.8%
	5.0%	281,000	255.6	3.0	22.19	86.5%	639.7%
	10.0%	188,600	187.5	3.0	22.19	86.5%	639.7%
fo9	**0.5%**	1,002,300	1,470.8	3.0	23.46	87.2%	682.1%
	1.0%	1,447,500	1,981.7	3.0	23.30	87.1%	676.8%
	5.0%	621,700	960.3	3.0	23.13	87.0%	670.8%
	10.0%	759,400	1,038.8	3.0	22.91	86.9%	663.5%
o7	**0.5%**	1,725,400	1,470.3	15.0	130.72	88.5%	771.5%
	1.0%	1,920,500	1,540.7	15.0	126.38	88.1%	742.6%
	5.0%	1,933,500	1,778.6	15.0	130.64	88.5%	770.9%
	10.0%	1,166,700	897.4	15.0	120.85	87.6%	705.7%
o8	**0.5%**	22,279,900	14,403.7	27.5	242.77	88.7%	782.8%
	1.0%	8,959,600	12,975.0	27.5	242.19	88.6%	780.7%
	5.0%	12,326,300	13,506.1	27.5	239.12	88.5%	769.5%
	10.0%	13,679,700	14,402.5	27.5	238.25	88.5%	766.4%
o9	**0.5%**	12,784,000	14,403.6	27.5	235.87	88.3%	757.7%
	1.0%	12,910,300	14,404.1	27.5	235.55	88.3%	756.6%
	5.0%	16,371,300	14,403.7	27.5	234.16	88.3%	751.5%
	10.0%	6,123,500	11,752.4	27.5	219.94	87.5%	699.8%
AVG	**0.5%**	3,168,909	2,665.3	11.5	84.62	78.4%	517.6%
	1.0%	2,121,834	2,592.9	11.5	84.15	78.4%	513.0%
	5.0%	2,632,766	2,579.1	11.5	84.08	78.4%	512.9%
	10.0%	1,831,380	2,360.0	11.5	81.88	78.2%	496.8%

P Problem instance
ε Accuracy level of the area representation
VI Valid inequalities
#NODE Number of nodes analyzed
CPU Runtime in seconds of CPU
LPREL LP relaxation solution value with VI
OPT Objective function value of the optimal solution
IGAP (OPT – LPREL) / OPT
IGAPLP (OPT – LPREL) / LPREL

Since Meller et al. (2006) do not report the accuracy levels of the area linearization constraints, the results obtained from our model are compared with the results given in Sherali et al. (2003), the study from which Meller et al. (2006) adapt the area linearization constraints. Common instances and their statistics are listed in Table 3.3, where no valid inequalities are used in either of the linearization methods. Number of nodes cannot be compared since Sherali et al. (2003) do not report them. Computation times were tried to be put on a similar scale but there is no instance that gives the same maximum error in both linearization methods. In addition, since solution times for m3-m7 instances are less than 1 second in our model, making comparisons can be misleading. The most sound basis of comparison is determined as the number of cuts generated by each method, where using an ε-accurate representation clearly dominates generating supports.

Table 3.3 – Comparison of ε-accurate and support generating area linearizations

P	LIN-M	CPU	OPT	%MAX-ERR	#CUT
m3	**S-10**	0.06	37.75	0.62	30
	S-20	0.04	37.78	0.09	60
	S-50	0.06	37.80	0.02	150
	ε-10%	0.00	37.80	0.00	9
	ε-5%	0.00	37.80	0.00	10
	ε-1%	0.00	37.80	0.00	18
	ε-0.5%	0.00	37.80	0.00	22
m4	**S-10**	0.04	51.03	0.62	40
	S-20	0.06	51.06	0.09	80
	S-50	0.08	51.09	0.02	200
	ε-10%	0.00	51.09	0.00	10
	ε-5%	0.00	51.09	0.00	11
	ε-1%	0.00	51.09	0.00	19
	ε-0.5%	0.00	51.09	0.00	23

Table 3.3 (continued)

P	LIN-M	CPU	OPT	%MAX-ERR	#CUT
m5	**S-10**	0.16	61.72	0.06	50
	S-20	0.24	61.71	0.09	100
	S-50	0.35	61.75	0.01	250
	ε-10%	0.00	61.02	6.04	13
	ε-5%	0.00	61.47	2.06	15
	ε-1%	0.00	61.65	0.77	26
	ε-0.5%	0.00	61.74	0.12	32
m6	**S-10**	0.37	82.12	0.37	60
	S-20	1.20	82.22	0.23	120
	S-50	1.80	82.25	0.05	300
	ε-10%	0.00	82.15	4.55	19
	ε-5%	0.00	82.21	2.39	24
	ε-1%	0.00	82.26	0.00	40
	ε-0.5%	0.00	82.26	0.00	55
m7	**S-10**	1.00	106.58	0.37	70
	S-20	2.30	106.72	0.23	140
	S-50	1.90	106.74	0.05	350
	ε-10%	0.52	106.76	0.00	22
	ε-5%	0.67	106.76	0.00	27
	ε-1%	2.23	106.76	0.00	45
	ε-0.5%	0.52	106.76	0.00	61
fo7	**S-10**	2100.00	20.94	0.12	70
	S-20	3600.00	20.95	0.05	140
	S-50	2300.00	20.95	0.01	350
	ε-10%	39.44	19.57	4.07	27
	ε-5%	43.47	20.40	1.56	34
	ε-1%	33.55	20.53	0.83	55
	ε-0.5%	43.23	20.71	0.07	75
fo8	**S-10**	4700.00	22.27	0.57	80
	S-20	5100.00	22.31	0.40	160
	S-50	7100.00	22.38	0.03	400
	ε-10%	188.78	22.19	1.71	32
	ε-5%	256.66	22.19	1.71	38
	ε-1%	177.16	22.25	0.88	64
	ε-0.5%	185.52	22.29	0.47	88

Table 3.3 (continued)

P	LIN-M	CPU	OPT	%MAX-ERR	#CUT
AVG	**S-10**	971.66	54.63	0.39	57.14
	S-20	1243.41	54.68	0.17	114.29
	S-50	1343.46	54.71	0.03	285.71
	ε-10%	32.68	54.37	2.34	18.86
	ε-5%	42.97	54.56	1.10	22.71
	ε-1%	30.42	54.62	0.35	38.14
	ε-0.5%	32.75	54.66	0.09	50.86

P	Problem instance
LIN-M	The area linearization method used
S-x	x supports are generated
ε-x%	ε accuracy is used with a level of x%
CPU	Runtime in seconds, values S-x instances are reported from Sherali et al. (2003), where the models are evaluated using AMPL interface with CPLEX 6.5 on a SUN Ultra-2 workstation.
OPT	Objective function value of the optimal solution
%MAX-ERR	Maximum realized error percentage over the area requirements of departments
#CUT	Number of cuts generated for linearization

Analysis of the LP Relaxation of the Problem

In this section, the values binary variables take at the LP relaxation solution of a problem are analyzed. LP relaxations of all problem instances are solved and histograms of the values of binary variables are tabulated, which are given in Appendix C. The percentage of integer solutions and solutions with a value equal to 0.5 are also reported in the tables. Graphs of two histograms are drawn for illustration for problem instance o9 at an ε level of 0.5%, with and without adding valid inequalities, which are given in Figure 3.10.

Histograms are symmetrical with respect to the 0.5 value on x-axis, since all binary variables have a pair, together which they sum up to 1 (see equation (3.21)). This is due to the fact that for example, if department A is located to the left of department B, then department B should be located to the right of department B. It is also noted that the values tend to move towards the center when valid inequalities are added.

Percent of integer values change between 23% and 39% on the average for different ε values. It is seen that adding valid inequalities usually slightly decreases the percentage of the integer solutions and sets around 33% of the values of the variables to 0.5. Based on these results, a rounding heuristic was found worthwhile to analyze.

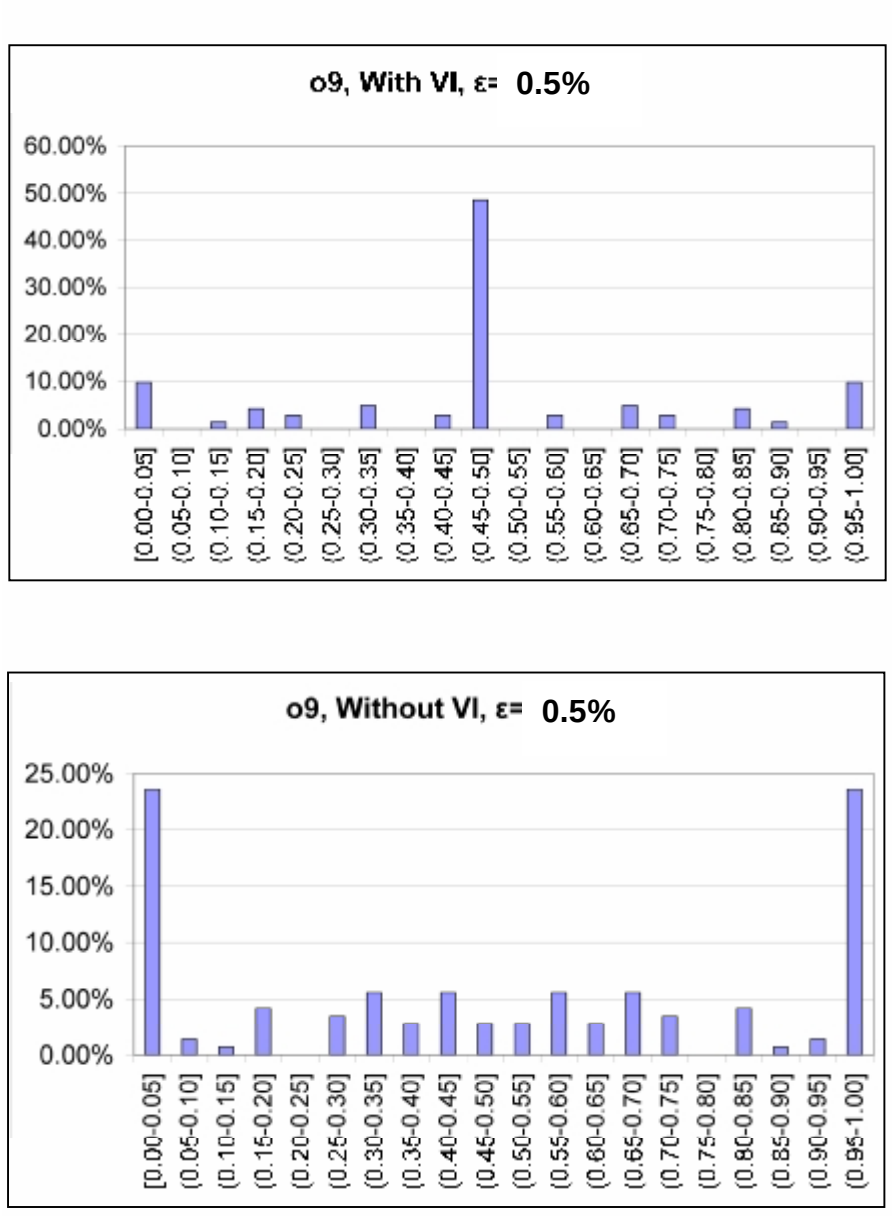


Figure 3.10 – Histograms of the LP relaxation solution values of problem instance o9, solved with and without using valid inequalities, at an ε value of 0.5%

3.4 LP Based Rounding Heuristic

When the research in the facility layout problem is analyzed, it is seen that although MIP models have received some attention, their LP relaxations have not been analyzed. This is reasonable, since the optimal objective function for the LP relaxation of this problem is always 0 if there are not any fixed departments in the problem instance and no valid inequalities are added to the model. This is due to the fact that the only variable type in the objective function is the distance between departments, which can easily be made 0 by putting all department centers on the same location. This also means that there are several alternative optima for this relaxation. The number of alternative optima for both MIP formulation and its LP relaxation increases if

- The available area is larger than the total area required by the departments (layout compactness decreases)
- Aspect ratio increases
- Department flows form groups. E.g., (1-2-5) interact only within each other and not with (3-4).

However, it was found worthwhile to analyze how a rounding heuristic would perform. A structure similar to the one used in Denizel and Süral (2006) was employed while constructing the heuristic. Two different strategies were experimented, namely, fixing the singles and fixing the multiples. In fixing the singles, the variable with the maximum value is chosen to be set equal to 1. Since all binary variables have a pair, this actually corresponds to fixing two variables simultaneously. In fixing multiples, the values of the binary variables are rounded and the ones with the maximum value are set to 1. This way, it is possible to fix more than two variables at each iteration. Fixing multiples is tested to see the effect of not differentiating among small differences between variables. Fixing is continued until the maximum value of a binary variable is higher than a specified threshold value. The variables whose values could not be set are then determined by applying truncated branch and bound. Details of the algorithm steps are provided below.

Algorithm for Fixing Singles

Step 1. Solve the LP relaxation of the model

Step 2. Fix the values of binary variables whose values are equal to 0 or 1

Step 3. Repeat until the maximum value taken by a binary variable is less than the threshold value

- Set the value of binary variable that takes the maximum value to 1 and its pair to 0 (i.e., if $z_{ij}^{s} = 1$, then $z_{ji}^{s} = 0$)
- Resolve the LP model
- If there is not a feasible solution obtained, reverse the fixation
  - If still not feasible, terminate
  - Else, continue
- Else, continue

Step 4. Let S_F be the set of binary variables fixed and S_{NF} be the set of binary variables free. Given S_F and S_{NF}, solve the corresponding truncated branch and bound.

Algorithm for Fixing Multiples

For this algorithm, Step 3 of Algorithm for fixing Singles should be replaced with the following step:

Step 3. Repeat until the maximum value taken by a binary variable rounded to two decimal digits is less than the threshold value set

- Set the values of the binary variables that are equal to the maximum value when rounded to two decimal digits to 1 and their pairs to 0 (i.e., if $z_{ij}^{s} = 1$, then $z_{ji}^{s} = 0$)
- Resolve the LP relaxation
- If there is not a feasible solution obtained, relax the most recently fixed variables. Apply the routine in Step 3 of Algorithm for fixing singles once.

How is feasibility ensured in fixing multiples?

There are three types of infeasibilities that can occur in the model:

- *Insufficient floor area:* This case occurs when the total area requirements of the departments exceed the total area available. This can be checked before solving an instance and be handled by increasing the ε value or just revising the area requirements.
- *Departments violating the aspect ratio:* In the model used, this is handled by the area linearization as a constraint, so it is not an issue here.
- *Given sequence pair imposes a layout that extends beyond floor area:* This case cannot be foreseen due to the facility width and lengths being decision variables. Imposing a sequence pair may lead to infeasibilities by departments violating the facility width and/or length boundaries. For example, two same sequences impose a row type layout, which usually is an infeasible alternative especially for larger problems.
- *Assigned values for binary variables conflict with each other:* For example, consider three departments, A, B and C. If we know that A comes before B and C comes after B, then A should certainly come before C. If not, then this is a meaningless sequence for which a corresponding layout cannot be obtained.

The third type of infeasibility needs to be analyzed further in fixing multiples, which is represented by the constraint $z_{ik} \geq z_{ij} + z_{jk}$-1 (± indexes are ignored for simplicity). This constraint forces z_{ik} to be 1 if both z_{ij} and z_{jk} take a value of 1. A conflicting situation might occur since the binary conditions on z_{ij} and z_{jk} are removed and the constraint ultimately may not impose any bound on z_{ik}. The problem would arise when the algorithm would try to set the values of z_{ij} and z_{jk} to 1 and the value of z_{ik} to 0 at the same iteration. Let's assume that we use a threshold value of $t > 1/2$. In order for z_{ik} to be set to 0, its value should be less than 1-t and similarly in order for z_{ij} and z_{jk} to be set to 1, their value should be more than t and the constraint $z_{ik} \geq z_{ij} + z_{jk}$-1 must still hold. When the left hand side is at its upper bound and the right hand side is at its lower bound, the inequality becomes $1 - t \geq t + t - 1$, which reduces to $t \leq 2/3$. This means that no bound may be

imposed on z_{ik} if the used threshold value is between 1/2 and 2/3. But, there is no problem as long as $t > 2/3$.

Note that this question does not deal with the problem of reaching an infeasible solution with respect to the facility boundaries due to the fact that some variables were/are going to be fixed in a way that leads the problem to total infeasibility. An infeasibility detecting routine in order to detect this kind of moves was tried to be developed, but could not have been achieved due to the added complexity by not knowing the width and length of the departments.

Addition of an Improvement Step

It is possible to improve the solutions obtained by algorithms by applying a recursive step. The proposed improvement step, which can be added to both of the algorithms, is as follows.

Step 5. Set the variables in S_F free and fix the variables in S_{NF} using their values obtained in Step 4.
- If termination condition is not met, set the variables in S_{NF} free and fix the variables in S_F using their values obtained in Step 5. Go to Step 4.

The ideal termination condition is the convergence of the solution. So, when there is no improvement in the last two steps, the algorithm should be stopped. However, this improvement step can be implemented in several ways:

- A fixed number of transitions between steps 4 and 5 can be imposed.
- The time allocated for steps 4 and 5 can be limited
- Steps 4 and 5 can be terminated when a certain optimality gap is reached.

If implementations like these are considered, determination of the related parameter settings, such as the time allocated to each step, becomes an important issue. In our implementation, we applied the improvement step only once, without checking the convergence. Also, CPU time limit of 2 hours was given to each of steps 4 and 5.

3.5 Experimentation and Findings

The problem instances considered in Section 3.3 were tested here, too, using the same computer. All algorithms were developed using C on Microsoft Visual C++ 6.0. Optimization models were solved by using CPLEX 8.1 and its callable library. Three different strategies with respect to use of valid inequalities were tested: using no valid inequalities at all (referred to as “novi”), using valid inequalities only at truncated branch and bound solving stages (so as to make the branch and bound hopefully faster, referred to as “vimip”) and using valid inequalities both at LP relaxation and MIP stages (referred to as “wvi”).

Four levels of accuracy for the department area representation were tested, which are 10%, 5%, 1% and 0.5%. Threshold value indicating the stopping condition of the LP based rounding was set to five different values, which are 0.75, 0.80, 0.85, 0.90 and 0.95. These settings were applied to both of the single and multiple fixing algorithms.

Results of the rounding heuristics and the one-step improvement are tabulated and provided at Appendices C and D, respectively. Our main observations on the results are as follows.

- At least around 40% of variables are fixed at the LP based rounding stage.
- Percent of fixed variables at the LP based rounding stage does not significantly differ among fixing singles and fixing multiples, although the number of iterations and the runtime of the rounding stage is higher for fixing singles.
- Percent of variables fixed decreases as the threshold value increases. Consequently, the percentage of improvement obtained by applying the improvement step decreases.
- Usually, improvement step obtains small improvements, up to 8% on the average, in short times.
- Infeasibilities encountered decreases as the threshold value increases. This is almost at the level of “being able to solve nothing at 0.75 but all at 0.95”.

- Solution quality increases as the threshold value decreases. However, good solutions at lower threshold values are only a few instances, since finding a feasible solution is the main issue for this region.
- Total runtime of the heuristic increases as the threshold value increases. This might be due to the fact that infeasibilities, which usually take less time to detect, are encountered more at small threshold values.
- When solving the LP without valid inequalities, adding the valid inequalities at truncated branch and bound stage decreases the computation time.
- There is no significant difference observed between fixing singles and fixing multiples with respect to the solution qualities and solution times.
- There is no significant difference observed between adding or not adding the valid inequalities at truncated branch and bound stage.

We also tried two quadratic assignment problem instances of sizes 15 and 25 by Nugent et al. (1968), with the same settings applied to other instances. It is seen that the proposed rounding heuristics perform very poorly for these instances. While feasible solutions cannot be found for low threshold values, high threshold values give solutions that vary between 51% and 109% worse than the optimal.

CHAPTER 4

EVOLUTIONARY ALGORITHM APPLICATIONS FOR THE FACILITY LAYOUT PROBLEM

4.1 Review of the Evolutionary Algorithm Applications on the Layout Problem

This section gives a brief overview on the evolutionary algorithm (EA) applications for the single period layout problem. These applications actually form a sub-category of the single period layout problem, on which an overview of the literature was given in Section 3.1. Since the problem is for a single period, especially determination of the aisles, I/O points and other detailed design issues find themselves a wide place within these applications, instead of determining only the block layout.

EAs are widely applied in almost all subject areas of industrial engineering. Dimopulos and Zalzala (2000), Aytug et al. (2003), Pieereval et al. (2003) and Chaudhry and Luo (2005) provide some recent reviews of EA applications in manufacturing. Although not very recent and hence excludes many recent papers, a survey of EA applications (together with simulated annealing (SA) applications) specifically for the facility problem is given by Mavridou and Pardalos (1997).

The aim of this review was to identify “good” EA applications and the features that make them good. A special attention was paid to identify the crossover and mutation operators that use the properties of the problem, instead of applying randomized operators such as uniform crossover. However, it was seen that this kind of “smart” operators were not employed at all. Besides, the use of different representations, which was explained in detail in Section 3.1, makes it almost impossible to adapt an operator from an algorithm to another.

Due to this dependency on the representation, the studies reviewed here are classified and discussed with respect to the representation scheme utilized. Here, studies that consider equal area departments are not reviewed in detail, since they deal with the quadratic assignment problem (QAP), which is a subject of extensive research on its own. Some examples of layout applications are Suresh et al. (1995), Mak et al. (1998), Tavakkoli-Moghaddain and Shayan (1998), Adel El-Baz (2004) and Wang and Okazaki (2005). A recent status report for QAP is given by Drezner et al. (2005). Also, an online QAP library containing commonly used problems for benchmarking, their best known solutions and by whom they are found is available at http://www.seas.upenn.edu/qaplib/, which is an online and up-to-date version of Burkard et al. (1997).

4.1.1 Studies That Use Grid Representation

Kochhar et al. (1998) develop an EA named HOPE, a detailed overview which is given in Table 4.1. The computational results provided are in three categories: QAP, unequal area single row problems and unequal area multi-row problems. Solutions to three problems of size 11 for unequal area multi-row problems are given, which are compared to and shown to outperform MULTIPLE by Bozer et al. (1994) and SABLE by Meller and Bozer (1996).

Kochhar and Heragu (1998) modify HOPE and name it MULTI-HOPE, which is designed to handle multiple floors. Computational results are provided for five problems of size 11, 21 and 40. The algorithm is shown to perform slightly better that MULTIPLE and SABLE, with the expense of a slight increase in the computation time.

In Gero and Kazakov (1998), the emphasis is on a methodology of selecting promising genes and evolving them, which is named “genetic engineering”. There are two example problems provided, one having equal and the other having unequal area departments. Similarly, Jo and Gero (1998) construct an EA and provide only one example (the unequal area department example in Gero and Kazakov (1998)), without further computational results.

Table 4.1 – An overview of Kochhar et al. (1998)

Element	**Feature**
Representation	Grid
Method overview	I/O point location decisions are integrated. GA structure similar to that of Tate and Smith (1995)
Representation	A string indicating the order of the departments over a pre-specified space-filling curve.
Handling infeasibility	Space-filling curve, adding bonus to feasible solutions
Initial population	Random
Crossover	Randomly pick N departments from the first parent and retain their values and positions. Fill in the blanks sticking to the sequence of the departments in the second parent. Apply the same by reversing parent roles to obtain the second offspring.
Mutation	Swap two departments
Fitness	C=material handling costs. Fitness is calculated using C, employing another function with parameter values of $\alpha = 0.5$ and $\beta = 1$. Also, the fitness values of feasible individuals are increased. N is given as the number of neighbor squares that belong to the same department. $k * C^{\alpha-1} * e^{-\beta C}(1 + N * \delta\%)$
Selection	Roulette wheel
Replacement	Elitist selection, copies top 10% of the best individuals to the next generation. If there is no improvement in the best solution for 100 consecutive generations, the rest of the generation (other than those top percent directly copied) is randomly generated.
Termination condition	4000 (20000) generations
Parameters	Population size=100 (20), % of solutions replaced by new generation=0.9, number of mutation rate (adaptive) = 0.1+0.8*(1-kt)1/t, where k is the ratio of the average population fitness to the best fitness in the current generations population and t changes from 4 to 1 as the run progresses.
Problem sizes solved	single row - 4, 10, 11, 20, 30; multi-row – 11, 15, 25

Hamamoto et al. (1999) provide a similar algorithm to the above, which is applied on a case study of small and medium sized facilities in pharmaceutical industry (having 15 and 26 departments respectively). Two types of space-filling curves (SFC) are tested, one with growing from the center and one with dividing the facility into bands. Solutions are manually modified so as to have a final layout with rectangular departments. The results are compared with human designers, CORELAP, CRAFT and BLOCPLAN. In general, the results are neither specifically good nor bad.

Lee and Kim (2000) consider an extension to the grid representation. They convert an irregular shaped layout that has been formed by using grid representation to a layout having all rectangular departments as shown in Figure 4.1.

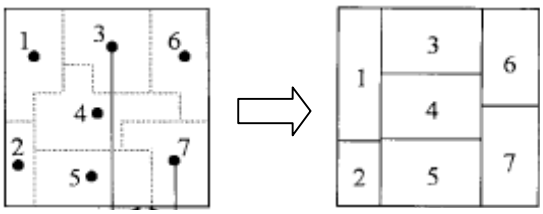


Figure 4.1 – Example of converting a layout constructed by using grid representation into a block layout (Lee and Kim, 2000)

Balakrishnan et al. (2003b) develop a software program called FACOPT that is able to use both SA and EA as computation tools. The SA and EA used are similar to SABLE and HOPE respectively. An overview of the employed EA is provided in Table 4.2. Computational results are given for parameter optimization and results of modified versions of 60 problems, with departments of sizes 10, 20 and 30. However, the authors only compare the performances of their SA and EA, showing that EA performs better with an expense of higher computation time. Further comparison with other algorithms is not provided.

Hu and Wang (2004) give some flexibility to the algorithm in defining the SFC by making the band sizes and the sweep directions decision variables. Computation is made for three problems of size 12, 14 and 20. The authors compare their results with PLANET and SHAPE and obtain better results, which are not very convincing in showing the goodness of the algorithm since the compared algorithms are neither recent nor competitive ones.

Table 4.2 – An overview of Balakrishnan et al. (2003b)

Element	**Feature**
Representation	Grid
Method overview	Main aim is to develop a software. Both a SA and GA algorithm are implemented.
Representation	A string indicating the order of the departments over a pre-specified space-filling curve.
Handling infeasibility	Space-filling curve, adding bonus to feasible solutions
Initial population	Random
Crossover	Randomly pick N departments from the first parent and retain their values and positions. Fill in the blanks sticking to the sequence of the departments in the second parent. Apply the same by reversing parent roles to obtain the second offspring.
Mutation	Swap two departments
Fitness	Total material handling costs.
Selection	Roulette wheel
Replacement	At each generation, a single offspring is produced and the worst parent is replaced by it.
Termination condition	4500 generations
Parameters	Population size=35, mutation rate = 0.09
Problem sizes solved	10, 20, 30

The paper by Wang et al. (2005) is almost the same as the one by Hu and Wang (2004). This time, they obtain results for three problems of size 8, 12 and 20.

4.1.2 Studies That Use Flexible-Bay Representation

Norman et al. (2001) measure the distance between department pairs considering the contours of the departments (similar to aisles) and meanwhile locating I/O points. A computational study is performed for several problems from the literature, with visual comparison of layouts. Details of the algorithm are given in Table 4.3.

Table 4.3 – An overview of Norman et al. (2001)

Element	**Feature**
Representation	Flexible bay
Method overview	I/O point location decisions are integrated. GA structure told to be similar to that of Tate and Smith (1995)
Representation	Variable length. "A permutation of departments that specifies their order within the layout, with a concatenated string indicating where the bay breaks within the permutation occur." Ex. G A F H B E K C L I J D **H K I**
Handling infeasibility	Repair (details not given)
Initial population	Not given
Crossover	"A variant of uniform crossover, where two parents create one offspring. The department sequence is subject to uniform crossover with repair to ensure feasible permutations. The bay structure is taken directly from one parent or another with equal probability"
Mutation	"Mutation consists of permutation altering, or adding, or deleting a bay. The permutation mutation is inversion between two randomly selected departments."
Fitness	Possible locations of I/O are determined and expressed as a network. Shortest path between each department pair is found and used as the distance metric while calculating the material handling costs.
Selection	"A rank-based quadratic method"
Replacement	Steady-state replacement
Termination condition	100,000 generations
Parameters	Population size=10, crossover rate=0.5, mutation rate=0.5 (altering=0.50, adding=0.25, deleting=0.25)
Problem sizes solved	10, 14, 20

Ozdemir et al. (2003) use different distance metrics for measuring the distances between department pairs, which is reasonable when the material handling devices are different. The algorithm can also handle a single distance metric, too. Computational results are given for 6 different parameter settings for each of the two problems of size 10 and 20. Comparative results are not given, due to this kind of study being made for the first time. Details of the algorithm are given in Table 4.4.

Gómez et al. (2003) and Aiello et al. (2002) provide a few computational results to be able to come up with conclusions. Gómez et al. (2003) use the adjacency relationship matrix (AEIOUX) for calculating the objective function. They test their algorithm on a set of problems with 20 departments. Aiello et al. (2002) determine the block layout and locate the I/O points simultaneously. They state that their

algorithm is one of its kind, so they just demonstrate it on a randomly generated 20 department problem.

Table 4.4 – An overview of Ozdemir et al. (2003)

Element	**Feature**
Representation	Flexible bay
Method overview	Different distance metrics for each department pair is used.
Representation	Similar to that of Norman et al. (2001) Ex. G A F H B E K C L I J D | 4 7 11
Handling infeasibility	Repair in crossover, penalty in fitness function. Two types of penalties are used: 1. Violation of aspect ratio 2. If the distance between two departments is measured by Tchebychev distance, then they should be located over the same axis in order for them to have an overhead crane installed amongst.
Initial population	"Randomly generated with $\sqrt{n}$ bays on average" (n: # of departments)
Crossover	"A variant of uniform crossover, where two parents create one offspring." Similar to uniform crossover, same departments in same locations in both parents are directly taken and "conflicts are then resolved to ensure that each departments occurs exactly once in the child"
Mutation	"Mutation consists of permutation altering, or adding, or deleting a bay. The permutation mutation is inversion between two randomly selected departments. For the bay mutation there is one that splits an existing bay into two adjacent bays and one that merges two adjacent bays into one."
Fitness	Possible locations of I/O are determined and expressed as a network. Shortest path between each department pair is found and used as the distance metric while calculating the material handling costs.
Selection	"A uniform random number between 1 and $\sqrt{p}$ is chosen (p: population size), then squared. The result is truncated and then taken to be the rank of the parent to be selected (where string zero is the best string in the population)"
Replacement	"Children and parents are pooled and the best p are retained." "A maximum of 80% of the population can be replaced by mutated solutions" "Generated solutions are not guaranteed to be unique."
Termination condition	When the best feasible solution found does not change for 20,000 generations
Parameters	Population size=200, crossover rate=0.5, mutation rate=0.5 (altering=0.50, adding=0.25, deleting=0.25)
Problem sizes solved	10, 20

Flexible-bay structure is also used when the width and length of the departments are fixed and known. This situation is usually faced when locating machinery on FMS lines. Some applications of this type are Lee and Lee (2002), Ficko et al. (2004) and Hicks (2004). Lee and Lee (2002) consider placing the departments within bays one by one, whose width are determined by the department placed on the bay that has the largest width. The algorithm used is a hybrid of simulated annealing, tabu search and evolutionary algorithms.

The approach of Ficko et al. (2004) is almost the same as the approach of Lee and Lee (2002). A difference is that they also consider aisle widths and measure the distance between two departments by the deterministic I/O points they locate at each department's mid-bottom. They provide a single 14 department example. The study by Hicks (2004) is also similar to the one by Lee and Lee (2002), where a case study based on real data is provided.

4.1.3 Studies That Use Slicing Tree Representation

First of all, although the details are not given here, it should be noted that the operators for this representation are usually more complicated than those for other representations, since the offspring should also be a slicing tree.

Tam (1992) was the first to introduce the problem into the layout literature. Details of the algorithm developed in this study are given in Table 4.5.

Wu and Appleton (2002) consider departments of known width, length and I/O points, where orientation of them is to be decided. For coding, slices are converted into aisles and a chromosome is formed by three strings: 1. Facilities string, 2. Cutting levels string that represents the aisles, 3. Orientation string. The method is demonstrated over an example, which is then compared with a random solution.

Table 4.5 – An overview of Tam (1992)

Element	**Feature**
Representation	Slicing tree (with 4 operators)
Method overview	Aim is to show that GA can be used (and is actually a good method to use) in facility layout problems
Representation	Structure of the tree is fixed, together with the information of which department is assigned to which node. Therefore, representation is postorder sequence with only "operator"s.
Handling infeasibility	Penalizing aspect ratio violations and "dead space ratio"s (ratio of the area of total occupied spaces –fixed departments– in the partition allocated to a facility to the area of the partition allocated to that facility)
Initial population	Determination of the tree structure: A distance measure, Aij, is constructed using the traffic information, using the formula Aij=1/(1+total flow between the pair). The values of the Aij matrix "is input to a numerical clustering procedure to generate a dendogram. The dendogram depicts the hierarchical process of clusters merging and can be used to provide the structure of a slicing tree." For clustering, average linkage method is used. Initial solutions: Randomly generated for fair comparison
Crossover	Single-point crossover
Mutation	Randomly changing a single operator in the string
Fitness	Total material handling costs plus the penalties. Linear scaling is used with a scaling factor of 1.8 to avoid premature convergence.
Selection	Roulette wheel
Replacement	"A generation gap of 1 is used"
Termination condition	150 generations
Parameters	population size=30, crossover rate=0.95, mutation rate=0.01
Problem sizes solved	12, 15, 20, 30

Azadivar and Wang (2000) also consider departments of known width and length. The algorithm is designed to use different objective functions than minimizing the total material handling cost, which are calculated by means of simulation. The paper only has a numerical example, which takes the objective function as minimizing the average cycle time for all parts.

Tam and Chan (1998) consider variable department sizes under area and aspect ratio constraints. Their study is mainly focused on making use of the parallel GA concept.

Gau and Meller (1999) add dummy departments to the slicing tree in order to enlarge the neighborhood and increase the exchange opportunities. A penalizing scheme is used for aspect ratio limit violations. As a result of the experimentation, there is

improvement observed on previous work. Details of their work are given in Table 4.6.

Table 4.6 – An overview of Gau and Meller (1999)

Element	**Feature**
Representation	Slicing tree (with 4 operators)
Method overview	Relative positions of departments in the layout (binary variables) are determined by evolving a slicing tree represented population. Then, a percent of these binary variables are fixed and a reduced MIP is solved. Then the result of the MIP is converted into a slicing tree and fed into the GA. Three GAs are developed. BASE-GA (a single tree structure is used), MULT-GA (different populations are evolved for a number of tree structures) and EXT-GA (the tree is expanded by adding dummy departments to each possible place)
Handling infeasibility	Penalizing solutions that violate aspect ratio constraints (Adapted from Coit et al., 1996)
Initial population	Random
Crossover	Perform a leaf-for-branch or a branch-for-branch exchange with equal probability
Mutation	"Do the simple crossover on department cuts, then randomly change one of the cuts or exchange a pair of the departments"
Fitness	Total material handling cost minus a penalty if aspect ratio is violated
Selection	Roulette wheel
Replacement	Top k percent of the population is directly transferred to the next generation. c percent is generated with crossover and the remaining m percent is generated with mutation (k+c+m=1)
Termination condition	500 generations
Parameters	population size=500, k=0.02, c=0.96, m=0.02, number of trees for MULT-GA=5
Problem sizes solved	10, 11, 12, 14, 15, 20, 25, 30

Some other applications of this structure can be found in Cohoon et al. (1991), Schnecke and Vornberger (1997), Al-Hakim (2000) and Shayan and Chittilappilly (2004).

4.1.4 Algorithms with Other Representation Types

Rajasekharan et al. (1998) do not pose any limits on the relative positionings of the departments. They present a two step EA, which first determines the relative

positions of the centroids of the departments, then determines on which axis the departments should be separated. Fitness of a solution is calculated by a branch and bound procedure, which is very easy to solve due to the fact that all integer variables are fixed. Details of the algorithm are given in Table 4.7.

Delmaire et al. (1997) work on layouts with given "spines", i.e. aisle structures. The fitness of a solution is calculated by solving an LP. Problems solved are of size 12, 22 and 32.

Table 4.7 – An overview of Rajasekharan et al. (1998)

Element	**Feature**
Category	Rectangular departments, no constraints
Representation	Binary string with three components: α_{ij}, β_{ij} and θ_{ij}. α_{ij} = 1 (0) if department i is to the right(left) of department j. β_{ij} = 1(0) if department i is above(below) department j. θ_{ij} = 1 (0) if departments i and j do not overlap in vertical(horizontal) direction.
Method overview	Step 1. For determining α and β, produce offspring as much as the population size. Select the one with the best fitness value among them. Step 2. Given the "good" α and β found in step 1, evolve only for determining θ.
Handling infeasibility	Discard infeasible individuals
Initial population	Random (for step 1)
Crossover	Single point
Mutation	Generate a number, M. Change M random bit values.
Fitness	Determined by branch and bound, easy since all α, β and θ are given for a solution.
Selection	An individual is accepted as parent if its fitness is above population average. It is accepted with a certain probability if its fitness is below population average.
Replacement	Steady-state reproduction without duplicates (generate a percent of the population by reproduction, let them into the population by discarding the solutions with worst fitness values)
Termination condition	Either when the change in the average fitness population is less than 1% or the best population in the solution has not changes for 10 subsequent generations.
Parameters	population size=50, crossover rate=0.8, mutation rate=0.1, percentage of solutions replaced by new generation=0.9, probability of accepting an individual with fitness value less than average as a parent=0.2
Problem sizes solved	4, 6, 8, 10, 12, 14, 16, 18

4.2 Development of an EA Scheme for FLP

Using the sequence pair representation to determine the relative positioning of departments is found especially worthwhile to analyze, since it is a new representation scheme that has a promising structure but is not explored by all means yet. The only layout application that uses the sequence pair representation is Kim and Goetschalckx (2005) employ an SA, where the layout, aisle structures and I/O points are determined simultaneously. In addition, there is a lack of applications in the literature that do not impose certain structures for departments. Therefore, this section describes the proposed EA scheme for the facility layout problem, using the sequence pair representation.

As discussed in Section 3.2, a sequence pair is described as "a pair of entity sequences that can be used to determine the relative location of entities in the two-dimensional space." (Meller et al., 2006). An example for a sequence pair for a layout problem of 6 departments is ((1 6 2 5 3 4), (4 5 1 6 2 3)). The relative positions of the departments in the layout are determined by the relative positions of the numbers that represent the departments in each sequence. For example, if department A comes before department B in both of the sequences, then the right boundary of department A should come before the left boundary of department B when the layout is projected onto the x-axis. In other words, the departments should be separated on the x-axis. In the example sequence pair, departments 6 and 3 should be separated on the x-axis, where 6 is to the left of 3, whereas departments 1 and 4 should be separated on the y-axis, where 1 is above 4.

SP-ε, given in Section 3.3, is an MIP model due to the presence of binary variables that define the relative positions of departments. When the values for these variables are specified, the problem becomes an LP, which can be solved in a very short time. This model is given below, where L and B are sets with the following definitions.

$L = \{(i, j) \mid$ department i should be placed to the left of department $j\}$
$B = \{(i, j) \mid$ department i should be placed below department $j\}$

LP-ε:

$$\text{Min} \sum_{\forall i} \sum_{\forall j>i} f_{ij}\left(d_{ij}^{x} + d_{ij}^{y}\right) \tag{4.1}$$

subject to

$$d_{ij}^{s} = \left| c_i^s - c_j^s \right| \qquad \forall\ i < j; \forall\ s \tag{4.2}$$

$$l_i^s \le c_i^s \le L^s\text{-}l_i^s \qquad \forall\ i; \forall\ s \tag{4.3}$$

$$lb_i^s \le 2l_i^s \le ub_i^s \qquad \forall\ i; \forall\ s \tag{4.4}$$

$$c_i^x + l_i^x \le c_j^x - l_j^x \qquad \forall\ (i, j) \in L \tag{4.5}$$

$$c_i^y + l_i^y \le c_j^y - l_j^y \qquad \forall\ (i, j) \in B \tag{4.6}$$

$$l_i^y + \frac{a_i}{\left(\bar{l}_{ik}\right)^2} l_i^x \ge \frac{a_i}{\bar{l}_{ik}} \qquad \forall\ i;\ k = 1,\ldots,N_i \tag{4.7}$$

$$c_i^s, l_i^s, d_{ij}^s \ge 0 \qquad \forall\ i; \forall\ j; \forall\ s \tag{4.8}$$

Infeasibility Issues

An LP model for which a sequence pair, i.e., the relative positions of the departments, is given can be infeasible if the given sequence pair imposes a layout that extends beyond the floor area due to some departments violating the facility width and/or length boundaries. In order to find out how often the type of infeasibility described above might be encountered, 100 random sequence pairs were generated and LP-ε was solved for the problem instances on hand (except m3 and m4 instances since the number of all possible solutions for them are very small, being 36 and 144, respectively). Table 4.8 reports the number of sequence pairs among the randomly generated 100, which result in a feasible layout. It is seen that the percentage of feasible solutions is less than 10% of the random solution population for 21 out of 56 test instances, on the average being around 20% on all test problem instances.

One might employ one of the several constraint handling techniques in EAs to overcome this situation. Although the easiest method is simply rejecting infeasible solutions, the infeasible solutions are decided to be kept in the population due to the fact that the feasible region can turn out to be very small. Here, it should be noted

that the decision of keeping the infeasible solutions in the population requires addition of some mechanisms to evolve them into feasible solutions.

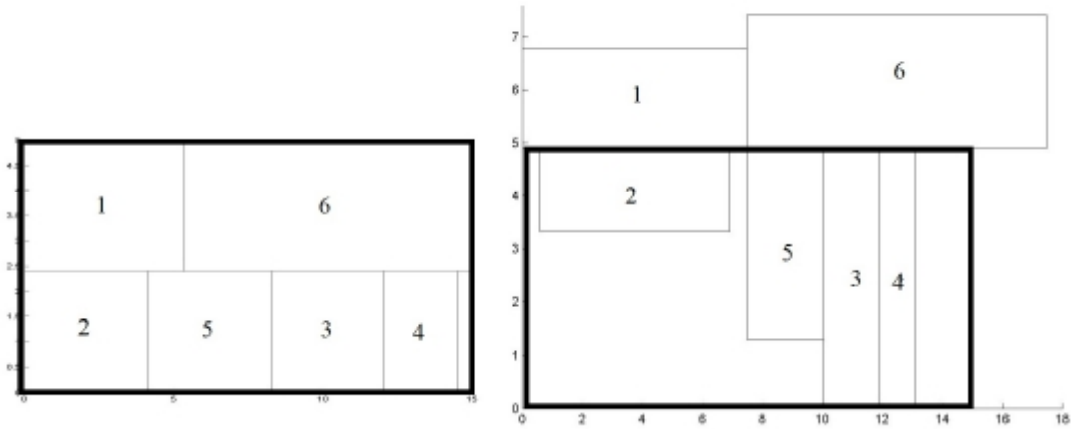
Table 4.8 – Number of feasible solutions out of 100 randomly generated instances

		P														
		m5	**m6**	**m7**	**o7**	**o8**	**o9**	**o10**	**Fo7**	**fo8**	**fo9**	**fo10**	**fo11**	**vc10**	**ab20**	**AVG**
ε	**10%**	1	10	3	18	16	12	16	15	24	10	14	99	88	7	23.79
	5%	2	12	4	12	15	11	6	6	17	16	5	96	85	3	20.71
	1%	1	10	1	11	12	6	6	14	12	4	6	95	76	3	18.36
	0.5%	1	9	6	11	15	9	6	17	10	10	11	96	70	0	19.36

To keep infeasible solutions in the population, the constraint set that causes the infeasibility, namely (4.3) that forces the departments to be located within facility boundaries, should be removed from the model while solving the LP. However, it is observed that the complete removal of these constraints makes potential feasible solutions to be evaluated “infeasible” by the model. This situation occurs because lower material handling costs for the sequence pair imposed can be obtained by changing the aspect ratios of the departments so that facility boundaries are violated. An example for this situation is illustrated in Figure 4.2. In the figure, two layouts generated using the same sequence pair, ((1 6 2 5 3 4), (2 5 3 4 1 6)), are shown, where the dark rectangles show the facility boundaries, layout (a) is obtained by keeping constraint set (4.3) and layout (b) is obtained by removing it. In layout (b), the half lengths of departments 3, 4 and 5 are reduced, hence, the distance between them is reduced, too.

To overcome the situation described above, each LP is first solved by removing the constraint set (4.3). If the solution found violates the facility boundaries, constraint set (4.3) is added to the model and the model is resolved. If a feasible solution is found, the solution found by solving the LP for the first time is overwritten by this

feasible solution found. Otherwise, the solution violating the boundaries, which had been obtained by solving the first LP, is kept. By this way, it is ensured that the layout corresponding to a sequence pair that can achieve to stay within the boundaries of the facility is forced to do so, as opposed to being evaluated as an infeasible layout with smaller material handling cost. Note that, the property of CPLEX to start from the final basis of the previously solved problem is utilized here to reduce the time spent in the step of resolving the LP with additional constraints.



(a) (b)

Figure 4.2 – Infeasibility of feasible problems caused by removal of constraints

4.3 Settings for the Proposed EA Scheme

4.3.1 Coding, Fitness Function and Selection and Replacement Schemes

As explained before, the sequence pair representation is to be used. Therefore, each solution in the population will be represented by a string of size 2*number of departments, which can be regarded as the “genotype” of the solution. The “phenotype” of the solution, namely, the centroid coordinates and the length and width of each department, is found by solving LP-ε as described in Section 4.2. If

the length, width and orientation of all departments are known, the phenotype can be found in O(n) time, as shown by Kodama and Fujiyoshi (2003). But since the length and width of the departments are decision variables, one needs to solve an LP.

The basic component of the fitness of a solution is the total material handling cost of the layout it corresponds to. Before determining the complete fitness function to be used, an important distinction should be made to determine how to handle infeasible solutions within the population and direct the population to evolve into a set of feasible solutions. Two methods are proposed for the answer of this question, which are explained in detail below.

Selection strategy for both methods is chosen as the ranking based selection proposed by Reeves in 1995 (see Demir, 2004). In this method, all solutions in the population are ranked according to some criterion and a solution with rank i among a population of n solutions is chosen with probability p_i as given by equation (4.9).

$$p_i = \frac{(n+1)-i}{\sum_{k=1}^{n} k} \tag{4.9}$$

A. Penalize the infeasible solutions and replace the worst(s)

In this approach, a penalty function is calculated and added to the material handling cost of each solution. The penalty function proposed, which is given below in equation (4.10), is adapted from Coit et al. (1996).

$$Cp_t = C_t + \left(C_{feas} - C_{all}\right)\left(\frac{n_t}{NFT}\right)^k \tag{4.10}$$

Cp_t is the penalized objective function value of solution t, C_t is the unpenalized objective function value of solution t, C_{feas} is the value of the best feasible solution yet found and C_{all} is the unpenalized value of the best solution yet found, n_t is the number of departments violating the facility boundaries in solution t. NFT is the

threshold distance to a feasible solution, which is taken as 1 and k is the severity parameter, which is taken as 2 (similar to Coit et al., 1996).

After an operator is applied, a steady-state replacement strategy is employed, where the offspring replace(s) the worst solution(s) in the population (unless its/their penalized fitness value is greater than that of the worst solution).

B. Do not penalize the infeasible solutions and replace the parent(s)

In this approach, fitness of each solution is simply the material handling cost under it. Feasible solutions are encouraged in the population by making them always dominate infeasible solutions in the population. Ties among same solution types, i.e., two feasible or two infeasible solutions, are broken by the material handling costs.

Replacement is designed to be made within a pool of parents and children, similar to Demir (2004). Among parents and children, the better solutions are selected to be kept in the population, within a steady-state replacement scheme. So, here, offspring replace parents if they are better, instead of the worst solutions in the population.

4.3.2 Reproduction

In order to come up with a crossover operator, one should first find how to combine two layouts so that their good properties are preserved. We concluded that this is a difficult question to answer, independent of any representation scheme. However, we propose a simple crossover operator developed for the sequence pair representation, which can be tested to see whether it produces “good” offspring.

Also, two mutation operators are proposed, which can be considered as improved versions of the neighborhood moves used in Kim and Goetschalckx (2005). These moves are designed within an enumerating scheme and some heuristic properties are added, which are expected to direct the algorithm towards feasibility.

A. Crossover operator

This operator uses two parents to produce two offspring. The second sequences of the parents are exchanged, resulting in two new solutions, which is illustrated in Figure 4.3.

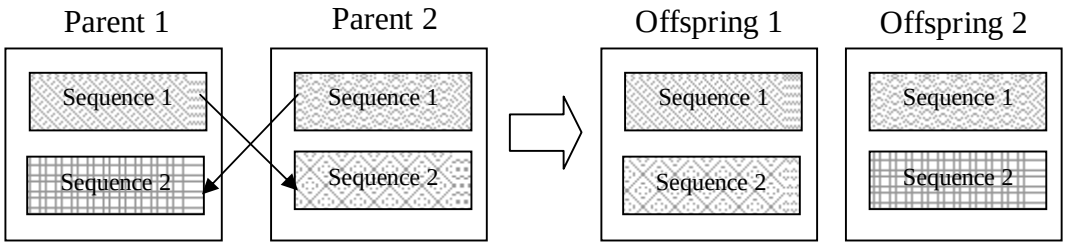


Figure 4.3 – Illustration of the proposed crossover operator

B. Mutation operator based on exchange

A remark here is that exchanging the same two departments in both of the sequences corresponds to actually swapping the departments, like a CRAFT move. However, it is preferred to consider more possibilities and develop an enumeration like operator.

In this mutation, firstly a department to be exchanged is chosen. Then, all possible exchanges for this department with other departments are enumerated in both of the sequences. Then, all possible combinations of these newly generated sequences are enumerated and the best alternative generated among them is chosen as the offspring. This operator evaluates (n-1) alternative layouts for each sequence and hence $(n\text{-}1)^2$ alternative layouts for each mutation operation

The first department for which enumeration will be made is chosen using either of the two strategies:

1. *Smart exchange mutation:* If the layout to be mutated is an infeasible one, one of the departments to be exchanged is chosen randomly among the

departments that violate either of the width or length boundary constraints. If the layout to be mutated is feasible, a department is selected randomly.

2. *Random exchange mutation:* The department to be mutated is chosen randomly, regardless of other conditions.

C. Mutation operator based on adjacency

Two departments can be made adjacent in the layout by putting them in adjacent places in both of the sequences. For each sequence, one of the departments is selected as "to be moved" and the other is considered "fixed". Moved department can either be placed to the left or to the right of the fixed department. Therefore, there are four possible ways to make two departments adjacent in one sequence. Similar to the exchange mutation, all of these 4 x 4 = 16 alternatives are generated and the best solution among them is chosen as the offspring.

The departments to be made adjacent are chosen probabilistically, where the probability of selecting a department pair is directly proportional to the flow amount between these departments. If the departments are already adjacent in one sequence, then only 4 alternatives are evaluated; if they are already adjacent in both sequences, then a new pair of departments is chosen for evaluation until a successful mutation is achieved.

Initial Population and Stopping Criteria

The initial population is proposed to be generated randomly in order to be able to assess the improvement achieved by the algorithm truly. Different stopping criteria can be employed, such as terminating after a certain number of iterations or when the population average does not improve for some time. The details are left to be determined by initial runs, where the convergence can be observed.

The flowchart of the proposed algorithm is given in Figure 4.4.

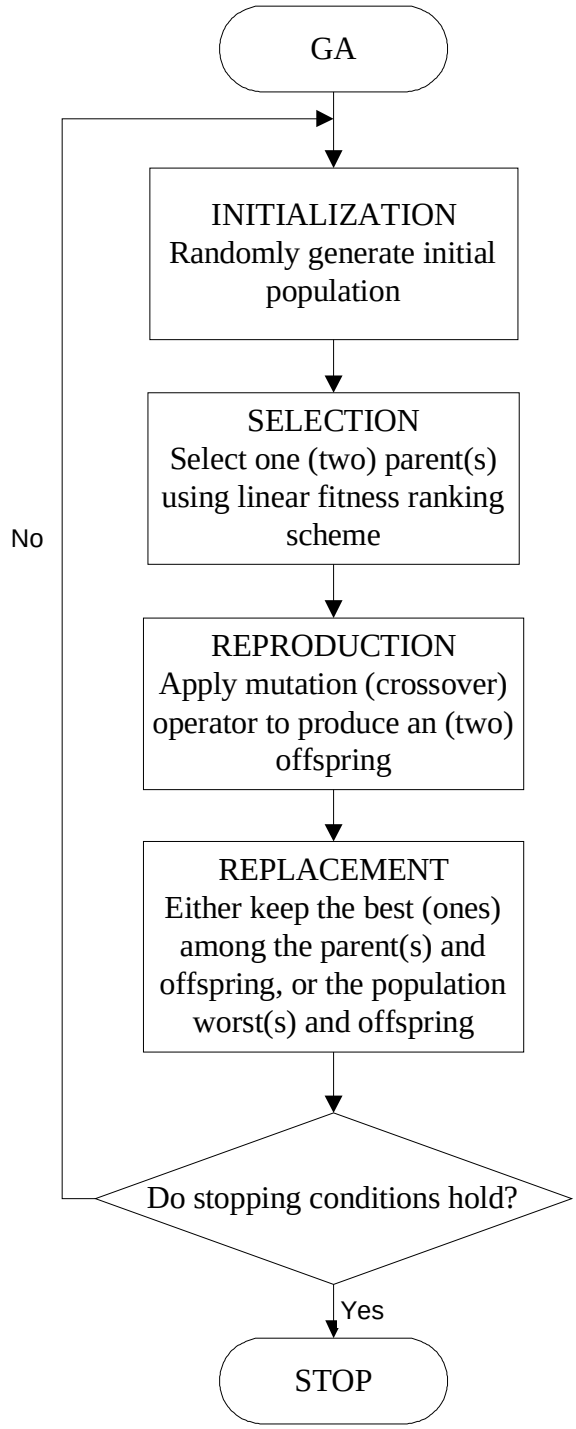


Figure 4.4 – The flowchart of the proposed EA scheme

4.4 Preliminary Computational Results

Each of the described mutation operators is implemented in an evolution-like routine to test their efficiency. In the routine, the option of penalizing the infeasible solutions and replacing the population worst solutions is implemented. The algorithm where the replacement scheme is to replace the parent is also developed, but not tested. The detailed steps of this algorithm are given in Appendix E.

A population size of 50 solutions is chosen. If there is no feasible solution in the initial population, an upper bound value is calculated to use in place of C_{feas}, by assuming that all departments are placed on opposite corners of the facility and both the length and width of the departments are at their lower bounds (hence, they travel the maximum possible distance). The upper bound values for the problems and their gaps with respect to the optimal solution values and LP relaxation solution values are given in Table 4.9. Algorithm is terminated when either 1000 generations are completed, or the average fitness value of the feasible solutions in the population does not change more than 0.1% for the last 100 iterations.

The same problem sets as those in Section 3.5 are used for the computational study. All computational settings are the same as those in Sections 3.5 and 3.6. Here, problem instance m3 was not tested since it has less alternative solutions (3!*3! = 36) than the population size. m4 was not tested for mutation based on adjacency, again due to being a small instance. All other test problems except ab20 were tested for two epsilon values, 1% and 10%. Results are given in Tables 4.10-4.12. Each row in the table is the average of results obtained by making 10 replications. Replications where the initial population does not have a feasible solution are not included in the averages reported in columns %F0, BFD0, AFD0, IMPB and IMPA. Similarly, replications where the final population does not have a feasible solution are not included in the averages reported in columns %FF, BFDF, AFDF.

Graphs given in Figures 4.5-4.6 illustrate the deviation of the best feasible solution from the optimal solution and the improvement obtained in the best feasible solution for all instances. Observations regarding the results are as follows.

- The population successfully evolves from infeasibility towards feasibility. On the average, the percentage of feasible solutions in the population increase from around 15% to 80%. The only exception for this pattern is fo11, whose feasible region is very large, as seen from Table 4.8.
- Qualities of solutions worsen and solution times increase as the problem size increases. This pattern is especially observed among the problems that belong to the same family, i.e., m*x*, o*x*, fo*x*. This can be attributed to the fact that the same population size is used for all instances. Hence, a smaller percentage of all possible solutions are represented in the population while solving larger problems.
- Mutation based on adjacency performs slightly worse than mutations based on exchange.
- There is no significant difference between mutations based on random and smart exchange. This is due to the fact that smart exchange is equivalent to random exchange when the parent is feasible. Since a high level of feasibility is achieved towards the end of the algorithm (around 80% on average), smart exchange behaves similar to random exchange, overall.
- m*x* and fo*x* instances use the iteration limit of 1000 as much as possible. Whereas, o*x* instances terminate around 580 iterations. In addition, improvements obtained in the deviation of the best feasible solution are smaller as compared to those obtained for m*x* and fo*x* instances. Identifying the properties of the parameters in these instances and the reasons why the operators perform good or bad might be worth to analyze in order to understand the problem better and develop new operators accordingly.

Table 4.9 – Deviation of the upper bound from optimal and LP relaxation solution values

P	UB	D Type	ε			
			0.5%	**1.0%**	**5.0%**	**10.0%**
m3	284.40	LPRELD	1895%	1895%	1895%	1895%
		OPTD	652%	652%	652%	652%
m4	504.53	LPRELD	∞	∞	∞	∞
		OPTD	888%	888%	888%	888%
m5	507.88	LPRELD	∞	∞	∞	∞
		OPTD	723%	724%	726%	732%
m6	1131.49	LPRELD	4963%	4963%	4963%	4963%
		OPTD	1276%	1276%	1276%	1277%
m7	1756.49	LPRELD	7760%	7760%	7760%	7760%
		OPTD	1545%	1545%	1545%	1545%
o7	928.42	LPRELD	6089%	6089%	6089%	6089%
		OPTD	610%	635%	611%	668%
o8	1806.42	LPRELD	6469%	6469%	6469%	6469%
		OPTD	644%	646%	655%	658%
o9	1863.00	LPRELD	6675%	6675%	6675%	6675%
		OPTD	690%	691%	696%	747%
o10	2017.20	LPRELD	7235%	7235%	7235%	7235%
		OPTD	NA	NA	NA	NA
fo7	234.98	LPRELD	9299%	9299%	9299%	9299%
		OPTD	1034%	1045%	1052%	1101%
fo8	310.84	LPRELD	10261%	10261%	10261%	10261%
		OPTD	1295%	1297%	1301%	1301%
fo9	367.50	LPRELD	12150%	12150%	12150%	12150%
		OPTD	1466%	1477%	1489%	1504%
fo10	427.10	LPRELD	14137%	14137%	14137%	14137%
		OPTD	1581%	NA	NA	1636%
fo11	553.50	LPRELD	18350%	18350%	18350%	18350%
		OPTD	NA	2081%	NA	2082%
vc10	320592.12	LPRELD	14765%	14765%	14765%	14765%
		OPTD	NA	NA	NA	NA
ab20	3578.07	LPRELD	39530%	39530%	39530%	39530%
		OPTD	NA	NA	NA	NA

P Problem instance
ε Accuracy level of the area representation
UB Upper bound value
D Type Type of deviation reported
LPRELD Deviation of the UB from LP relaxation solution value (UB – LPREL) / LPREL
OPTD Deviation of the UB from optimal solution value (UB – OPT) / OPT

Table 4.10 – Results of mutation based on smart exchange for instances whose optimal solutions are known

P	ε	#UB	#INF	#ITER	%F0	%FF	BFD0	BFDF	AFD0	AFDF	CPU	IMPB	IMPA
m4	**0.01**	1	0	875.2	2.6	78.8	25.8%	5.6%	33.6%	15.7%	70.78	21.6%	18.5%
	0.1	2	0	845.1	4	86.2	51.3%	0.0%	62.6%	10.6%	64.33	51.3%	51.6%
m5	**0.01**	9	0	931.4	0.2	34.6	22.0%	8.2%	22.0%	15.9%	150.58	18.7%	7.9%
	0.1	9	0	912.0	0.2	47	53.2%	14.8%	53.2%	23.8%	147.37	30.0%	15.8%
m6	**0.01**	0	0	903.8	8.6	68.8	98.2%	21.8%	154.0%	42.5%	308.78	76.3%	111.5%
	0.1	0	0	897.4	11.8	78.6	81.3%	24.4%	140.9%	42.9%	278.00	56.9%	98.0%
m7	**0.01**	4	0	990.7	2.2	97	217.5%	45.4%	239.5%	90.8%	541.20	171.3%	149.9%
	0.1	1	0	980.7	3.8	96.8	163.9%	48.4%	209.7%	80.7%	486.78	115.7%	129.0%
fo7	**0.01**	0	0	886.1	13.6	89.8	58.7%	13.8%	103.1%	32.6%	477.00	44.9%	70.5%
	0.1	0	0	841.8	21.8	93.2	49.1%	18.9%	102.0%	32.8%	429.12	30.2%	69.1%
fo8	**0.01**	0	0	905.2	10	86.2	100.8%	43.1%	145.0%	65.4%	734.66	57.7%	79.6%
	0.1	0	0	896.3	17.2	91.2	94.7%	37.8%	145.6%	57.5%	641.53	56.9%	88.1%
fo9	**0.01**	0	0	947.2	11.6	89.6	132.6%	57.2%	169.1%	80.1%	1135.08	75.4%	89.0%
	0.1	0	0	923.6	18.6	95.8	122.4%	61.1%	183.3%	83.0%	973.63	61.3%	100.3%
fo10	**0.1**	0	0	910.7	12.8	93.6	127.2%	72.1%	202.0%	102.4%	1353.80	55.1%	99.7%
fo11	**0.01**	0	0	862.6	92.8	84.2	112.0%	67.2%	214.4%	85.4%	1903.79	44.8%	129.0%
	0.1	0	0	969.0	93.6	82	114.8%	62.9%	207.8%	79.1%	1885.73	51.9%	128.7%
o7	**0.01**	0	0	594.8	13.8	77.8	18.2%	6.0%	35.1%	10.9%	340.92	12.2%	24.2%
	0.1	0	0	598.3	23	74.6	21.3%	6.2%	42.5%	11.9%	309.09	15.1%	30.6%
o8	**0.01**	0	0	586.7	9.2	85	29.2%	9.5%	41.2%	16.1%	514.39	19.7%	25.2%
	0.1	0	0	557.7	16.2	77.6	22.7%	8.0%	37.7%	14.1%	444.97	14.8%	23.6%
o9	**0.01**	0	0	598.2	9.6	97	36.9%	13.2%	48.6%	21.0%	739.70	23.7%	27.6%
	0.1	0	0	460.3	18.2	94.6	34.6%	18.8%	53.0%	26.3%	514.80	15.8%	26.7%
AVG	**0.01**	1.3	0.0	825.6	15.84	80.80	77.4%	26.5%	109.6%	43.3%	628.81	51.5%	66.6%
	0.10	1.0	0.0	816.1	20.10	84.27	78.0%	31.1%	120.0%	47.1%	627.43	46.2%	71.8%

P	Problem instance
ε	Accuracy level of the area linearization
#UB	# of replications where no feasible soln. exists at initial pop.
#INF	# of replications where algorithm could not find a feasible solution
#ITER	Number of iterations algorithm terminates
suffix 0	… at initial population
suffix F	… at final population
%F	Percent of feasible solutions
BFD	Deviation of best feasible solution from optimal
AFD	Dev. of avg. of feas solns. in pop. from optimal.
CPU	Runtime of algorithm in seconds
IMPB	Impr. obtained in best solution wrt initial pop.
IMPA	Impr. obtained in pop. feas. avg. wrt initial pop.

Table 4.11 – Results of mutation based on random exchange for instances whose optimal solutions are known

P	ε	#UB	#INF	#ITER	%F0	%FF	BFD0	BFDF	AFD0	AFDF	CPU	IMPB	IMPA
m4	**0.01**	1	0	699.1	2.6	94	27.1%	5.6%	38.6%	15.9%	51.96	20.8%	22.5%
	0.1	2	0	895.0	4.2	90.2	42.7%	0.0%	60.5%	4.7%	61.90	42.7%	56.5%
m5	**0.01**	8	0	923.5	0.4	42.2	45.4%	8.6%	45.4%	14.1%	142.68	43.7%	38.7%
	0.1	8	0	827.2	0.4	41.4	36.0%	13.7%	36.0%	18.5%	120.39	15.0%	6.4%
m6	**0.01**	0	0	967.4	7.8	62.2	107.2%	22.5%	159.0%	35.1%	304.63	84.7%	123.8%
	0.1	0	0	949.3	12.2	71.4	77.7%	22.9%	140.7%	34.6%	272.91	54.8%	106.2%
m7	**0.01**	2	0	1000.0	2.2	97	182.5%	45.4%	190.5%	76.0%	504.87	138.5%	119.0%
	0.1	0	0	984.6	3	98.8	197.6%	46.1%	219.5%	68.9%	450.17	151.5%	150.5%
fo7	**0.01**	0	0	897.8	13.8	89	54.8%	13.5%	103.6%	25.6%	457.17	41.4%	77.9%
	0.1	0	0	951.5	21.6	78.2	52.5%	18.5%	103.8%	26.2%	446.49	33.9%	77.7%
fo8	**0.01**	0	0	922.0	9.6	81.4	102.3%	41.9%	145.0%	59.3%	697.47	60.3%	85.7%
	0.1	0	0	896.6	17	95.4	96.2%	36.6%	145.5%	51.3%	610.59	59.7%	94.2%
fo9	**0.01**	0	0	984.5	11.2	92.2	128.6%	57.0%	169.7%	74.7%	1115.95	71.6%	95.1%
	0.1	0	0	857.6	18.4	94	125.2%	55.7%	185.8%	78.8%	853.21	69.5%	107.0%
fo10	**0.1**	0	0	855.8	12.8	95.8	137.4%	70.6%	205.5%	97.4%	1192.65	66.8%	108.1%
fo11	**0.01**	0	0	958.0	91.8	85	108.5%	62.1%	215.0%	79.5%	2012.65	46.4%	135.5%
	0.1	0	0	890.7	94.2	75.2	106.2%	60.2%	207.2%	74.6%	1651.55	46.1%	132.6%
o7	**0.01**	0	0	626.5	13.6	80.6	18.2%	5.7%	35.1%	9.7%	329.17	12.5%	25.4%
	0.1	0	0	672.7	22.8	67.4	19.7%	5.5%	42.6%	10.0%	323.11	14.3%	32.6%
o8	**0.01**	0	0	623.9	9.6	84.8	29.5%	9.1%	40.7%	14.7%	507.28	20.4%	26.0%
	0.1	0	0	579.8	16.6	81.8	21.5%	7.6%	37.5%	13.2%	425.54	13.9%	24.3%
o9	**0.01**	0	0	723.1	9.6	99.2	33.7%	13.1%	46.9%	18.8%	832.28	20.6%	28.1%
	0.1	0	0	544.6	18.2	91.8	29.8%	18.4%	52.2%	25.0%	559.19	11.4%	27.2%
AVG	**0.01**	1.0	0.0	847.8	15.65	82.35	76.2%	25.8%	108.1%	38.4%	632.34	51.0%	70.7%
	0.10	0.8	0.0	826.0	20.05	82.15	78.6%	30.0%	119.7%	42.1%	581.83	48.3%	76.9%

P	Problem instance
ε	Accuracy level of the area linearization
#UB	# of replications where no feasible soln. exists at initial pop.
#INF	# of replications where algorithm could not find a feasible solution
#ITER	Number of iterations algorithm terminates
suffix 0	… at initial population
suffix F	… at final population
%F	Percent of feasible solutions
BFD	Deviation of best feasible solution from optimal
AFD	Dev. of avg. of feas solns. in pop. from optimal.
CPU	Runtime of algorithm in seconds
IMPB	Impr. obtained in best solution wrt initial pop.
IMPA	Impr. obtained in pop. feas. avg. wrt initial pop.

Table 4.12 – Results of mutation based on adjacency for instances whose optimal solutions are known

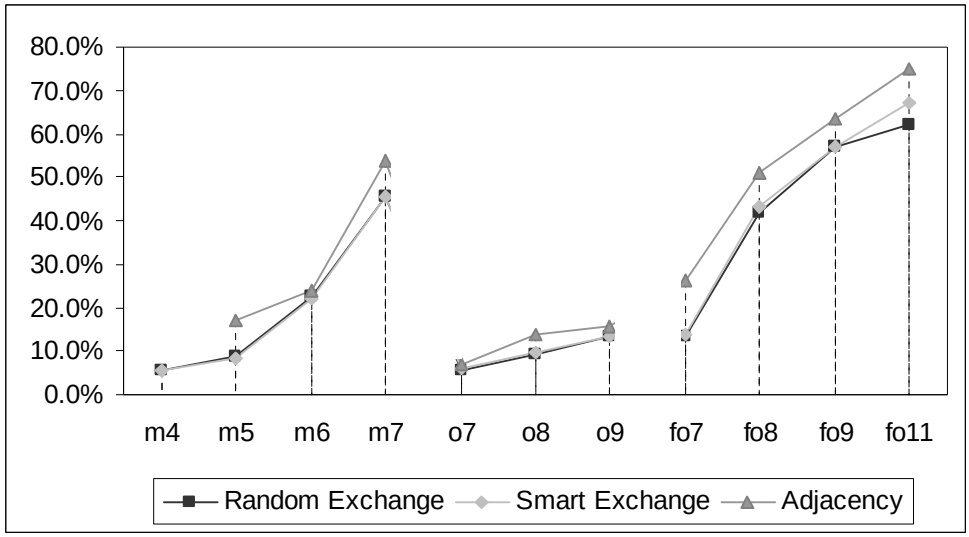
P	ε	#UB	#INF	#ITER	%F0	%FF	BFD0	BFDF	AFD0	AFDF	CPU	IMPB	IMPA
m5	**0.01**	8	2	884.8	0.6	24.6	25.3%	17.0%	26.1%	20.5%	71.47	0.1%	-9.6%
	0.1	7	0	907.8	0.2	27.2	11.0%	28.1%	33.1%	31.0%	67.10	1.5%	-7.9%
m6	**0.01**	0	0	972.6	10.4	82.8	99.0%	23.8%	170.0%	41.0%	100.51	75.2%	129.0%
	0.1	0	0	954.4	8.4	73.2	131.8%	20.9%	184.4%	37.1%	87.97	110.9%	147.3%
m7	**0.01**	2	0	981.8	2.6	79.4	189.7%	53.7%	208.5%	110.0%	126.48	137.6%	107.1%
	0.1	2	0	950.7	2.6	98.8	163.2%	60.3%	189.7%	104.1%	106.88	107.4%	90.6%
fo7	**0.01**	0	0	930.9	11.6	93.8	56.7%	26.1%	101.3%	41.2%	127.63	30.6%	60.1%
	0.1	0	0	927.7	24.4	85.4	62.3%	23.9%	113.9%	39.5%	108.71	38.4%	74.4%
fo8	**0.01**	0	0	938.3	8.2	97	122.6%	51.3%	151.2%	77.5%	143.78	71.4%	73.7%
	0.1	0	0	907.6	15.8	90.4	87.6%	42.5%	136.5%	65.0%	125.46	45.1%	71.4%
fo9	**0.01**	0	0	921.8	8.8	96.8	134.5%	63.7%	176.8%	99.9%	171.78	70.8%	76.8%
	0.1	0	0	949.5	16.6	80.4	115.8%	56.4%	182.0%	91.2%	152.58	59.4%	90.8%
fo10	**0.1**	0	0	884.1	15	99.4	143.5%	89.3%	204.4%	123.1%	161.06	54.2%	81.3%
fo11	**0.01**	0	0	876.2	94.2	87.4	125.1%	75.0%	212.9%	98.9%	217.22	50.1%	114.1%
	0.1	0	0	851.7	94.2	83.8	110.8%	73.2%	210.3%	96.2%	189.52	37.5%	114.0%
o7	**0.01**	0	0	827.3	12	73.6	19.5%	6.9%	32.3%	13.4%	109.07	12.7%	19.0%
	0.1	0	0	657.9	23.2	83.6	22.9%	7.5%	40.7%	14.9%	79.02	15.5%	25.8%
o8	**0.01**	0	0	818.4	7.2	87.8	32.3%	13.8%	41.4%	21.0%	130.60	18.4%	20.5%
	0.1	0	0	726.4	14.4	89	28.2%	10.0%	41.6%	17.6%	103.99	18.3%	24.0%
o9	**0.01**	0	0	864.2	8.6	86.4	37.9%	15.6%	50.4%	25.0%	164.44	22.3%	25.4%
	0.1	0	0	767.2	17.8	95	35.1%	19.6%	53.4%	30.2%	125.06	15.5%	23.1%
AVG	**0.01**	1.0	0.2	901.6	16.42	80.96	84.2%	34.7%	117.1%	54.8%	136.30	48.9%	61.6%
	0.10	0.8	0.0	862.3	21.15	82.38	82.9%	39.3%	126.4%	59.1%	118.85	45.8%	66.8%

P	Problem instance
ε	Accuracy level of the area linearization
#UB	# of replications where no feasible soln. exists at initial pop.
#INF	# of replications where algorithm could not find a feasible solution
#ITER	Number of iterations algorithm terminates
suffix 0	… at initial population
suffix F	… at final population
%F	Percent of feasible solutions
BFD	Deviation of best feasible solution from optimal
AFD	Dev. of avg. of feas solns. in pop. from optimal.
CPU	Runtime of algorithm in seconds
IMPB	Impr. obtained in best solution wrt initial pop.
IMPA	Impr. obtained in pop. feas. avg. wrt initial pop.

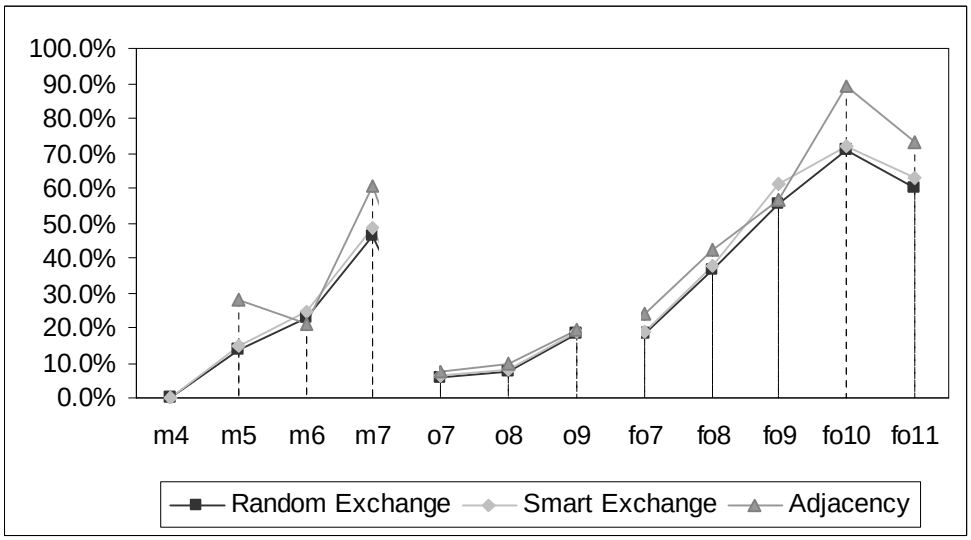
Table 4.13 – Results of instances whose optimal solutions are not known

MT	ε	P	#UB	#INF	#ITER	%F0	%FF	BFDLP0	BFDLPF	AFDLP0	AFDLPF	CPU	IMPB	IMPA
SE	1%	fo10	0	0	859.8	8.4	96.2	2084.9%	1465.9%	2428.1%	1670.0%	1445.11	619.0%	758.1%
		o10	0	0	642.7	7.8	94	1145.2%	947.8%	1250.8%	1023.3%	1091.27	197.4%	227.6%
		vc10	0	0	906.8	71.2	97.8	1436.4%	952.7%	2511.9%	1158.6%	1906.61	483.7%	1353.3%
	10%	o10	0	0	581.0	17	92.4	1053.1%	889.2%	1212.3%	972.0%	873.90	164.0%	240.2%
		vc10	0	0	891.4	88	98.2	1218.3%	876.7%	2251.4%	1040.5%	1460.81	341.5%	1210.8%
RE	1%	fo10	0	0	994	8.6	90.6	2032.5%	1434.3%	2336.8%	1571.2%	1598.1	68.3%	97.3%
		o10	0	0	654.4	7.8	97.2	1168.6%	951.8%	1276.1%	1010.6%	1064.1	184.0%	407.5%
		vc10	0	0	894	70.8	92.6	1418.2%	941.0%	2479.5%	1105.5%	1803.5	71.3%	99.6%
	10%	o10	0	0	577.7	16.4	94.4	1058.8%	879.0%	1225.1%	963.2%	819.83	181.5%	406.5%
		vc10	0	0	923.9	87.4	97.8	1300.9%	879.5%	2254.4%	1024.1%	1432.6	72.6%	100.6%
MA	1%	fo10	0	0	990.3	5.6	96.8	2283.7%	1519.0%	2516.5%	1892.3%	212.85	194.2%	439.0%
		o10	0	0	935.5	5.4	99	1207.4%	967.8%	1281.6%	1085.0%	200.68	129.5%	165.0%
		vc10	0	0	926.3	76	88	1359.5%	995.7%	2433.2%	1207.5%	260.01	167.7%	418.4%
	10%	o10	0	0	792.6	15.4	97.4	1065.9%	929.8%	1224.3%	1011.4%	148.44	169.8%	292.1%
		vc10	0	0	885.6	87.8	87.2	1343.5%	974.9%	2288.6%	1174.7%	185.25	151.0%	346.4%

P Problem instance
MT Mutation type
SE Mutation based on smart exchange
RE Mutation based on random exchange
MA Mutation based on adjacency
ε Accuracy level of the area linearization
#UB # of replications where no feasible soln. exists at initial pop.
#INF # of replications where algorithm could not find a feasible solution
#ITER Number of iterations algorithm terminates
suffix 0 … at initial population
suffix F … at final population
%F Percent of feasible solutions
BFDLP Deviation of best feasible solution from LP relaxation
AFDLP Dev. of avg. of feas solns. in pop. from LP relaxation
CPU Runtime of algorithm in seconds
IMPB Impr. obtained in best solution wrt initial pop.
IMPA Impr. obtained in pop. feas. avg. wrt initial pop.

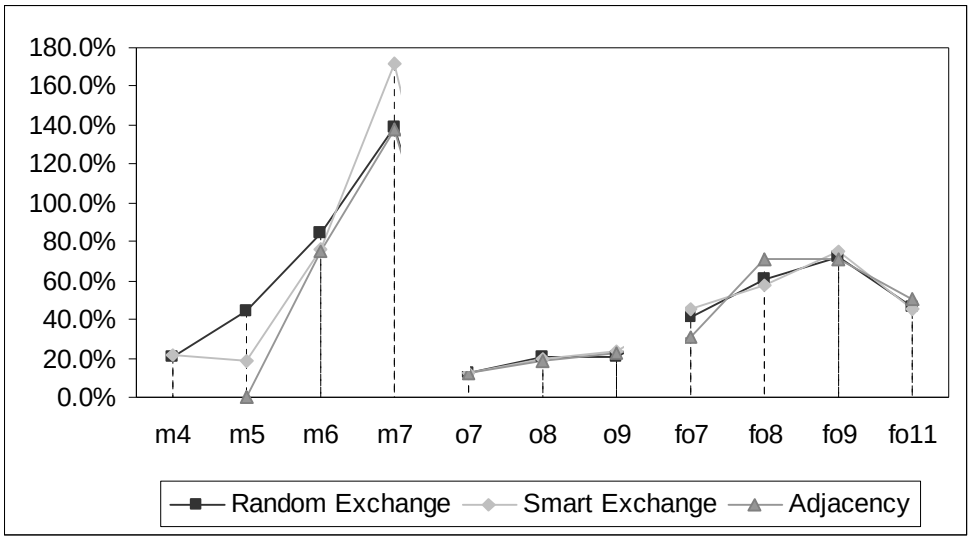


(a)

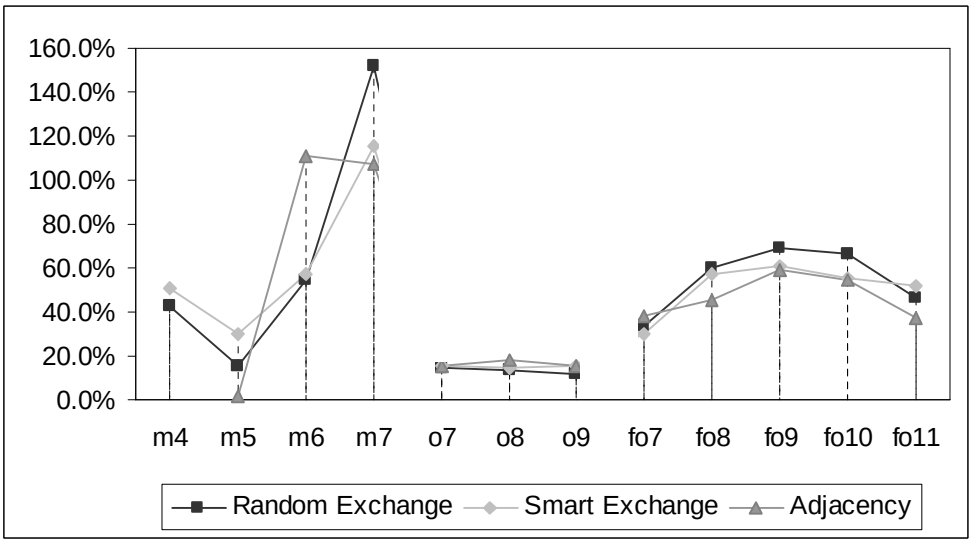


(b)

Figure 4.5 – Performance of the three mutation operators
for the deviation of the best feasible solution from optimal,
(a) at ε = 1%, (b) at ε = 10%



(a)



(b)

Figure 4.6 – Performance of the three mutation operators for the percent improvement in the best feasible solution, (a) at ε = 1%, (b) at ε = 10%

CHAPTER 5

CONCLUSION

In this chapter, first concluding remarks on the work done are provided, then directions for future work are discussed.

5.1 Concluding Remarks

In this thesis, the layout problem was analyzed. The problem was reviewed within the context of dynamic environments. Handling the problem by using different modeling objectives and structures, under an environment of changing requirements was observed to capture the problem better. It was also seen that obtaining good solutions for the single period layout problem had remained an unsolved issue. Therefore, effort was put on developing such algorithms. An LP relaxation based rounding heuristic and an evolutionary algorithm scheme were proposed in this study, both of which have encouraging results with room for improvement.

5.2 Future Work

Upon observation of the results, it is seen that the thesis has several promising further research issues that can be implemented fast. In the thesis, the improvement step after the LP relaxation based rounding heuristic is applied only once. Besides, the runtime for each of the truncated branch and bounds were given a limit of 2 hours of CPU time. Applying the improvement routine over and over, until it converges to a local optimum is an approach worth considering. It should be noted that, with this approach, the time limit on each improvement step becomes an important issue, too. Time limits currently being considered for experimentation range from 5 to 20 minutes.

It is also seen that the integrality gap of the solutions are very large. Testing the use of additional valid inequalities, and their impact on the LP relaxation values, and the LP relaxation based rounding heuristic is another further research issue.

Although being used in VLSI design for about ten years as of today, the sequence pair is a novel representation in the layout literature, allowing us to represent the solution to a layout without any limitations imposed on the shape of the layout. Therefore, heuristics exploiting this structure deserve an extra attention.

Firstly, the mutation operators discussed can be implemented as simple improvement heuristics. The alternatives considered at each iteration can be changed by applying a number of methods such as 1-opt, 2-opt, 1-exchange, 2-exchange, etc.

The second issue that is planned to be implemented is to develop an evolutionary algorithm firstly with employing combinations of the mutation operators and the crossover operator developed, then with new operators. Inclusion of a repair operator is also worth for analyzing.

REFERENCES

Adel El-Baz, M., 2004. A genetic algorithm for facility layout problems of different manufacturing environments *Computers & Industrial Engineering* **47**, 233-246.

Aiello, G. and Enea, M., 2001. Fuzzy approach to the robust facility layout in uncertain production environments *International Journal of Production Research* **39** (18), 4089-4101.

Aiello, G., Enea, M. and Galante, G., 2002. An integrated approach to the facilities and material handling system design *International Journal of Production Research* **40** (15), 4007-4017.

Al-Hakim, L., 2000. On solving facility layout problems using genetic algorithms *International Journal of Production Research* **38** (11), 2573-2582.

Amaral, A.R.S., 2006. On the exact solution of a facility layout problem. *European Journal of Operational Research* **173**, 508–518.

Anjos, M.F. and Vannelli, A., 2006. A new mathematical programming framework for facility-layout design. *INFORMS Journal on Computing* **18** (1), 111-118.

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

Aytug, H., Khouja, M. and Vergara, F.E., 2003. Use of genetic algorithms to solve production and operations management problems: a review *International Journal of Production Research* **41** (17), 3955-4009.

Azadivar, F. and Wang, J.J., 2000. Facility layout optimization using simulation and genetic algorithms *International Journal of Production Research* **38** (17), 4369-4383.

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

Balakrishnan, J., 1993 . Notes: "The dynamics of plant layout" *Management Science* **39** (5), 654-655.

Balakrishnan, J. and Cheng, C.H., 1998. Dynamic layout algorithms: A state-of-the-art survey *Omega, The International Journal of Management Science* **26** (4), 507-521.

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

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

Balakrishnan, J., Cheng, C.H. and Wong, K.F., 2003. FACOPT: a user friendly FACility layout OPTimization system *Computers & Operations Research* **30**, 1625-1641.

Batta, R., 1987. Comment on "The dynamics of plant layout" *Management Science* **33** (8), 1065.

Baykasoğlu, A. and Gindy, N.N.Z., 2001. A simulated annealing algorithm for dynamic layout problem *Computers & Operations Research* **28**, 1403-1426.

Baykasoğlu, A. and Gindy, N.N.Z., 2004. Erratum to a simulated annealing algorithm for the dynamic layout problem *Computers & Operations Research* **31**, 313-315.

Bayus, B.L., 1994. Are product life cycles really getting shorter? *Journal of Product Innovation Management* **11**, 300-308.

Bazaraa, M. S., 1975. Computerized layout design: A branch and bound approach. *AIIE Transactions* **7**, 432–438.

Benjaafar, S. and Sheikhzadeh, M., 2000. Design of flexible plant layouts *IIE Transactions* **32** (4), 309-322.

Benjaafar, S., Heragu, S.S. and Irani, S.A., 2002. Next generation factory layouts: Research challenges and recent progress *Interfaces* **32** (6), 58-76.

Benjaafar, S., 2002. Modeling and analysis of congestion in the design of facility layouts *Management Science* **48** (5), 679-704.

Bozer, Y.A., Meller, R.D. and Erlebacher, S.J., 1994. An improvement-type layout algorithm for single and multiple-floor facilities *Management Science* **40** (7), 918-932.

R.E. Burkard, S.E. Karisch and F. Rendl,. 1997. QAPLIB—A Quadratic Assignment Problem Library. *Journal of Global Optimization*, **10**, 391–403, electronic update: http://www.seas.upenn.edu/qaplib/ (revised 18.12.2005).

Cagliano, R., Acur, N. and Boer, H., 2005. Patterns of change in manufacturing strategy configurations. *International Journal of Operations & Production Management* **25** (7), 701-718.

Castillo, I. and Westerlund, T., 2005. An ε-accurate model for optimal unequal-area block layout design. *Computers & Operations Research* **32**, 429-447.

Castillo, I., Westerlund, J., Emet, S. and Westerlund, T., 2005. Optimization of block layout design problems with unequal areas: A comparison of MILP and MINLP optimization methods. *Computers & Chemical Engineering* **30**, 54-69.

Chang, M., Ohkura, K., Ueda, K. and Sugiyama, M., 2002. A symbiotic evolutionary algorithm for dynamic facility layout problem. *Proceedings of the 2002 Congress on Evolutionary Computation (CEC '02)* **2**, 1745-1750.

Chau, K.W., 2004. A two-stage dynamic model on allocation of construction facilities with genetic algorithm. *Automation in Construction* **13**, 481-490.

Chaudhry, S.S. and Luo, W., 2005. Application of genetic algorithms in production and operations management: a review *International Journal of Production Research* **43** (19), 4083-4101.

Chen, Y.K., Lin, S.W. and Chou, S.Y., 2002. An efficient two-staged approach for generating block layouts *Computers & Operations Research* **29**, 489-504.

Cohoon, J.P., Hedge, S.U., Martin, W.N. and Richards, D.S., 1991. Distributed genetic algorithms for the floorplan design problem *IEEE Transactions on Computer-Aided Design* **10** (4), 483-492.

Coit D.W., Smith A.E. and Tate, D.M., 1996. Adaptive penalty methods for genetic optimization of constrained combinatorial problems. *INFORMS Journal on Computing*, **8**,173-182.

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

De Toni, A. and Tonchia, S., 1998. Manufacturing flexibility: a literature review *International Journal of Production Research* **36** (6), 1587-1617.

Delmaire, H., Langevin, A. and Riopel, D., 1997. Skeleton-based facility layout design using genetic algorithms *Annals of Operations Research* **69**, 85-104.

Demir, E., 2004. Analysis of evolutionary algorithms for constrained routing problems. *MSc Thesis*, Middle East Technical University, Ankara.

Denizel, M. and Süral, H., 2006. On alternative mixed integer programming formulations and LP-based heuristics for lot-sizing with setup times, *Journal of the Operational Research Society*, **57**, 389-399.

Dimopoulos, C. and Zalzala, A.M.S., 2000. Recent developments in evolutionary computation for manufacturing optimization: problems, solutions, and comparisons *IEEE Transactions on Evolutionary Computation* **4** (2), 93-113.

Drezner, Z., Hahn, P.M. and Taillard, E.D., 2005. Recent advances for the quadratic assignment problem with special emphasis on instances that are difficult for meta-heuristic methods *Annals of Operations Research* **139**, 65-94.

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

Elbeltagi, E., Hegazy, T. and Eldosouky, A., 2004. Dynamic layout of construction temporary facilities considering safety *Journal of Construction Engineering and Management* **130** (4), 534-541.

Enea, M., Galante, G. and Panascia, E., 2005. The facility layout problem approached using a fuzzy model and a genetic search *Journal of Intelligent Manufacturing* **16**, 303-316.

Ettlie, J.E. and Penner-Hahn, J.D., 1994. Flexibility ratios and manufacturing strategy *Management Science* **40** (11), 1444-1454.

Ficko, M., Brezocnik, M. and Balic, J., 2004. Designing the layout of single- and multiple-rows flexible manufacturing system by genetic algorithms *Journal of Materials Processing Technology* **157-158**, 150-158.

Fu, M.C. and Kaku, B.K., 1997. Minimizing work-in-process and material handling in the facilities layout problem. *IIE Transactions* **29**, 29–36.

Gau, K.Y. and Meller, R.D., 1999. An iterative facility layout algorithm *International Journal of Production Research* **37** (16), 3739-3758.

Gero, J.S. and Kazakov, V.A., 1998. Evolving design genes in space layout planning problems *Artificial Intelligence in Engineering* **12**, 163-176.

Gupta, R.M., 1986. Flexibility in layouts: A simulation approach *Material Flow* **3**, 243-250.

Gómez, A., Fernández, Q.I., De la Fuente García, D. and García, P.J., 2003. Using genetic algorithms to resolve layout problems in facilities where there are aisles *International Journal of Production Economics* **84**, 271-282.

Hamamoto, S., Yih, Y. and Salvendy, G., 1999. Development and validation of genetic algorithm-based facility layout - a case study in the pharmaceutical industry *International Journal of Production Research* **37** (4), 749-768.

Hicks, C., 2004. A genetic algorithm tool for designing manufacturing facilities in the capital goods industry. *International Journal of Production Economics* **90**, 199-211.

Hu, M.H. and Wang, M.J., 2004. Using genetic algorithms on facilities layout problems. *International Journal of Advanced Manufacturing Technology* **23**, 301-310.

Jo, J.H. and Gero, J.S., 1998. Space layout planning using an evolutionary approach. *Artificial Intelligence in Engineering* **12**, 149-162.

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

Kim, J.G. and Goetschalckx, M., 2005. An integrated approach for the concurrent determination of the block layout and the input and output point locations based on the contour distance. *International Journal of Production Research* **43** (10), 2027-2047.

Kochhar, J.S., Foster, B.T. and Heragu, S.S., 1998. HOPE: A genetic algorithm for the unequal area facility layout problem. *Computers & Operations Research* **25** (7-8), 583-594.

Kochhar, J.S. and Heragu, S.S., 1998. MULTI-HOPE: a tool for multiple floor layout problems. *International Journal of Production Research* **36** (12), 3421-3435.

Kochhar, J.S. and Heragu, S.S., 1999. Facility layout design in a changing environment. *International Journal of Production Research* **37** (11), 2429-2446.

Kodama, C. and Fujiyoshi, K., 2003. Selected sequence-pair: An efficient decodable packing representation in linear time using sequence-pair. In *Proceedings of the 2003 conference on Asia South Pacific Design Automation*, 331-337.

Kouvelis, P., Kurawarwala, A.A. and Gutiérrez, G.J., 1992. Algorithms for robust single and multiple period layout planning for manufacturing systems. *European Journal of Operational Research* **63**, 287-303.

Kulturel-Konak, S., Smith, A.E. and Norman, B.A., 2004. Layout optimization considering production uncertainty and routing flexibility. *International Journal of Production Research* **42** (21), 4475-4493.

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

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

Lacksonen, T.A., 1997. Preprocessing for static and dynamic facility layout problems. *International Journal of Production Research* **35** (4), 1095-1106.

Lahmar, M. and Benjaafar, S., 2005. Design of distributed layouts *IIE Transactions* **37**, 303-318.

Lee, G.C. and Kim, Y.D., 2000. Algorithms for adjusting shapes of departments in block layouts on the grid-based plane *Omega, The International Journal of Management Science* **28**, 111-122.

Lee, Y.H. and Lee, M.H., 2002. A shape-based block layout approach to facility layout problems using hybrid genetic algorithm *Computers & Industrial Engineering* **42**, 237-248.

Ma, Z., Shen, Q. and Zhang, J., 2005. Application of 4D for dynamic site layout and management of construction projects *Automation in Construction* **14**, 369-381.

MacDuffie, J.P., Sethuraman, K. and Fisher, M.L., 1996. Product variety and manufacturing performance: Evidence from the international automotive assembly plant study *Management Science* **42** (3), 350-369.

Mak, K.L., Wong, Y.S. and Chan, F.T.S., 1998. A genetic algorithm for facility layout problems *Computer Integrated Manufacturing Systems* **11** (1-2), 113-127.

Marsh, R.F., Meredith, J.R. and McCutcheon, D.M., 1997. The life cycle of manufacturing cells. *International Journal of Operations & Production Management* **17** (12), 1167-1182.

Mavridou, T.D. and Pardalos, P.M., 1997. Simulated annealing and genetic algorithms for the facility layout problem: a survey. *Computational Optimization and Applications* **7**, 111-126.

Mawdesley, M.J., Al-jibouri, S.H. and Yang, H., 2002. Genetic algorithms for construction site layout in project planning. *Journal of Construction Engineering and Management* **128** (5), 418-426.

Meller, R.D. and Bozer, Y.A., 1996. A new simulated annealing algorithm for the facility layout problem. *International Journal of Production Research* **34** (6), 1675-1692.

Meller, R.D. and Gau, K.Y., 1996a. The facility layout problem: Recent and emerging trends and perspectives. *Journal of Manufacturing Systems* **15** (5), 351-366.

Meller, R.D. and Gau, K.Y., 1996b. Facility layout objective functions and robust layouts. *International Journal of Production Research* **34** (10), 2727-2742.

Meller, R.D., Narayanan, V. and Vance, P.H., 1999. Optimal facility layout design. *Operations Research Letters* **23**, 117-127.

Meller, R.D., Chen, W. and Sherali, H.D., 2006. A new formulation for optimal facility layout design. *Working paper.*

Meng, G., Heragu, S.S. and Zijm, H., 2004. Reconfigurable layout problem *International Journal of Production Research* **42** (22), 4709-4729.

Meyers, F.E. and Stephens, M.P., 2005. Manufacturing facilities design and material handling, 3rd edition *Pearson Prentice Hall*.

Molleman, E., Slomp, J. and Rolefes, S., 2002. The evolution of a cellular manufacturing system: A longitudinal case study *International Journal of Production Economics* **75**, 305-322.

Montreuil, B. and Ratliff, H.D., 1989. Utilizing cut trees as design skeletons for facility layout *IIE Transactions* **21** (2), 136-143.

Montreuil, B. and Venkatadri, U., 1991. Strategic interpolative design of dynamic manufacturing systems layouts *Management Science* **37** (6), 682-694.

Montreuil, B. and Laforge, A., 1992. Dynamic layout design given a scenario tree of probable futures *European Journal of Operational Research* **63**, 271-286.

Murata, H., Fujiyoshi, K., Nakatake, S. and Kajitani, Y., 1995. Rectangle packing-based module placement. In *Proceedings of IEEE International Conference on Computer-Aided Design*, 472–479.

Murthy, I., Mote, J. and Olson, D., 1991. A parametric approach to solving bicriterion shortest path problems *European Journal of Operational Research* **53**, 81-92.

Norman, B.A., Arapoglu, R.A. and Smith, A.E., 2001. Integrated facilities design using a contour distance metric *IIE Transactions* **33**, 337-344.

Norman, B.A. and Smith, A.E., 2006. A continuous approach to considering uncertainty in facility design *Computers & Operations Research* **33**, 1760-1775.

Nugent, C.E., Vollman, T.E. and Ruml, J., 1968. An experimental comparison of techniques for the assignment of facilities to locations. *Operations Research*, **16**, 150-173.

Ozdemir, G., Smith, A.E. and Norman, B.A., 2003. Incorporating heterogeneous distance metrics within block layout design *International Journal of Production Research* **41** (5), 1045-1056.

Pierreval, H., Caux, C., Paris, J.L. and Viguier, F., 2003. Evolutionary approaches to the design and organization of manufacturing systems *Computers & Industrial Engineering* **44**, 339-364.

Rajasekharan, M., Peters, B.A. and Yang, T., 1998. A genetic algorithm for facility layout design in flexible manufacturing systems *International Journal of Production Research* **36** (1), 95-110.

Rosenblatt, M.J., 1979. The facilities layout problem: a multi-goal approach *International Journal of Production Research* **17** (4), 323-332.

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

Rosenblatt, M.J. and Lee, H.L., 1987. A robustness approach to facilities design *International Journal of Production Research* **25** (4), 479-486.

Rosenblatt, M.J. and Kropp, D.H., 1992. The single period stochastic plant layout problem *IIE Transactions* **24** (2), 169-176.

Sanchez, L.M. and Nagi, R., 2001. A review of agile manufacturing systems *International Journal of Production Research* **39** (16), 3561-3600.

Savsar, M., 1991. Flexible facility layout by simulation *Computers & Industrial Engineering* **20** (1), 155-165.

Schnecke, V. and Vornberger, O., 1997. Hybrid genetic algorithms for constrained placement problems *IEEE Transactions on Evolutionary Computation* **1** (4), 266-277.

Shayan, E. and 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.P. and Meller, R.D., 2003. Enhanced model formulations for optimal facility layout. *Operations Research* **51** (4), 629-644.

Shewchuk, J.P. and Moodie, C.L., 2000. Flexibility and manufacturing system design: an experimental investigation. *International Journal of Production Research* **38** (8), 1801-1822.

Stockmeyer, L., 1983. Optimal orientations of cells in slicing floorplan designs, *Information and Control*, **57** (2-3), 91-101.

Suresh, G., Vinod, V.V. and Sahu, S., 1995. A genetic algorithm for facility layout. *International Journal of Production Research* **33** (12), 3411-3423.

Sweeney, D.S. and Tatham, R.L., 1976. An improved long-run model for multiple warehouse location *Management Science* **22** (7), 748-758.

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

Tam, K.Y. and Chan, S.K., 1998. Solving facility layout problems with geometric constraints using parallel genetic algorithms: experimentation and findings *International Journal of Production Research* **36** (12), 3253-3272.

Tate, D.M. and Smith, A.E., 1995. Unequal-area facility layout by genetic search. *IIE Transactions* **27**, 465–472.

Tavakkoli-Moghaddain, R. and Shayan, E., 1998. Facilities layout design by genetic algorithms *Computers & Industrial Engineering* **35** (3-4), 527-530.

Tommelein, I.D. and Zouein, P.P., 1993. Interactive dynamic layout planning *Journal of Construction Engineering and Management* **119** (2), 266-287.

Urban, T.L., 1992. Computational performance and efficiency of lower-bound procedures for the dynamic facility layout problem *European Journal of Operational Research* **57**, 271-279.

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

Urban, T.L., 1998. Solution procedures for the dynamic facility layout problem *Annals of Operations Research* **76**, 323-342.

van Camp, D.J., Carter, M.W. and Vannelli, A., 1991. A nonlinear optimization approach for solving facility layout problems *European Journal of Operational Research* **57**, 174-189.

Wang, M.J., Hu, M.H. and Ku, M.Y., 2005. A solution to the unequal area facilities layout problem by genetic algorithm *Computers in Industry* **56**, 207-220.

Wang, R.L. and Okazaki, K., 2005. Solving facility layout problem using an improved genetic algorithm *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences* **E88-A** (2), 606-610.

Webster, D.B. and Tyberghein, M.B., 1980. Measuring flexibility of job-shop layouts *International Journal of Production Research* **18** (1), 21-29.

Wemmerlöv, U. and Hyer, N.L., 1989. Cellular manufacturing in the US industry: A survey of users *International Journal of Production Research* **27** (9), 1511-1530.

Wu, Y. and Appleton, E., 2002. The optimisation block layout and aisle structure by a genetic algorithm *Computers & Industrial Engineering* **41**, 371-387.

Yang, T. and Peters, B.A., 1998. Flexible machine layout design for dynamic and uncertain production environments *European Journal of Operational Research* **108**, 49-64.

Zouein, P.P. and Tommelein, I.D., 1999. Dynamic layout planning using a hybrid incremental solution method *Journal of Construction Engineering and Management* **125** (6), 400-408.

APPENDIX A

A REVIEW OF THE DYNAMIC LAYOUT PROBLEM*

Similar to the Single Period Layout Problem (SPLP), there are mainly two types of mathematical formulations for the DLP:

1. For equal sized departments (EDLP)
2. For unequal sized departments (UDLP)

EDLP is usually formulated as the Quadratic Assignment Problem (QAP) and UDLP as Mixed Integer Linear Programming (MILP).

As mentioned, there are mainly two objectives in DLP: minimization of material handling cost and rearrangement cost. The problem is usually handled as single objective where both are given in monetary terms. It can also be handled as a multiple objective problem and selection can be made from a set of efficient solutions.

Single period layout problem is a special case of the DLP where the planning horizon is limited to one period. DLP is relevant when there is a tradeoff between the material handling cost and the rearrangement cost objectives. If the relayout cost is dominant, then the problem reduces to a simple SPLP where the flow matrix is found by adding up the flow matrices of all periods. In this case, no relayout is done and a single layout is used for all periods. On the other hand, if the material handling cost is dominant, then the problem reduces to a series of SPLPs, the optimal solution to the DPLP being the collection of the optimal SPLP solutions for each period.

* A summarized version of this review was used as a part of the course material for IE 425 – Facilities Location and Layout course in Spring 2006.

A.1 Formulation for EDLP

The following formulation is taken from Kaku and Mazzola (1997).

Index sets

N : Set of departments as well as the set of locations

T : Set of time periods in the planning horizon

$T' = T \setminus \{1\}$

Parameters

f_{ikt} : the cost of the work flow (per unit distance) between departments i and k in period t

d_{jl} : the distance between locations j and l

c_{ijt} : the cost of assigning department i to location j in period t

s_{it} : the cost of moving department i at the beginning of period t to a location different from which it was located in the previous period

g_t : the fixed cost incurred in the beginning of period t if any department is moved to a new location

Decision variables

$$x_{ijt} = \begin{cases} 1, & \text{if department } i \text{ is assigned to location } j \text{ in period } t \\ 0, & \text{otherwise} \end{cases}$$

$$y_{it} = \begin{cases} 1, & \text{if department } i \text{ is moved to a new location at the beginning of period } t \\ 0, & \text{otherwise} \end{cases}$$

$$z_t = \begin{cases} 1, & \text{if any department is moved to a new location at the beginning of period } t \\ 0, & \text{otherwise} \end{cases}$$

$$\textit{Minimize} \quad \sum_{t\in T}\sum_{i\in N}\sum_{j\in N}\sum_{k\in N}\sum_{l\in N} f_{ikt} d_{jl} x_{ijt} x_{klt} \tag{A.1}$$

$$+ \sum_{t\in T}\sum_{i\in N}\sum_{j\in N} c_{ijt} x_{ijt} \tag{A.2}$$

$$+ \sum_{t\in T}\sum_{i\in N} s_{it} y_{it} \tag{A.3}$$

$$+ \sum_{t\in T} g_t z_t \tag{A.4}$$

subject to

$$\sum_{j\in N} x_{ijt} = 1 \qquad i\in N,\ t\in T \tag{A.5}$$

$$\sum_{i\in N} x_{ijt} = 1 \qquad j\in N,\ t\in T \tag{A.6}$$

$$y_{it} = \sum_{j\in N} x_{ijt-1}\left(\sum_{l\in N\setminus\{j\}} x_{ilt}\right) \qquad i\in N,\ t\in T' \tag{A.7}$$

$$z_t \geq y_{it} \qquad i\in N,\ t\in T' \tag{A.8}$$

$$x_{ijt} \in \{0,1\} \qquad i\in N,\ j\in N,\ t\in T' \tag{A.9}$$

$$y_{it} \geq 0 \qquad i\in N,\ t\in T' \tag{A.10}$$

$$z_t \geq 0 \qquad t\in T' \tag{A.11}$$

In the objective function, (A.1) is the material handling cost dependent on the total distance traveled. (A.2) is the total fixed cost of locating a department in a specific location in a specific period, which is usually not included, with the assumption that the cost of building a department is independent from the location it is placed. However, it can be incorporated into most of the algorithms if desired. (A.3) shows the cost of moving a department in a specific period, which depends only on the department itself, independent from the place it is moved to. This is due to the fact that fixed cost of moving the machinery dominates the variable cost that is usually dependent on the total distance traveled for the move. However, for the situations where this assumption is not valid, the parameter can be modified as s_{ijlt}, the cost of moving department i from location j location l in period t. Such a formulation is adopted in Balakrishnan et al. (1992). This of course requires the variable y_{it} to be changed into y_{ijlt}, together with the relevant modifications in the constraints and the objective function. (A.4) calculates the fixed cost of making any change in the layout in a specific period.

Constraint set (A.5) restrict each department to be assigned to a unique location in each time period, and constraint set (A.6) requires that in each time period, each location has exactly one department assigned to it. Nonlinear constraint set (A.7) keeps track of whether a department has moved or not. Similarly, constraint set (A.8) keeps track of whether a rearrangement has taken place or not. Constraints (A.9), (A.10) and (A.11) are simple sign constraints.

Unless otherwise stated, the discussed solution procedures attempt to solve the problem formed by (A.1), (A.3), (A.5), (A.6), (A.7), (A.9) and (A.10).

An important note is that since the problem was never attempted to be solved by only using the mathematical model, most of the papers (even the first person to name the problem, Rosenblatt, 1986) do not give a complete formulation. Within the ones that provide a formulation, none attempts to linearize the constraint set (A.7).

A.2 Solution Approaches for EDLP

A.2.1 Dynamic Programming

Rosenblatt (1986) was the first to introduce the dynamic layout problem, where the issue of designing a layout is considered as a task that evolves through time. He introduces a dynamic programming approach to solve the problem.

- The number of departments is constant throughout the planning horizon in the original formulation. Lacksonen and Enscore (1993) adapt the algorithm to handle changing number of departments. They create states by placing the new department required in each location.
- DP cannot be handled computationally as the number of departments increases since $(n!)^t$ options would have to be explicitly or implicitly evaluated in order to find the optimal solution.

The following notation and explanations for the Dynamic Programming (DP) formulation are taken from Balakrishnan and Cheng (1998) with some small

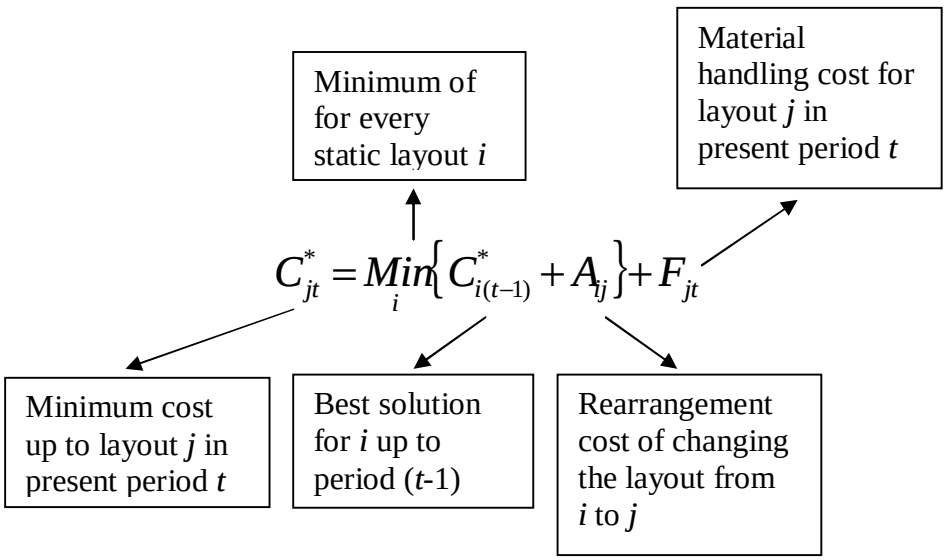
changes. Each period in the planning horizon forms a stage and each static layout forms a state. The DP is solved using backward recursion.

L_i : Layout i

A_{ij} : Rearrangement (sum of department shifting costs) cost when changing from layout L_i layout L_j. This cost is independent of the period in which it occurs.

F_{it} : Material handling cost for layout L_i in period t.

C_{it}^*: Minimum total costs (material handling and shifting) for all periods up to t where L_i is used in period t.



The network representation of the DP in the form of a shortest path problem is given in Figure A.1, which is similar to how Balakrishnan et al. (1992) handle the problem. In the figure, each node corresponds to a layout configuration and the arcs show the transitions between layouts. The cost of each arc is shown above it. The cost of the arcs who are directed between same layouts only include the material handling costs of that layout, whereas arcs that point a different layout also include the rearrangement cost from the previous layout to the following layout.

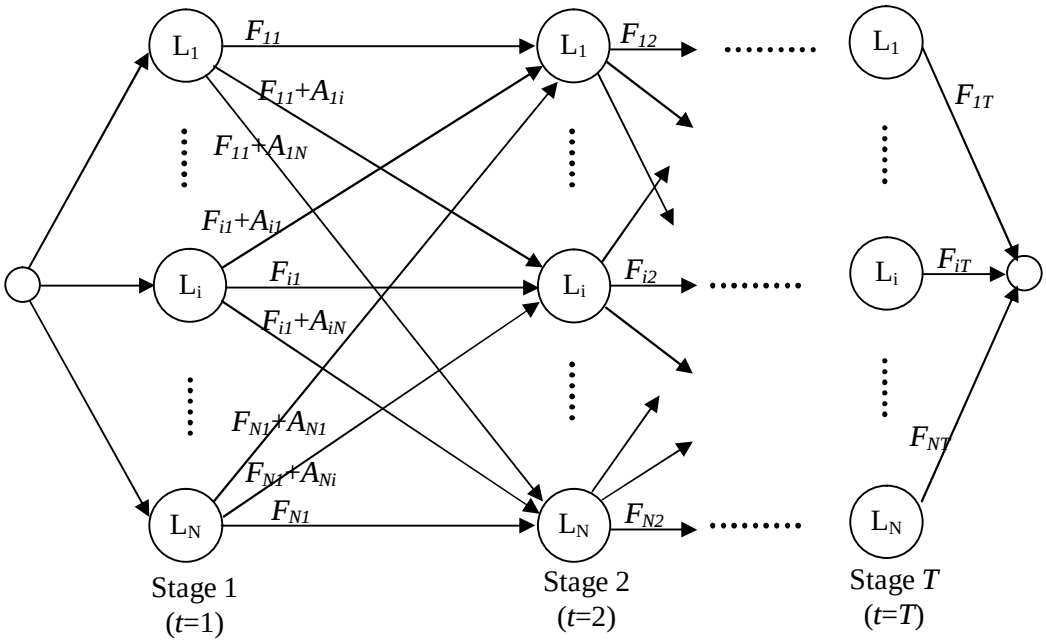


Figure A.1 – The network representation of the DP formulation

Node elimination (fathoming) strategies

Firstly, Rosenblatt (1986) proposed an adaptation from Sweeney and Tatham (1976). The notation is as follows:

Z_{tr} : The value of the r^{th} best solution to the SPLP for period t

Z^{LB} : Lower bound on the value of the optimal multi-period solution

Z^{UB} : Upper bound corresponding to the best incumbent feasible solution to the multi-period problem (rearrangement costs included)

Please note that a discussion on lower and upper bounds for the problem will be given in Sections A.2.6 and A.2.7 respectively.

If $K = Z^{UB} - Z^{LB}$ and R is given by $Z_{t,R} - Z_{t,1} \leq K$ and $Z_{t,R+1} - Z_{t,1} > K$, then, in any period t, no static solution with value r>R may become part of an optimal solution.

This condition eliminates states (layouts) whose material handling costs exceed the best layout in their stage (time period) greater than the gap between the lower bound

(LB) and the upper bound (UB). Such layouts cannot be optimal since the total cost exceeds the UB no matter what.

This procedure is not very efficient since the condition is very strong and requires a good UB, but still may not eliminate any states even if the optimal solution is used as the UB. Moreover, it requires finding the layouts' ranks, which corresponds to solving the QAP for $(N\text{-}1)T$ times. However, an ease it brings is that F_{it} required in the DP can be directly taken from the SPLP solution if this method is used for fathoming.

Balakrishnan (1993) proposes another fathoming procedure whose performance depends on the cost of shifting relative to the cost of material handling flow. With the same notation used, he suggests that

In period t, no static layout configuration L_r, with cost Z_{tr}, may become part of the optimal multi-period solution if $Z_{tr} - Z_{t1} > 2.\text{Max}\{A_{ij}\}$, where A_{ij} denotes the rearrangement cost of shifting from configuration L_i to L_j.

This condition can be explained in the following manner: Instead of using L_r, the layout corresponding to Z_{tr}, one can use the best layout of that stage (time period) and in the worst case make two costly rearrangements (one due to the previous stage and one for the following stage), and would still be better-off. Note that the condition reduces to $Z_{tr} - Z_{t1} > \text{Max}\{A_{ij}\}$ for the first and last stages since there is only one rearrangement that can occur.

A.2.2 Other Procedures That Consider Additional Assumptions

Balakrishnan et al. (1992) consider the problem when the cost of relayouts is limited by a budget. They model the problem as a Constrained Shortest Path (CSP) network flow, motivated by the efficiency of the network based models. By relaxing the constraint and considering it as another objective, the CSP model is converted into a bi-criterion shortest path model. Pareto-optimal paths are obtained by solving this problem, which provide upper bounds. In the paper, the implementation of a

specialized primal shortest-path simplex algorithm is compared with dynamic programming. Since both algorithms are designed to produce optimal solutions, the only area they can compete is the solution time. Here, it is found out that CSP performs better than DP in most cases, except for those with small problem sizes or with tight budget constraints or both. The reasons behind these situations are explained by the authors as follows:

> When going from small to large problems, the solution time increase was much greater for the DP algorithm than for the CSP. This difference is due to the fact that the DP algorithm is a pure enumeration algorithm, whereas the CSP is a combination of the simplex method and enumeration. Enumeration methods are sensitive to both the number of constraints and the number of arcs, while the simplex method is sensitive mainly to the number of constraints. Since the Constrained Dynamic Plant Layout Problem (CDPLP) has many more arcs as compared to constraints (nodes or layouts), the effects of increased problem size is felt more by the DP algorithm. … The constraint level also had an effect on solution time. Tighter constraints lead to more fathoming of arcs and thus reduced solution time. This effect was also greater on the DP approach.

Overall, the CSP approach seems to be a strong rival to the DP approach that has not received much attention later. However, one should first analyze the algorithm used by the authors, which is an algorithm developed by Murthy at al. (1991), in order to be able to come up with strong conclusions.

Urban (1998) proposes an incomplete dynamic programming approach for the solution of the problem when only a fixed cost is incurred when a rearrangement is made (i.e. the objective function is formed by (A.1) and (A.4)). In the algorithm, the main decision variable is Z, the vector showing whether a rearrangement is made or not in each period. Followed by the proof of Batta (1987), the optimal layout for a part of the vector beginning with a 1 and followed by zeros can be solved as a static QAP, by summing up the workflows of the corresponding periods. Hence, if m changes are made, the problem can be solved by solving $m+1$ static QAPs. The reason that the procedure is called “incomplete DP” is that the cost of a configuration cannot be evaluated for a period where rearrangement is not made,

until a period with rearrangement is reached. The real ease of the problem is that once a rearrangement decision is made, the rest of the problem becomes independent of the previous arrangements. So, after calculating all possible costs (by solving *T(T+1)/2* QAP subproblems), one can formulate the problem as a shortest path network.

Urban (1998) further discusses the fixed rearrangement cost case by considering the situations where the material handling cost and the rearrangement cost are not in comparable terms. To handle this situation, he takes the problem as a bicriteria problem and generates a discrete efficiency frontier (i.e. a set of nondominated solutions).

A.2.3 Conventional Heuristics

The first group of heuristics is based on the dynamic programming formulation of the model. Here, not all states (layouts) are included in a stage, but a number of them are picked instead. The ways of choosing these states differ by the method of generating and/or picking the states to be included; and they can be summarized as follows:

1. For each stage, include only the best static layout of each period. With this method, the maximum number of layouts for each period (*N*, with the notation used in Figure 1) and the number of QAPs that should be optimally solved is *T*.
2. Very similar to method 1, include only the *m* best static layouts of each period for each stage. With this method, the maximum number of layouts for each period (*N*) and the number of QAPs that should be optimally solved is *mT*. In fact, method 1 is the case where *m*=1. Since this method is adapted from Ballou (see Rosenblatt, 1986), it as also referred to as the "Ballou method".
3. Include randomly generated layouts. An important factor for this method is the percentage of random layouts to total number of possible layouts.
4. Balakrishnan et al. (1992) also use an equal weight combination of random and best layouts. This method is usually referred to as a "mixed strategy".

5. Balakrishnan et al. (2000) suggest using all layouts generated by the forecast windows of Urban's pairwise exchange heuristic (1993).

Balakrishnan et al. (1992) use the same percentage of layouts for all best, random and mixed strategies; the level being either 7% or 14%. Urban (1993) use 7% and 14% for the random strategies, 3% for the best strategy where m=4.

Balakrishnan et al. (1992) observe that when the number of layouts considered for each period is the same, selecting best layouts performs better than selecting random layouts. More interestingly, selecting a mixture of best and random layouts performs almost as good as selecting purely best layouts.

Urban's Pairwise Exchange Heuristic (1993)

The review paper by Balakrishnan and Cheng (1998) explains the Urban's algorithm (1993) very well. The explanation is as follows with small modifications to the original text:

"The pairwise interchange procedure proposed by Urban (1993) is the multi-period equivalent of CRAFT and incorporates layout rearrangement costs ($\sum_{\forall i} s_{it} y_{it}$, with the notation used in Section A.2 and A_{ij}, with the notation used in Section A.3.1). This heuristic makes use of 'forecast windows', m, to find different sets of good layout plans for the planning horizon. The forecast window, m, ranging from 1 to T, is the number of periods being considered when the pairwise exchange is performed. Using an initial layout and a steepest-descent pairwise-interchange procedure, one set of layouts is obtained for the given planning horizon for each forecast window.

For example, when the forecast window is 1, in each run of the pairwise interchange procedure, only material flows from a single period are considered. An assumed or existing initial layout is used to find the most appropriate layout for period 1 by considering the material flows for period 1 only. Then this newly generated layout for period 1 is used as the initial layout for period 2, for which pairwise interchange procedure is now used to determine a good layout by considering the material flow of period 2 only. The pair of departments that will be interchanged at each iteration are those that maximize the reduction in total cost (material handling cost reduction

less any variable rearrangement costs included). This process is repeated for each period in the planning horizon using the newly generated layout for the previous period as the initial layout for pairwise interchange. Thus, a layout plan for the entire planning horizon is obtained. The total cost of the plan is the sum of the material handling flows for every period in the planning horizon and the costs of rearranging the layout at the end of each period if necessary.

The next stage of the procedure involves using a forecast window of 2. This is shown in Figure A.2.

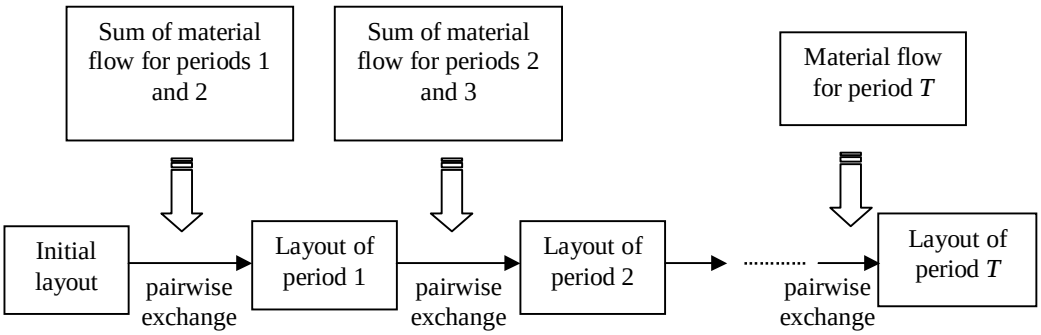


Figure A.2 – Application of Urban's heuristic with a forecast window of 2

The material flows of periods 1 and 2 are combined to determine the layout for period 1 from an initial layout using pairwise interchange. This final layout for period 1 serves as an initial layout for the next period's analysis. Similarly, the flow costs for periods 2 and 3 are combined and used to determine the layout for period 2, and so on. Thus, a look ahead principle is used. Note that in period T, only the material flow for period T is used as period T+1 does not exist. This process is repeated for m=1,2,…, T and a layout plan and an associated cost is obtained for each m. The plan with the lowest cost is selected as the solution."

In this heuristic, as the time window is enlarged, more periods are taken into consideration while determining the layout of a period. Hence, the effects of the changes in the interdepartmental flows are alleviated, mostly reducing the needs for relayout. On the other hand, smaller time windows are more sensitive to changes in the flows, mostly resulting in more frequent changes in the layout. By considering all possible time windows and comparing their costs, one can obtain information regarding the dynamics of the manufacturing environment besides a nice layout plan. This "information" can be explained as follows:

Depending on the relative importance of the relayout costs to the material handling costs, not all changes in the interdepartmental flows would be worth making a change in the layout. These small changes can be handled by using a "compromise" layout, that is not necessarily the best for each time period, but is considerably good for all combined together. Then, a time would come when these flow changes are too costly to be handled by a compromise, which is the time of relayout. This heuristic tests several time windows for determining for how many periods to handle the changes with a compromise. So, this might provide an insight to the manufacturing dynamics. However, it should be kept in mind that here the number of periods whose data is considered is always constant, which may reduce the solution quality. On the other hand, since this procedure is only a heuristic, disregarding this issue is most probably not much of a deal.

Since the algorithm is of the improvement type, the solution quality is dependent on the initial solution used. In the comparison tests made by the author, the pairwise exchange heuristic initialized with two different layouts outperformed DP with randomly generated layouts and performed approximately the same as the best layouts. When it is considered that one needs to solve T QAPs for the best layouts method, it becomes obvious that the heuristic performs well.

Balakrishnan et al. (2000) propose an improvement to the pairwise exchange algorithm of Urban (1993). In Urban's heuristic, a layout for a period is never changed once it is determined. Thus, the quality of the solution in a later period

depends on the quality of the preceding layouts. In the backward method, this disadvantage is reduced. Initially Urban's method is used to solve the problem. Then, a backward pass pairwise exchange is performed on each solution obtained by a forecast window (m backward passes in total, each backward pass having a forecast window of 1). The best of these solutions is selected as the final solution. This method will not generate a layout that is worse than the forward pass.

A.2.4 Adaptations from SPLP Algorithms

As well as providing a comparison of five algorithms, Lacksonen and Enscore (1993) also make a contribution by adapting four algorithms developed for the SPLP to DLP. The algorithms with the modifications are explained below. The explanations are written by combining the texts by Lacksonen and Enscore (1993) and Balakrishnan and Cheng (1998), with some modifications.

Exchange Algorithm (CRAFT)

CRAFT starts with one or more random layouts and exchanges pairs of departments in order to find solutions with lower cost values. The process is repeated until no more improvements can be found. In the adaptation of Lacksonen and Enscore (1993), each run starts with eight random layouts. Then pairs of departments are analyzed for exchanges over all consecutive blocks of time in a CRAFT based method. This converts the static pair-wise exchange into a dynamic one.

Cutting Planes

This heuristic developed for QAP combines cutting planes with an exchange routine in an iterative heuristic. The routine starts with a random solution. At each iteration, an assignment routine finds the estimated best assignment subject to all departments moving to new locations. Since this 'cut' may eliminate the optimal solution, each iteration ends with an exchange routine. The process is repeated for different starting solutions. In solving the DLP, the assignment routine considers departments stationary between time periods and only the exchange routine is used to consider rearrangements.

Branch and Bound

This method is a modification of a branch and bound algorithm developed for optimally solving QAP. Each node in the tree represents a partial assignment with a lower bound on cost. The node with the highest lower bound is expanded into two new nodes by including and excluding the assignment of the next unassigned department to a given location. Nodes are expanded until they are fathomed (their lower bound exceeds the best known upper bound) or until a complete assignment is found which is a new best solution and upper bound. The algorithm employs the cutting plane results as an upper bound. To handle multiple time periods, departments are only permitted to be assigned to the proper time period and the lower bound calculation is revised by adding a term which estimates the cost of all time periods which do not have any assignments made yet. The procedure is run as a heuristic by storing only the 25 most promising nodes and by terminating after 50,000 nodes are analyzed.

Dynamic Programming

This is the DP method of Rosenblatt (1986), where not all possible layouts, but a number of them are used as the states, as explained in Section A.3.3. The number of layouts used at each stage in the DP ranges from 30 to 700, which depends on the problem size (N, T) and whether new departments are introduced or not. The first T states to use are determined by applying the exchange algorithm separately to each period's data. The rest of the states are created by swapping department pairs to create hybrids between the good layouts of consecutive time periods. In addition, when new departments are required, states are created by placing them in each location.

Cut Trees

Cut trees were introduced as alternative design skeletons for SPLP by Montreuil and Ratliff (1989). Brief information on cut trees and their application in layout problems is given below, which is summarized from Montreuil and Ratliff (1989).

“A cut between two nodes in a graph is a partition of all nodes into two distinct subsets, each subset containing one of the two cut nodes. A cut tree for a graph is a spanning tree where the arc of minimum weight on the unique path separating two nodes corresponds to the minimum cut separating the two nodes in the original graph.

The cut tree has some very useful characteristics when used as a design skeleton for a layout:

1. If you want to partition the cells into two non-empty sets so that the flow between the two sets is minimized, the cut tree indicates the optimum partition. This indicates a fundamental difference between the cut tree and the other graph theoretic models proposed since then for obtaining a design skeleton. The cut tree focuses on which cells should be separated while the other models focus on which departments should be made adjacent to each other.
2. The numbers on the links of the cut tree indicate the average amount of flow which would cross each link if the tree is used as the flow network. This provides the designer with some very valuable insights with regard to the cost of increasing the length of the aisles.
3. If the aisle structure for the layout is restricted to be a tree with all aisle segments the same length, then the cut tree provides the aisle structure which minimizes the number of trips times the distance traveled.”

In the original application, each department is represented by a node and material flows are represented by the arcs. The cut tree obtained from the resulting graph is used as a design skeleton for forming layout alternatives by hand, among which a satisfactory solution is selected. The skeleton can also be used as an input to a mathematical formulation for computerized applications.

In the adaptation by Lacksonen and Enscore (1993), each department in each time period is represented by a node. Arcs represent material flow and layout rearrangement costs. Cut tree arcs are then manually converted into layouts. This

manual layout is then fed into the exchange algorithm to obtain better results, since there is not a unique way of converting the tree into a layout.

32 problems ranging in size from six departments and three periods to thirty departments and five periods were tested. In the larger problems optimality could not be proved. The authors found that the cutting plane and branch and bound algorithms performed better than the exchange algorithm. DP and cut tree performed poorly. Also, the performance of the cut trees varied a lot. In addition, they were not applied to the thirty department problems as its implementation became difficult. The branch and bound was not able to solve the thirty department problems within the allotted CPU time because of insufficient bounds. On the thirty department problems the exchange algorithm and DP provided solutions that were respectively, 1.3% and 3.7% more costly on average than the cutting plane solution. The authors state the best algorithm as the cutting plane, however the cutting plane method required 27 times and 70 times respectively as much computation time on average than the exchange algorithm and DP. Further, the cutting plane is limited to equal sized layouts. Also, different implementations of the compared algorithms may result in different solutions. For example, the DP would probably perform better if it included more states. Hence, the DP whose duration is as much as the cutting plane might give better results. Similarly, the branch and bound might perform better with tighter bounds.

Although this study is not the best algorithm comparison ever made, the effort to adapt algorithms designed for SPLP to DLP is noteworthy and the solutions provide some insight.

A.2.5 Metaheuristics

Use of meta-heuristics for solving combinatorial optimization problems is currently one of the most attractive research areas. This applies to DLP as well. Currently, there is one application of Simulated Annealing (SA) by Baykasoglu and Gindy (2001), one application of Tabu Search (TS) by Kaku and Mazzola (1997) and several applications of Evolutionary Algorithms (EA) or Genetic Algorithms (GA).

One common point of all these meta-heuristics is the need of a neighborhood definition. This corresponds to the mutation operator in EA and the moves in SA and TS. In Baykasoglu and Gindy (2001) and Balakrishnan and Cheng (2000), a move is simply a swap (interchange) of departments in a single period. Note that a swap applied to a feasible solution always results in a feasible solution since departments are of equal size.

All GAs proposed use a single string representation where each static layout is represented by a string and the concatenation of them form the dynamic layout string. An example representation is shown in Figure A.3.

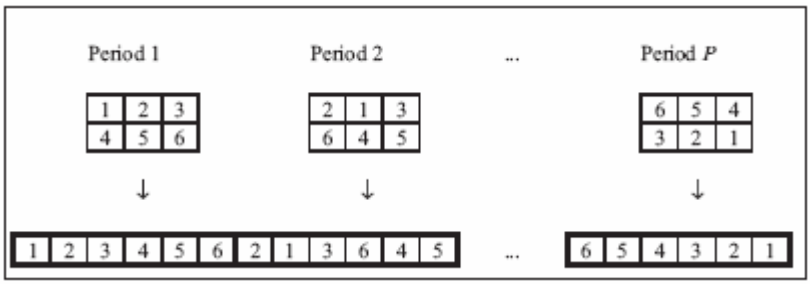


Figure A.3 – Encoding of a string in dynamic layout (Balakrishnan et al., 2003a)

Conway and Venkataramanan (1994) developed the first solution procedure based on GA to DLP, which will be referred as CVGA. They work on the extension studied by Balakrishnan et al. (1992), which considers a budget constraint for the rearrangement costs over the whole planning horizon. They use a single-point crossover operator and handle the possible infeasibilities by a kind of repair mechanism. They do not employ any mutation operator. The selection is based on the roulette wheel technique. They demonstrate their results on the 2x3, 5 period example of Rosenblatt (1986) and a 3x3, 5 period problem.

The next application of meta-heuristics is a tabu search heuristic by Kaku and Mazzola (1997). In their method, pairwise interchange is used as the neighborhood move. Classical elements of a tabu search are employed, such as a tabu list and intensification and diversification strategies.

As mentioned before, Kochhar and Heragu (1999) claim that changes in the flow patterns are known just slightly ahead of when they occur, so that one can decide about only a single relayout at a time. They consider two cases: a known initial layout and an initial layout to be developed. Then they modify the layout in the second period. Their approach is basically an adaptation of a multi-floor single period layout heuristic (MULTI-HOPE) to a two-period DLP. The algorithm they use is named “DHOPE”, which is based on GAs.

The GA proposed by Balakrishnan and Cheng (2000), which will be referred as NLGA, makes use of a nested loop structure, where the inner loop is dedicated to intensification and the outer loop is dedicated to diversification by the addition of randomly generated solution. The procedure employs a different crossover operator, named as “point-to-point crossover operator”, which can basically be described as applying one-point crossover at every possible crossover point and choosing the best feasible solution among the solutions generated as the offspring. They also apply the simple mutation operator of a swap (interchange) of departments in a single period. They demonstrate their results on 6, 15 and 30 department size problems with 5 and 10 periods. On the average, CVGA performs better than NLGA only in the 15 department – 5 period case, being outperformed by NLGA in all other problem types.

The application of Baykasoglu and Gindy (2001) is a standard SA, with no modifications special to the DLP. The results of their algorithm on the test problems of Balakrishnan and Cheng (2000) outperform both CVGA and NLGA in all instances for 15 and 30 department problems; based on the correction they published (Baykasoglu and Gindy, 2004). The algorithms perform similar for 6 department problems.

However, the hybrid GA approach developed by Balakrishnan et al. (2003a) outperforms the SA in all instances for the 15 and 30 department size problems. The most significant novelty of their procedure is the use of the knowledge in the literature on DLP. Firstly, they use a crossover operator based on Dynamic Programming. A number of layouts are chosen from the mating pool with the tournament selection method. Then, they are cut into their pieces of single period layouts. Then, Dynamic Programming is applied to these and the found solution is taken as the offspring. The mutation operator also makes use of a well known method – CRAFT. Instead of a single random swap, all pairwise exchanges are evaluated and the one with the most improvement is selected until no improvement can be made. The third procedure they apply is the use of Urban's pairwise exchange heuristic (1993) for initial population generation. These populations give better results than randomly generated populations. A summary of their choice of parameters and methods is given in Table A.1.

Although all results are available, no information on the computational time is given in any of the papers. Balakrishnan et al. (2003a) state that the largest problem (30 departments – 10 periods) was solved in about 1000 CPU seconds; hence the algorithm is able to solve practical sized problems in a reasonable amount of time.

Chang et al (2002) propose a symbiotic EA, where there is a population for the layout of each period. Solutions are evaluated by the "symbion" that consists of randomly selected individuals from each population and top ranked symbia are used as seeds to reproduce populations for the next generation. The algorithm can find the optimal in most of the 6 department size problems and gives results better than SA, CVGA and NLGA, but worse than the hybrid GA. Authors state that the algorithm can be improved by adding a crossover operator, hence the procedure is promising.

Table A.1 – Summary of the choice of parameters and methods
for the GA proposed by Balakrishnan et al. (2003a)

Operation	Procedure
Encoding	Single string representation
Crossover Operator	Dynamic programming
Fitness Function	The total cost: Sum of flow costs and shifting costs
Selection Scheme	Tournament selection
Mutation	Pairwise exchange heuristic (CRAFT)
Parent Pool	Created using Urban's (1993) method
Replacement	Offspring replaces the worst parent if offspring is unique to pool
Termination	Fixed number of generations

A.2.6 Lower bound methods

Tight lower bounds are required

- To apply the fathoming procedure introduced by Rosenblatt (1986)
- For branch and bound applications
- For testing the effectiveness of heuristic methods when optimal solutions are not available
- Since the QAP cannot be solved in a reasonable amount of time

The first lower bound is proposed by Rosenblatt (1986). He suggests taking the sum of the workflow costs of the optimal layouts for the static problem of each period. This is a valid lower bound since the minimum possible material handling costs are taken and no rearrangement costs are considered. It requires the solution of T QAPs.

Urban (1998) improves this bound by considering two situations separately: either a rearrangement will occur sometime during the planning horizon or it will not. So, Urban finds a lower bound for each of these two cases and takes their minimum in

order to obtain the lower bound. For the case where a rearrangement occurs, then it should affect at least two departments. The lower bound in this case is found by adding the minimum two relayout costs for each period, finding the minimum over the whole planning horizon and adding it to the lower bound suggested by Rosenblatt (1986). For the case where no rearrangement occurs, the workflow cost is simply obtained by solving a static QAP using the sum of the workflows over the entire planning horizon.

Urban (1992) introduces Gilmore-type bounds. To obtain this bound for a single period, the workflows between facilities (w_i) are sorted in monotonically decreasing order and the distances between all pairs of locations (d_i) are sorted in monotonically increasing order. The bound then can be found by taking the inner product of the two vectors, $L = \sum_{i=1}^{n(n-1)/2} w_i d_i$, where n is the number of departments.

The dynamic bound can then be obtained by summing the static lower bound of each period of the planning horizon. This bound is dominated by Rosenblatt's bound (1986) since he uses the optimal static layouts. Another option is to sum up the flow costs of all periods and find the bound in one step, analogous to the upper bound presented by Batta (1987). However, this is obviously dominated by the original method and can be used only for very large problems or when there is little variability of workflow between facilities from one period to the next.

The second type of bound given by Urban (1992) is probabilistic bounds (pbounds). They are found by $G = \mu - k\sigma$, where μ and σ are the mean and standard deviation of the assignment cost distribution. By using a pbound, a risk of identifying a lower bound that is actually greater than the optimal solution is taken. On the other hand, by Tchebycheff's theorem, there is a probability of $\alpha=1/k^2$ that an assignment can be found with a cost less than G. With this information, the actual risk can be manipulated by using appropriate values of k. As k increases, the bound becomes smaller but the risk of exceeding the optimal solution decreases. However, the probability of exceeding the optimal solution is somewhat reduced in the DPLP

since the sums of the static bounds would offset each other. In addition, the rearrangement costs are included in the optimal solution and not in the lower bound, the probability even reduces more.

A.2.7 Upper bound methods

Rosenblatt (1986) suggested using the cost incurred when the initial layout is used during the whole planning horizon. Another approach is to consider using the best layout of each time period during the whole planning horizon and pick the one with the minimum cost.

Batta (1987) showed that if the same layout is used in every period, then the DFLP reduces to an SFLP in which the interdepartmental flow can be determined by adding the flow data for all the periods. So, this "SFLP" would be a feasible solution to the DFLP where no rearrangement costs are incurred, and it can be used as an upper bound to the problem.

Urban (1998) develops an upper bound inspired from the incomplete DP. In the method he proposes, all possible values for the vector showing whether a relayout is made in that period are enumerated. Then, the optimal layouts are found for all consecutive periods whose layouts do not change. An upper bound is obtained when the rearrangement costs are included and the relayout configuration with minimum cost is selected. At first glance, this method seems to find the optimal solution. However, it does not in all cases since "it may be cost effective not to use the layout representing the minimum workflow cost for every subproblem if a higher cost arrangement results in a lower variable rearrangement cost".

Urban (1998) nicely summarizes when each of these three bounds are better to be used. Rosenblatt's method is effective for very low values of rearrangement costs; Batta's method is effective for relatively high rearrangement costs and Urban's method is most effective for low to moderate rearrangement cost values, particularly when the workflow data exhibit high variability over time.

A.3 Formulation and Solution Approaches for UDLP

UDLP applications are few, compared to EDLP. Montreiul was the first to model the single period layout problem as an MIP (see Meller and Gau, 1996a). This model has been modified and used in two approaches developed for DLP, which are the studies by Montreiul and Venkatadri (1991) and Montreiul and Laforge (1992).

Montreiul and Venkatadri (1991) deal with a projection of a future layout and present a study on the intermediate stages to achieve this final layout. Their approach is based on the expansion of the departments, which can be applied for shrinking departments, too. This study is based on "interpolation", instead of extrapolation, which is the case for almost all other DLP applications. In fact, the study is more concerned wth the transition phases rather than the dynamic nature of the problem.

"In Montreiul and Laforge (1992), a DLP is considered when multiple possible future states are possible for each time period. For each state, the analyst must specify its probability of occurrence, department areas and flows, and a design skeleton (spatial arrangement of departments). This model minimizes a weighted average of flow costs and department displacement costs. An analyst experienced in layout and knowledgeable about the facility could use this model to generate good layouts. On the other hand, requiring the analyst to specify design skeletons may lead to unanticipated costs, too."

"Most of the algorithms in dynamic layout either assume equal department sizes, or if they handle different department sizes, require the decision maker to provide skeleton layouts. Lacksonen (1994) proposes a two-stage algorithm that incorporates the advantages of both the formulations discussed above. Stage 1 of his procedure involves solving the equal sized department formulation. The cutting plane and exchange routine analyzed by Lacksonen and Enscore (1993) is used to determine the relative locations of layouts in each period. Departments that are stationary in this stage are required to remain stationary in the second stage also. By setting the relative positions of the layouts, the rearrangement costs are completely defined in

this stage. In Stage 2, these departments are modified to give the various sizes and shapes as required for each period's data by using a mixed integer linear programming model to minimize flow costs." The model used is as follows:

Decision Variables

xu_i, yu_i : Coordinates of the upper right corner of department *i*

xl_i, yl_i : Coordinates of the lower left corner of department *i*

S_{ij} : Separation direction (x-axis or y-axis) of two departments. [It implies that departments are to the right or left of each other (x-axis separation) when it is equal to 1; and that departments are below or above each other (y-axis separation) when it is equal to 0.]

x_{1i}, x_{2i} : Variables corresponding to the right and left distances from the breakpoint in the piecewise linearization of the area requirement function. (i.e. $X_{1i}>0$ implies that x<y, $X_{2i}>0$ implies that x>y and $X_{1i}=X_{2i}=0$ implies that x=y)

x_{ij}^+, x_{ij}^- : The absolute value of the x-axis distance between departments *i* and *j*

y_{ij}^+, y_{ij}^- : The absolute value of the y-axis distance between departments *i* and *j*
(Their sum gives the rectilinear distance between departments *i* and *j*)

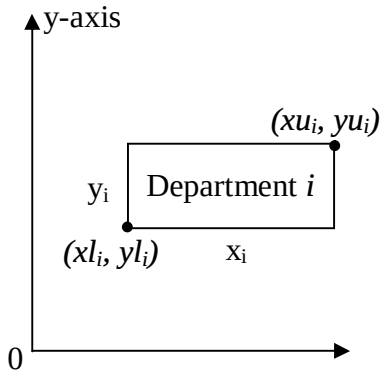


Figure A.4 – Sample department coordinates

Parameters

F_{ij} : Flow cost between departments i and j

A_i : Area requirement of department i

R : The maximum aspect (length/width) ratio allowed for the departments

XUB, YUB : Coordinates that specify the size of the building

M : A large number, which can be assigned *XUB*+*YUB*

xut_i, xlt_i, yut_i, ylt_i : Coordinates of department i in the previous period (defined only for stationary departments i, the departments whose places are not to be changed based on the solution obtained in Stage 1)

$$\textit{Minimize} \quad \sum_{i\in N}\sum_{j\in N} F_{ij}\left(x_{ij}^{+} + x_{ij}^{-} + y_{ij}^{+} + y_{ij}^{-}\right) \tag{A.12}$$

subject to

$\forall i$

$$yu_i - yl_i + R^{0.5}x_{1i} + R^{0.5}x_{2i} = \left(A_i R\right)^{0.5} \tag{A.13}$$

$$xu_i - xl_i - x_{1i} - x_{2i} = \left(A_i / R\right)^{0.5} \tag{A.14}$$

$$x_{1i} \le A_i^{0.5}\left(1 - R^{0.5}\right) \tag{A.15}$$

$$x_{2i} \le A_i^{0.5}\left(R^{0.5} - 1\right) \tag{A.16}$$

$$xu_i \le XUB \tag{A.17}$$

$$yu_i \le YUB \tag{A.18}$$

$\forall (i,j)$ ∋ $F_{ij}>0$

$$xu_i + xl_i - xu_j - xl_j - 2x_{ij}^{+} + 2x_{ij}^{-} = 0 \tag{A.19}$$

$$yu_i + yl_i - yu_j - yl_j - 2y_{ij}^{+} + 2y_{ij}^{-} = 0 \tag{A.20}$$

one of the constraint sets A, B, C or D

one of the constraint sets E, F or G

$\forall i$

$$xu_i, xl_i, yu_i, yl_i \ge 0 \tag{A.21}$$

$\forall (i,j)$

$$x_{1i,}\, x_{2i} \ge 0 \tag{A.22}$$

$$x_{ij}^{+}, x_{ij}^{-}, y_{ij}^{+}, y_{ij}^{-} \ge 0 \tag{A.23}$$

$\forall (i,j)$ ∋ j is below or to the left of i

$$A = \left\{ \begin{matrix} xl_i - xu_j + \mathrm{M}.s_{ij} \ge 0 \\ yl_i - yu_j - \mathrm{M}.s_{ij} \ge -\mathrm{M} \end{matrix} \right\}$$

$\forall (i,j)$ ∋ j is below or to the right of

$$B = \left\{ \begin{matrix} xl_j - xu_i + \mathrm{M}.s_{ij} \ge 0 \\ yl_i - yu_j - \mathrm{M}.\mathrm{s}_{ij} \ge -\mathrm{M} \end{matrix} \right\}$$

$\forall(i,j)\ \ni j$ is to the right of i

$$C = \{xl_j - xu_i \geq 0\}$$

$\forall(i,j)\ \ni j$ is below i

$$D = \{yl_j - yu_i \geq 0\}$$

for stationary depts i that grow

$$E = \begin{Bmatrix} xlt_i - xl_i \geq 0 \\ xu_i - xut_i \geq 0 \\ ylt_i - yl_i \geq 0 \\ yu_i - yut_i \geq 0 \end{Bmatrix}$$

$$F = \begin{Bmatrix} xl_i - xlt_i \geq 0 \\ xut_i - xu_i \geq 0 \\ yl_i - ylt_i \geq 0 \\ yut_i - yu_i \geq 0 \end{Bmatrix}$$

for stationary departments i that shrink

$$G = \begin{Bmatrix} xl_i - xlt_i = 0 \\ xut_i - xu_i = 0 \\ yl_i - ylt_i = 0 \\ yut_i - yu_i = 0 \end{Bmatrix}$$

for stationary departments i that stay the same size

Based on the solution to the equal-sized problem (the relative locations of the departments) solved in Stage 1, the model given in the previous page is solved sequentially for each period. In this model, the objective function, given by equation (A.12), minimizes the total flow costs by multiplying the flow between each department pair with the rectilinear distance between them.

Constraints (A.13)-(A.16) ensure that the area requirements are satisfied with a two-interval piecewise linearization that is better than the ones proposed up till then. The linearization procedure can be summarized as follows: (please refer to the definitions of decision variables and parameters for explanations of A_i, R, x_{1i} and x_{2i})

The curve in Figure A.5 shows all values x_i and y_i can take (x_i and y_i are the horizontal and vertical lengths of department i, which were shown on Figure A.4). X_{max}, Y_{max} and X_{min}, Y_{min} correspond to the maximum and minimum values x_i and y_i can take. X_{med}, Y_{med} are the x_i and y_i values for a square department, which are set as the breakpoint for the piecewise linearization. These values are uniquely determined for each department with the following equations, where A corresponds to the area of the department:

$$Y_{max} = X_{max} = (AR)^{0.5} \quad \text{(A.24)}$$

$$Y_{med} = X_{med} = A^{0.5} \quad \text{(A.25)}$$

$$Y_{min} = X_{min} = (A/R)^{0.5} \quad \text{(A.26)}$$

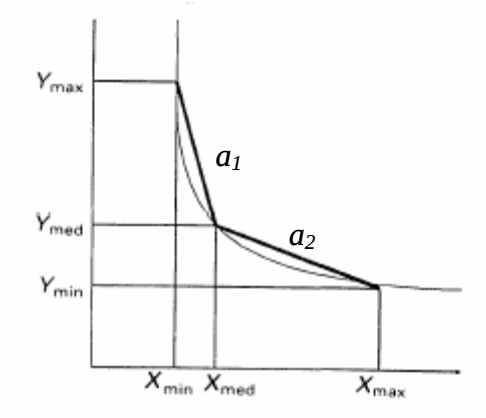


Figure A.5 – Area linearization (Lacksonen, 1994)

The slopes of the linear segments, a_1 and a_2, can be expressed in terms of R:

$$a_1 = -R^{0.5} \quad \text{(A.27)}$$

$$a_2 = -1/R^{0.5} \quad \text{(A.28)}$$

Then, the equations for length and width can be written as follows:

$$yu_i - yl_i = Y_{max} + a_1 x_{1i} + a_2 x_{2i} \quad \text{(A.29)}$$

$$xu_i - xl_i = X_{min} + x_{1i} + x_{2i} \quad \text{(A.30)}$$

$$x_{1i} \leq X_{med} - X_{min} \quad \text{(A.31)}$$

$$x_{2i} \leq X_{max} - X_{med} \quad \text{(A.32)}$$

Finally, by substituting the equations (A.24)-(A.28) into (A.29)-(A.32), one obtains the constraints (A.13)-(A.16). However, it should be noted that only the dynamic layout literature is considered here and better linearizations exist in the SPLP literature.

Constraints (A.17) and (A.18) ensure that each department lies within the building boundaries. Constraints (A.19) and (A.20) calculate the distances between the departments given the coordinates. Constraint set (A.21) is the non-overlapping

constraints and they are used to feed the relative locations obtained from Stage 1 to the model. Based on the relationship, either set A or set B can be used. If constraint set A is used, it implies that department *j* is to the left of department *i* (x-axis separation) when it is equal to 1; and implies that department *j* is below department *i* (y-axis separation) when it is equal to 0.

Sets C and D correspond to sets B and A respectively when the distance between two departments is relatively large (defined as "at least three departments apart in either direction"). Motivation behind using these constraints when applicable is to reduce the number of integer variables in the model.

Constraint set (A.22) is related with stationary departments, whose places are not to be changed based on the solution obtained in Stage 1. However, since the changes in area requirements are allowed, these set of constraints guarantee that if a department grows or shrinks, then the smaller area is enclosed in the larger area. Sets E, F or G can be used based on the direction of the change in the area requirement.
This formulation both offers improved linearizations for non-linear departmental area constraints and improved departmental overlapping constraints over previous ones. Another superiority it provides is making use of the extensive work on EDLP instead of just using "expert opinion".

"In Lacksonen (1997), this procedure is improved in terms of solution time by developing a preprocessing routine. In preprocessing, fixing the integer value fixes the relative positions of certain department pairs. This is done by replacing the non-overlap constraints by those which state that 'far away' departments in Stage 1 maintain the same orientation in Stage 2. Lacksonen (1997) uses three key factors for identifying the certainty that two departments will be separated from each other: estimated stage one orientation, department areas and flows to other departments."

The study of Dunker et al. (2005) is an application of GA to unequal size departments in rectangular shape, where the I/O points are also determined and the distance measures are found accordingly. The algorithm is similar to that of

Balakrishnan et al. (2003a) in the sense that it also employs Dynamic Programming. However, in this paper, the layout for each time period evolves in a different population. Dynamic programming is used to evaluate the fitness values of the layouts (since the fitness of a layout cannot be evaluated without knowing its predecessor and successor).

APPENDIX B

HISTOGRAMS OF THE VALUES BINARY VARIABLES TAKE IN THE LP RELAXATION SOLUTIONS

The following abbreviations are used in Table B.1:

P	Problem instance solved
ε	Accuracy level of the area linearization
VI?	Whether valid inequalities are added or not
wvi	LP relaxation solved with valid inequalities
novi	LP relaxation solved without valid inequalities
% [0] + % [1]	Percent of integer solutions
% [0.50]	Percent of solutions whose value is equal to 0.50
% (x-y]	Percent of solutions between x and y, including solutions whose value is equal to y, but not including solutions whose value is equal to x

Since the histograms are symmetrical with respect to 0.50, values are given only for the intervals with values above 0.50. The symmetry interval of each interval is listed below for convention.

% [0.45-0.50) **ß à** % (0.50-0.55]
% [0.40-0.45) **ß à** % (0.55-0.60]
% [0.35-0.40) **ß à** % (0.60-0.65]
% [0.30-0.35) **ß à** % (0.65-0.70]
% [0.25-0.30) **ß à** % (0.70-0.75]
% [0.20-0.25) **ß à** % (0.75-0.80]
% [0.15-0.20) **ß à** % (0.80-0.85]
% [0.10-0.15) **ß à** % (0.85-0.90]
% [0.05-0.10) **ß à** % (0.90-0.95]
% [0.00-0.05) **ß à** % (0.95-1.00]

Table B.1 – Histograms of LP relaxation solution values

P	ε	10.0%		5.0%		1.0%		0.5%	
	VI?	wvi	novi	wvi	novi	wvi	novi	wvi	Novi
m3	% [0] + % [1]	33.3	-	50.0	-	33.3	-	33.3	-
	% [0.50]	33.3	-	33.3	-	33.3	-	33.3	-
	% (0.50-0.55]	-	-	-	-	-	-	-	-
	% (0.55-0.60]	-	-	-	-	-	-	-	-
	% (0.60-0.65]	-	-	-	16.7	-	-	-	-
	% (0.65-0.70]	-	-	-	-	-	-	-	-
	% (0.70-0.75]	-	-	-	-	-	-	-	-
	% (0.75-0.80]	-	-	-	-	-	-	-	-
	% (0.80-0.85]	-	16.7	-	16.7	-	16.7	-	16.7
	% (0.85-0.90]	-	33.3	-	-	-	33.3	-	33.3
	% (0.90-0.95]	-	-	-	16.7	-	-	-	-
	% (0.95-1.00]	33.3	-	33.3	-	33.3	-	33.3	-
m4	% [0] + % [1]	-	-	8.3	16.7	-	-	-	-
	% [0.50]	66.7	-	66.7	-	66.7	-	66.7	-
	% (0.50-0.55]	-	-	-	-	-	-	-	-
	% (0.55-0.60]	-	-	-	-	-	-	-	-
	% (0.60-0.65]	-	-	-	-	-	-	-	-
	% (0.65-0.70]	-	-	-	4.2	-	-	-	-
	% (0.70-0.75]	-	-	4.2	8.3	-	-	-	-
	% (0.75-0.80]	-	-	-	-	-	-	-	-
	% (0.80-0.85]	-	-	-	-	-	-	-	-
	% (0.85-0.90]	8.3	33.3	8.3	25.0	8.3	33.3	8.3	33.3
	% (0.90-0.95]	8.3	16.7	-	-	8.3	16.7	8.3	16.7
	% (0.95-1.00]	-	-	4.2	12.5	-	-	-	-
m5	% [0] + % [1]	-	10.0	5.0	15.0	-	10.0	-	10.0
	% [0.50]	50.0	-	50.0	-	50.0	-	50.0	-
	% (0.50-0.55]	-	-	-	-	-	-	-	-
	% (0.55-0.60]	-	-	-	-	-	-	-	-
	% (0.60-0.65]	5.0	-	5.0	-	5.0	-	5.0	-
	% (0.65-0.70]	-	2.5	-	2.5	-	2.5	-	2.5
	% (0.70-0.75]	-	2.5	2.5	5.0	-	2.5	-	2.5
	% (0.75-0.80]	-	-	-	2.5	-	-	-	-
	% (0.80-0.85]	-	-	-	-	-	-	-	-
	% (0.85-0.90]	15.0	27.5	15.0	20.0	15.0	27.5	15.0	27.5
	% (0.90-0.95]	5.0	12.5	-	7.5	5.0	12.5	5.0	12.5
	% (0.95-1.00]	-	5.0	2.5	12.5	-	5.0	-	5.0

Table B.1 (continued)

P	ε	10.0%		5.0%		1.0%		0.5%	
	VI?	wvi	novi	wvi	novi	wvi	novi	wvi	novi
m6	% [0] + % [1]	60.0	60.0	56.7	73.3	56.7	56.7	56.7	56.7
	% [0.50]	36.7	-	33.3	-	36.7	-	36.7	-
	% (0.50-0.55]	-	1.7	-	5.0	-	1.7	-	1.7
	% (0.55-0.60]	-	-	1.7	1.7	-	-	-	-
	% (0.60-0.65]	-	-	-	3.3	1.7	-	-	-
	% (0.65-0.70]	-	3.3	-	1.7	-	5.0	-	5.0
	% (0.70-0.75]	1.7	-	-	-	-	-	-	-
	% (0.75-0.80]	-	1.7	-	-	-	-	1.7	1.7
	% (0.80-0.85]	-	1.7	-	-	-	3.3	1.7	3.3
	% (0.85-0.90]	-	-	-	-	-	-	-	-
	% (0.90-0.95]	-	11.7	3.3	1.7	1.7	10.0	-	8.3
	% (0.95-1.00]	30.0	30.0	28.3	36.7	28.3	30.0	28.3	30.0
m7	% [0] + % [1]	28.6	19.0	31.0	28.6	33.3	28.6	31.0	26.2
	% [0.50]	40.5	-	38.1	-	38.1	-	42.9	-
	% (0.50-0.55]	2.4	-	-	-	-	3.6	2.4	2.4
	% (0.55-0.60]	-	2.4	1.2	4.8	1.2	3.6	4.8	3.6
	% (0.60-0.65]	-	-	3.6	3.6	6.0	-	-	1.2
	% (0.65-0.70]	3.6	8.3	1.2	4.8	-	6.0	-	1.2
	% (0.70-0.75]	6.0	4.8	3.6	2.4	1.2	2.4	1.2	7.1
	% (0.75-0.80]	1.2	-	1.2	3.6	1.2	1.2	3.6	4.8
	% (0.80-0.85]	1.2	10.7	2.4	1.2	1.2	2.4	-	2.4
	% (0.85-0.90]	-	3.6	-	1.2	1.2	8.3	1.2	4.8
	% (0.90-0.95]	-	1.2	-	1.2	1.2	1.2	-	4.8
	% (0.95-1.00]	15.5	19.0	17.9	27.4	17.9	21.4	15.5	17.9
o7	% [0] + % [1]	38.1	35.7	26.2	21.4	45.2	42.9	26.2	40.5
	% [0.50]	38.1	-	38.1	-	38.1	-	38.1	-
	% (0.50-0.55]	-	3.6	3.6	-	-	-	1.2	2.4
	% (0.55-0.60]	4.8	7.1	1.2	-	-	-	-	3.6
	% (0.60-0.65]	-	3.6	-	-	-	-	-	3.6
	% (0.65-0.70]	-	-	-	-	1.2	-	-	6.0
	% (0.70-0.75]	3.6	-	1.2	7.1	-	7.1	3.6	2.4
	% (0.75-0.80]	-	3.6	-	4.8	6.0	14.3	-	6.0
	% (0.80-0.85]	1.2	3.6	7.1	13.1	1.2	3.6	7.1	1.2
	% (0.85-0.90]	1.2	6.0	3.6	-	-	-	1.2	-
	% (0.90-0.95]	-	1.2	-	2.4	-	-	2.4	3.6
	% (0.95-1.00]	20.2	21.4	14.3	22.6	22.6	25.0	15.5	21.4

Table B.1 (continued)

P	E	10.0%		5.0%		1.0%		0.5%	
	VI?	wvi	novi	wvi	novi	wvi	novi	wvi	novi
o8	% [0] + % [1]	33.9	44.6	23.2	23.2	39.3	39.3	25.0	42.9
	% [0.50]	44.6	-	42.9	-	44.6	-	42.9	-
	% (0.50-0.55]	-	2.7	2.7	-	-	-	1.8	10.7
	% (0.55-0.60]	4.5	5.4	1.8	-	0.9	-	1.8	5.4
	% (0.60-0.65]	0.9	8.0	0.9	-	-	-	2.7	6.3
	% (0.65-0.70]	0.9	-	2.7	0.9	0.9	-	0.9	2.7
	% (0.70-0.75]	0.9	-	0.9	2.7	-	-	0.9	-
	% (0.75-0.80]	0.9	3.6	-	6.3	-	7.1	3.6	-
	% (0.80-0.85]	-	6.3	-	7.1	5.4	10.7	0.9	-
	% (0.85-0.90]	1.8	-	-	9.8	0.9	7.1	3.6	-
	% (0.90-0.95]	0.9	1.8	8.0	1.8	-	2.7	-	-
	% (0.95-1.00]	17.0	22.3	11.6	21.4	19.6	22.3	12.5	25.0
o9	% [0] + % [1]	30.6	47.2	18.1	43.1	26.4	34.7	19.4	33.3
	% [0.50]	52.8	-	48.6	-	47.2	-	48.6	-
	% (0.50-0.55]	-	4.9	-	-	0.7	-	-	2.8
	% (0.55-0.60]	0.7	2.1	2.8	5.6	-	-	2.8	5.6
	% (0.60-0.65]	2.8	5.6	-	7.6	1.4	-	-	2.8
	% (0.65-0.70]	2.8	-	4.9	-	0.7	-	4.9	5.6
	% (0.70-0.75]	-	5.6	2.8	10.4	-	-	2.8	3.5
	% (0.75-0.80]	-	-	-	-	-	2.8	-	-
	% (0.80-0.85]	-	4.2	4.2	-	1.4	2.8	4.2	4.2
	% (0.85-0.90]	2.1	-	1.4	1.4	3.5	13.2	1.4	0.7
	% (0.90-0.95]	-	2.1	-	1.4	2.1	6.9	-	1.4
	% (0.95-1.00]	15.3	25.7	9.7	23.6	16.7	24.3	9.7	23.6
o10	% [0] + % [1]	21.1	36.7	21.1	32.2	41.1	24.4	18.9	27.8
	% [0.50]	37.8	-	33.3	-	34.4	-	35.6	-
	% (0.50-0.55]	2.2	3.3	4.4	5.0	-	-	5.0	1.1
	% (0.55-0.60]	-	1.1	2.8	1.7	-	-	1.7	3.9
	% (0.60-0.65]	5.6	1.1	1.1	1.7	-	-	2.8	-
	% (0.65-0.70]	3.3	1.7	4.4	4.4	1.1	-	0.6	1.7
	% (0.70-0.75]	1.7	1.7	2.2	10.6	-	-	2.8	17.8
	% (0.75-0.80]	-	7.2	-	2.8	-	-	5.0	0.6
	% (0.80-0.85]	1.1	0.6	1.1	2.8	3.3	6.1	1.1	3.3
	% (0.85-0.90]	3.3	12.2	2.8	-	7.8	13.9	2.2	1.1
	% (0.90-0.95]	2.2	2.8	2.8	0.6	-	10.0	0.6	1.7
	% (0.95-1.00]	11.7	18.3	11.7	20.6	20.6	20.0	10.6	18.9

Table B.1 (continued)

P	ε	10.0%		5.0%		1.0%		0.5%	
	VI?	wvi	novi	wvi	novi	wvi	novi	wvi	Novi
fo7	% [0] + % [1]	42.9	38.1	33.3	35.7	66.7	42.9	35.7	38.1
	% [0.50]	23.8	-	26.2	-	23.8	-	26.2	-
	% (0.50-0.55]	-	2.4	1.2	6.0	-	-	3.6	3.6
	% (0.55-0.60]	-	10.7	10.7	7.1	-	-	8.3	1.2
	% (0.60-0.65]	1.2	1.2	1.2	8.3	1.2	-	1.2	2.4
	% (0.65-0.70]	2.4	-	-	-	2.4	-	-	4.8
	% (0.70-0.75]	9.5	1.2	-	3.6	-	7.1	-	4.8
	% (0.75-0.80]	-	2.4	-	-	-	14.3	-	4.8
	% (0.80-0.85]	1.2	6.0	-	-	1.2	3.6	-	2.4
	% (0.85-0.90]	-	3.6	6.0	-	-	-	4.8	1.2
	% (0.90-0.95]	2.4	1.2	-	-	-	-	1.2	3.6
	% (0.95-1.00]	21.4	21.4	17.9	25.0	33.3	25.0	17.9	21.4
fo8	% [0] + % [1]	37.5	48.2	35.7	46.4	67.9	48.2	28.6	48.2
	% [0.50]	23.2	-	25.0	-	21.4	-	26.8	-
	% (0.50-0.55]	-	2.7	1.8	2.7	-	-	3.6	-
	% (0.55-0.60]	-	5.4	0.9	8.9	-	-	-	13.4
	% (0.60-0.65]	0.9	2.7	11.6	9.8	-	-	-	10.7
	% (0.65-0.70]	1.8	6.3	-	3.6	1.8	-	9.8	-
	% (0.70-0.75]	0.9	-	0.9	-	-	-	2.7	-
	% (0.75-0.80]	0.9	5.4	-	-	1.8	5.4	3.6	0.9
	% (0.80-0.85]	11.6	-	-	-	0.9	17.0	0.9	-
	% (0.85-0.90]	0.9	2.7	3.6	-	0.9	2.7	-	-
	% (0.90-0.95]	2.7	-	-	0.9	-	0.9	-	0.9
	% (0.95-1.00]	18.8	25.0	18.8	24.1	33.9	24.1	16.1	24.1
fo9	% [0] + % [1]	25.0	43.1	20.8	45.8	68.1	38.9	20.8	45.8
	% [0.50]	23.6	-	36.1	-	19.4	-	33.3	-
	% (0.50-0.55]	3.5	4.2	4.2	-	-	-	3.5	-
	% (0.55-0.60]	7.6	4.2	4.9	6.3	-	-	7.6	6.9
	% (0.60-0.65]	6.3	2.1	0.7	6.9	-	-	0.7	4.2
	% (0.65-0.70]	2.1	-	3.5	8.3	2.1	-	4.2	10.4
	% (0.70-0.75]	-	9.0	4.2	1.4	-	-	2.1	1.4
	% (0.75-0.80]	-	6.9	-	1.4	1.4	4.9	-	1.4
	% (0.80-0.85]	-	-	1.4	0.7	0.7	4.9	3.5	0.7
	% (0.85-0.90]	0.7	-	-	-	2.1	15.3	-	-
	% (0.90-0.95]	4.9	1.4	2.8	1.4	-	3.5	1.4	1.4
	% (0.95-1.00]	13.2	22.2	10.4	23.6	34.0	21.5	10.4	23.6

Table B.1 (continued)

P	ε	10.0%		5.0%		1.0%		0.5%	
	VI?	wvi	novi	wvi	novi	wvi	novi	wvi	Novi
fo10	% [0] + % [1]	26.7	37.8	30.0	48.9	67.8	36.7	27.8	50.0
	% [0.50]	21.1	-	22.2	-	17.8	-	23.3	-
	% (0.50-0.55]	4.4	2.2	3.9	4.4	-	-	2.2	5.6
	% (0.55-0.60]	0.6	4.4	1.7	4.4	-	-	5.0	3.9
	% (0.60-0.65]	5.6	1.1	2.8	3.3	-	-	6.7	2.2
	% (0.65-0.70]	3.3	2.2	8.3	4.4	2.2	-	5.0	3.9
	% (0.70-0.75]	3.3	2.8	4.4	5.0	-	-	1.1	4.4
	% (0.75-0.80]	1.1	5.6	0.6	2.8	1.1	-	0.6	4.4
	% (0.80-0.85]	-	10.0	-	0.6	0.6	8.3	1.7	0.6
	% (0.85-0.90]	3.3	-	-	-	3.3	17.2	1.7	-
	% (0.90-0.95]	1.7	-	-	-	-	1.1	0.6	-
	% (0.95-1.00]	16.1	21.7	17.2	25.0	33.9	23.3	13.9	25.0
fo11	% [0] + % [1]	30.9	26.4	30.0	19.1	67.3	33.6	30.9	24.5
	% [0.50]	16.4	-	19.1	-	16.4	-	20.0	-
	% (0.50-0.55]	1.4	1.4	2.3	2.7	-	-	-	4.1
	% (0.55-0.60]	-	-	1.8	2.7	-	-	0.9	-
	% (0.60-0.65]	2.3	1.8	-	0.9	2.3	-	3.2	1.4
	% (0.65-0.70]	3.2	2.3	0.9	1.8	0.9	-	0.9	1.8
	% (0.70-0.75]	0.9	3.2	5.9	0.9	-	-	6.8	4.1
	% (0.75-0.80]	2.7	0.5	5.9	11.4	-	-	4.1	8.2
	% (0.80-0.85]	0.9	8.2	5.5	4.1	-	5.9	6.4	8.6
	% (0.85-0.90]	11.8	11.4	0.9	3.2	5.0	16.4	-	-
	% (0.90-0.95]	0.5	2.3	1.4	3.2	-	5.5	1.4	5.0
	% (0.95-1.00]	18.2	19.1	15.9	19.1	33.6	22.3	16.4	16.8
vc10	% [0] + % [1]	7.8	13.3	11.1	21.1	6.7	22.2	10.0	16.7
	% [0.50]	26.7	-	21.1	-	20.0	-	28.9	-
	% (0.50-0.55]	7.8	1.1	2.2	7.2	3.9	6.7	8.3	5.0
	% (0.55-0.60]	5.0	9.4	4.4	-	2.2	5.0	3.3	2.8
	% (0.60-0.65]	1.7	4.4	5.0	3.9	3.3	-	0.6	6.1
	% (0.65-0.70]	0.6	2.2	0.6	6.1	5.6	1.1	2.2	0.6
	% (0.70-0.75]	-	4.4	6.1	0.6	3.9	3.9	1.1	3.3
	% (0.75-0.80]	0.6	-	1.7	2.2	5.0	1.7	0.6	2.2
	% (0.80-0.85]	2.8	-	1.1	2.8	2.8	8.3	3.3	8.9
	% (0.85-0.90]	1.7	5.6	5.6	9.4	2.8	5.6	0.6	3.3
	% (0.90-0.95]	6.7	9.4	6.1	1.1	4.4	0.6	5.0	2.2
	% (0.95-1.00]	10.0	13.3	6.7	16.7	6.1	17.2	10.6	15.6

Table B.1 (continued)

P	E	10.0%		5.0%		1.0%		0.5%	
	VI?	wvi	novi	wvi	novi	wvi	novi	wvi	Novi
ab20	% [0] + % [1]	8.2	11.6	5.5	11.1	8.4	5.5	9.2	13.7
	% [0.50]	38.9	38.9	38.9	38.9	38.9	38.9	38.9	38.9
	% (0.50-0.55]	2.8	2.8	2.8	2.8	2.8	2.8	2.8	2.8
	% (0.55-0.60]	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4
	% (0.60-0.65]	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7
	% (0.65-0.70]	2.5	2.5	2.5	2.5	2.5	2.5	2.5	2.5
	% (0.70-0.75]	4.9	4.9	4.9	4.9	4.9	4.9	4.9	4.9
	% (0.75-0.80]	4.2	4.2	4.2	4.2	4.2	4.2	4.2	4.2
	% (0.80-0.85]	2.6	2.6	2.6	2.6	2.6	2.6	2.6	2.6
	% (0.85-0.90]	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9
	% (0.90-0.95]	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0
	% (0.95-1.00]	6.6	6.6	6.6	6.6	6.6	6.6	6.6	6.6
AVG	% [0] + % [1]	26.5	29.5	25.4	30.1	39.3	29.0	23.3	29.6
	% [0.50]	35.9	2.4	35.8	2.4	34.2	2.4	37.0	2.4
	% (0.50-0.55]	1.5	2.0	1.8	2.2	0.5	0.9	2.1	2.6
	% (0.55-0.60]	1.6	3.4	2.4	2.8	0.4	0.7	2.4	3.3
	% (0.60-0.65]	2.0	2.0	2.0	4.2	1.3	-	1.5	2.6
	% (0.65-0.70]	1.6	2.0	1.8	2.8	1.3	1.1	1.9	3.0
	% (0.70-0.75]	2.1	2.5	2.7	3.9	0.6	1.7	1.9	3.5
	% (0.75-0.80]	0.7	2.6	0.8	2.6	1.3	3.5	1.7	2.4
	% (0.80-0.85]	1.4	4.4	1.6	3.2	1.3	6.0	2.1	3.4
	% (0.85-0.90]	3.2	8.8	3.0	4.4	3.2	12.2	2.6	6.6
	% (0.90-0.95]	2.4	4.2	1.7	2.7	1.6	4.7	1.8	4.1
	% (0.95-1.00]	15.5	16.9	14.2	19.8	21.3	18.0	13.6	17.2

APPENDIX C

RESULTS OBTAINED FOR LP ROUNDING HEURISTIC

The following abbreviations are used in Tables C.1-C.16:

ε	Accuracy level of the area linearization
#ABV	Number of binary variables used in the model averaged for the instances in question
IGB	Averages of the gaps of the best known solutions from the LP relaxation solution value for the problems under consideration
t	Threshold value
VI?	How valid inequalities are treated
INT0 (%)	Percent of integer solutions at the initial LP relaxation solution
#ITER	Number of iterations performed in rounding heuristic
FIX/ITER (#)	Number of variables fixed per iteration
FIX/ITER (%)	Percent of variables fixed per iteration
FIX (%)	Total percent of variables fixed at rounding heuristic iterations
CPU_LP	Runtime of the rounding heuristic in CPU seconds
#NODE	Number of nodes analyzed at the truncated branch and bound step
CPU_TBB	Runtime of the truncated branch and bound in CPU seconds
CPU_Σ	Total runtime of the heuristic in CPU seconds
IGAP (%)	Gap of the solution found with respect to the optimal solution value
LPGAP (%)	Gap of the solution found with respect to the LP relaxation solution value
CPUIMP (%)	Improvement over the CPU time required to solve the problem to optimality
#LP-INF	Number of infeasible instances detected during rounding heuristic
#TBB-INF	Number of infeasible instances detected during truncated branch and bound
#INF (over x)	Total # of infeasible instances, x being the number of instances under consideration

Table C.1 – Fixing multiples, fo7, fo8, fo9, fo10, fo11, o7, o8, o9 (optimal solutions known), ε=10%, #ABV=135

t	VI?	INT0 (%)	#ITER	FIX/ITER (%)	FIX (%)	CPU_LP	#NODE	CPU_TBB	CPU_Σ	IGAP (%)	CPUIMP (%)	#LP-INF	#TBB-INF	#INF (over 8)
0.75	**novi**	40.1	12.1	3.7	82.4	0.23	3794.8	0.78	0.82	13.3	100.0	2	5	7
	vimip	40.1	12.1	3.7	82.4	0.23	3733.5	1.45	0.41	13.3	100.0	2	5	7
	wvi	40.1	11.6	3.8	81.5	0.23	4105.4	0.69	0.83	13.3	100.0	1	6	7
0.80	**novi**	40.1	11.8	3.6	80.5	0.22	3126.9	0.75	0.88	13.3	100.0	1	6	7
	vimip	40.1	11.8	3.6	80.5	0.22	3074.3	1.44	0.40	13.3	100.0	1	6	7
	wvi	40.1	11.3	3.7	79.6	0.22	3038.0	0.68	0.89	13.3	100.0	0	7	7
0.85	**novi**	40.1	6.5	3.8	62.5	0.14	1316.8	4.50	4.64	18.9	99.6	0	3	3
	vimip	40.1	6.5	3.8	62.5	0.14	1348.9	2.81	1.90	18.9	99.8	0	3	3
	wvi	40.1	6.3	3.8	61.6	0.13	1221.0	4.61	4.74	27.5	99.7	0	2	2
0.90	**novi**	40.1	3.5	3.1	48.3	0.09	1946.5	8.16	8.24	20.0	99.3	0	0	0
	vimip	40.1	3.5	3.1	48.3	0.09	1720.4	2.57	2.66	20.0	99.7	0	0	0
	wvi	40.1	3.5	3.1	48.3	0.09	1946.5	8.21	8.29	20.0	99.2	0	0	0
0.95	**novi**	40.1	2.5	3.4	45.2	0.07	4673.6	23.95	24.02	18.8	99.1	0	0	0
	vimip	40.1	2.5	3.4	45.2	0.07	3602.8	5.14	5.21	18.8	99.6	0	0	0
	wvi	40.1	2.5	3.4	45.2	0.07	4673.6	24.01	24.08	18.8	99.1	0	0	0

Table C.2 – Fixing multiples, fo7, o10, vc10, ab20 (optimal solutions not known), ε=10%, #ABV=373, IGB=1960%

t	VI?	INT0 (%)	#ITER	FIX/ITER (%)	FIX (%)	CPU_LP	#NODE	CPU_TBB	CPU_Σ	LPGAP (%)	#LP-INF	#TBB-INF	#INF (over 3)
0.75	**novi**	20.5	49.3	2.2	95.9	7.08	390.0	0.16	7.14	NA	2	1	3
	vimip	20.5	49.3	2.2	95.9	7.08	390.0	NA	7.08	NA	2	1	3
	wvi	20.5	49.3	2.2	95.9	7.08	662.0	0.17	7.14	NA	2	1	3
0.80	**novi**	20.5	35.0	2.2	77.2	4.52	61481.7	4.08	8.60	1015.0	0	2	2
	vimip	20.5	35.0	2.2	77.2	4.52	61506.3	0.97	4.84	1015.0	0	2	2
	wvi	20.5	35.0	2.2	77.2	4.52	72034.7	4.08	8.60	1015.0	0	2	2
0.85	**novi**	20.5	27.7	2.6	64.7	4.13	9492.7	154.46	158.59	680.9	0	2	2
	vimip	20.5	27.7	2.6	64.7	4.13	8689.0	38.73	17.04	680.9	0	2	2
	wvi	20.5	27.7	2.6	64.7	4.11	9420.7	307.93	312.04	680.9	0	2	2
0.90	**novi**	20.5	17.7	2.3	42.3	3.40	53917.3	2565.99	2569.39	1773.3	0	0	0
	vimip	20.5	17.7	2.3	42.3	3.40	130376.3	2427.29	2430.68	1720.5	0	0	0
	wvi	20.5	17.7	2.3	42.3	3.44	54003.7	2567.21	2570.66	1773.3	0	0	0
0.95	**novi**	20.5	14.0	1.9	38.4	2.50	85635.7	2693.47	2695.97	1776.9	0	0	0
	vimip	20.5	14.0	1.9	38.4	2.50	139998.7	2453.62	2456.12	1732.1	0	0	0
	wvi	20.5	14.0	1.9	38.4	2.53	85559.0	2694.66	2697.19	1776.9	0	0	0

Table C.3 – Fixing singles, fo7, fo8, fo9, fo10, fo11, o7, o8, o9 (optimal solutions known), ε=10%, #ABV=135

t	VI?	INT0 (%)	#ITER	FIX/ITER (#)	FIX (%)	CPU_LP	#NODE	CPU_TBB	CPU_Σ	IGAP (%)	CPUIMP (%)	#LP-INF	#TBB-INF	#INF (over 8)
0.75	**novi**	40.1	34.0	2.0	85.0	0.62	2502.0	1.08	1.16	13.3	100.0	4	3	7
	vimip	40.1	34.0	2.0	85.0	0.62	2410.0	1.45	0.80	13.3	100.0	4	3	7
	wvi	40.1	34.0	2.0	85.0	0.61	3228.3	1.09	1.15	13.3	100.0	4	3	7
0.80	**novi**	40.1	31.9	2.0	81.7	0.57	2841.7	0.73	1.21	13.3	100.0	1	6	7
	vimip	40.1	31.9	2.0	81.7	0.57	2789.1	1.44	0.75	13.3	100.0	1	6	7
	wvi	40.1	31.9	2.0	81.7	0.58	3975.3	0.74	1.22	13.3	100.0	1	6	7
0.85	**novi**	40.1	18.9	2.0	62.5	0.38	1320.3	4.74	5.12	18.9	99.6	0	3	3
	vimip	40.1	18.9	2.0	62.5	0.38	1348.9	2.82	2.14	18.9	99.8	0	3	3
	wvi	40.1	18.9	2.0	62.5	0.37	1302.8	4.77	5.13	18.9	99.6	0	3	3
0.90	**novi**	40.1	7.3	2.0	48.3	0.17	1943.0	8.12	8.29	20.0	99.2	0	0	0
	vimip	40.1	7.3	2.0	48.3	0.17	1720.4	2.54	2.71	20.0	99.6	0	0	0
	wvi	40.1	7.3	2.0	48.3	0.15	1943.0	8.15	8.31	20.0	99.3	0	0	0
0.95	**novi**	40.1	4.9	2.0	45.2	0.12	4414.0	18.55	18.67	18.8	99.1	0	0	0
	vimip	40.1	4.9	2.0	45.2	0.12	3602.8	5.12	5.23	18.8	99.6	0	0	0
	wvi	40.1	4.9	2.0	45.2	0.12	4414.0	18.47	18.59	18.8	99.1	0	0	0

Table C.4 – Fixing singles, o10, vc10, ab20 (optimal solutions not known), ε=10%, #ABV=373, IGB=1960%

t	VI?	INT0 (%)	#ITER	FIX/ITER (#)	FIX (%)	CPU_LP	#NODE	CPU_TBB	CPU_Σ	LPGAP (%)	#LP-INF	#TBB-INF	#INF (over 3)
0.75	**novi**	20.5	149.3	2.0	95.5	17.59	100726.0	0.05	17.61	NA	2	1	3
	vimip	20.5	149.3	2.0	95.5	17.59	100726.0	NA	17.59	NA	2	1	3
	wvi	20.5	149.3	2.0	95.5	17.58	37737.0	0.05	17.59	NA	2	1	3
0.80	**novi**	20.5	134.7	2.0	79.2	17.28	13083.0	34.76	40.45	1060.2	1	1	2
	vimip	20.5	134.7	2.0	79.2	17.28	7795.0	13.30	21.71	1060.2	1	1	2
	wvi	20.5	134.7	2.0	79.2	17.32	13839.0	35.29	40.85	1060.2	1	1	2
0.85	**novi**	20.5	129.0	2.0	77.7	16.78	7036.3	24.56	41.34	1060.2	0	2	2
	vimip	20.5	129.0	2.0	77.7	16.78	3511.0	14.20	21.51	1060.2	0	2	2
	wvi	20.5	134.3	2.0	79.1	17.15	7050.0	23.96	41.12	1060.2	0	2	2
0.90	**novi**	20.5	81.7	2.0	48.3	12.14	28987.0	2546.69	2558.83	795.9	0	1	1
	vimip	20.5	81.7	2.0	48.3	12.14	18749.7	40.96	39.44	795.9	0	1	1
	wvi	20.5	80.7	2.0	48.0	11.76	28992.0	2545.02	2556.78	795.9	0	1	1
0.95	**novi**	20.5	53.0	2.0	39.0	7.93	81879.0	2691.34	2699.28	1905.0	0	0	0
	vimip	20.5	53.0	2.0	39.0	7.93	121687.7	2453.60	2461.54	1742.1	0	0	0
	wvi	20.5	53.0	2.0	39.0	7.96	81896.7	2694.36	2702.33	1905.0	0	0	0

Table C.5 – Fixing multiples, o7, o8, o9, fo7, fo8, fo9 (optimal solutions known), ε=5%, #ABV=113

t	VI?	INT0 (%)	#ITER	FIX/ITER (%)	FIX (%)	CPU_LP	#NODE	CPU_TBB	CPU_Σ	IGAP (%)	CPUIMP (%)	#LP-INF	#TBB-INF	#INF (over 6)
0.75	**novi**	35.9	13.5	4.0	76.2	0.21	371.7	1.02	0.72	24.6	98.6	2	0	2
	vimip	35.9	13.5	4.0	76.2	0.21	346.0	0.54	0.39	24.6	99.4	2	0	2
	Wvi	35.9	13.5	4.0	76.2	0.22	371.3	1.02	0.73	24.6	98.6	2	0	2
0.80	**novi**	35.9	9.5	4.0	64.7	0.15	970.3	2.36	2.52	22.9	99.3	0	1	1
	vimip	35.9	9.5	4.0	64.7	0.15	795.7	1.04	0.85	22.9	99.7	0	1	1
	Wvi	35.9	9.5	4.0	64.7	0.15	1105.3	2.41	2.55	22.9	99.3	0	1	1
0.85	**novi**	35.9	4.8	4.0	51.0	0.09	1090.7	3.81	3.89	22.3	99.5	0	0	0
	vimip	35.9	4.8	4.0	51.0	0.09	888.3	1.26	1.34	22.3	99.7	0	0	0
	Wvi	35.9	4.8	4.0	51.0	0.08	1090.7	3.84	3.92	22.3	99.5	0	0	0
0.90	**novi**	35.9	3.8	4.4	47.5	0.07	3236.2	11.71	11.78	20.3	99.4	0	0	0
	vimip	35.9	3.8	4.4	47.5	0.07	2284.8	3.37	3.44	20.3	99.7	0	0	0
	Wvi	35.9	3.8	4.4	47.5	0.08	3236.2	11.79	11.87	20.3	99.4	0	0	0
0.95	**novi**	35.9	3.2	4.7	46.6	0.06	4832.5	16.38	16.45	20.3	99.2	0	0	0
	vimip	35.9	3.2	4.7	46.6	0.06	4350.0	6.39	6.45	20.3	99.7	0	0	0
	Wvi	35.9	3.2	4.7	46.6	0.06	4832.5	16.48	16.55	20.3	99.2	0	0	0

Table C.6 – Fixing multiples, o10, fo10, fo11, vc10, ab20 (optimal solutions not known), ε=5%, #ABV=304, IGB=1471%

t	VI?	INT0 (%)	#ITER	FIX/ITER (%)	FIX (%)	CPU_LP	#NODE	CPU_TBB	CPU_Σ	LPGAP (%)	#LP-INF	#TBB-INF	#INF (over 5)
0.75	**novi**	26.5	36.6	2.6	94.6	4.74	5088.3	1.18	5.45	NA	2	3	5
	vimip	26.5	36.6	2.6	94.6	4.74	5088.3	NA	4.74	NA	2	3	5
	wvi	26.5	36.6	2.6	94.6	4.74	85997.0	1.19	5.45	NA	2	3	5
0.80	**novi**	26.5	26.4	3.1	78.3	3.34	39900.2	309.67	313.01	2458.5	0	2	2
	vimip	26.5	26.4	3.1	78.3	3.34	32686.4	84.36	53.96	2458.5	0	2	2
	wvi	26.5	26.6	3.1	78.3	3.35	39680.6	312.34	315.69	2458.5	0	2	2
0.85	**novi**	26.5	17.6	3.0	52.4	3.10	49841.0	594.70	597.80	1689.4	0	0	0
	vimip	26.5	17.6	3.0	52.4	3.10	38038.8	72.48	75.58	1689.4	0	0	0
	wvi	26.5	17.6	3.0	52.4	3.10	49727.8	598.27	601.37	1689.4	0	0	0
0.90	**novi**	26.5	14.6	2.9	48.6	2.57	41835.0	1706.25	1708.82	969.4	0	1	1
	vimip	26.5	14.6	2.9	48.6	2.57	32847.0	54.18	45.92	969.4	0	1	1
	wvi	26.5	14.6	2.9	48.6	2.57	41839.6	1707.02	1709.59	969.4	0	1	1
0.95	**novi**	26.5	10.4	3.1	44.9	1.77	60198.6	1719.03	1720.80	1592.2	0	0	0
	vimip	26.5	10.4	3.1	44.9	1.77	83680.6	1486.39	1488.16	1519.5	0	0	0
	wvi	26.5	10.4	3.1	44.9	1.72	60201.6	1718.62	1720.34	1592.2	0	0	0

Table C.7 – Fixing singles, o7, o8, o9, fo7, fo8, fo9 (optimal solutions known), ε=5%, #ABV=113

t	VI?	INT0 (%)	#ITER	FIX/ITER (#)	FIX (%)	CPU_LP	#NODE	CPU_TBB	CPU_Σ	IGAP (%)	CPUIMP (%)	#LP-INF	#TBB-INF	#INF (over 6)
0.75	**novi**	35.9	25.0	2.0	77.6	0.35	2066.4	0.58	0.84	24.6	98.7	1	3	4
	vimip	35.9	25.0	2.0	77.6	0.35	2060.2	0.52	0.53	24.6	99.4	1	3	4
	wvi	35.9	25.0	2.0	77.6	0.33	2745.2	0.59	0.82	24.6	98.7	1	3	4
0.80	**novi**	35.9	17.0	2.0	63.2	0.24	882.2	2.25	2.11	23.3	99.3	1	1	2
	vimip	35.9	17.0	2.0	63.2	0.24	797.4	1.03	0.92	23.3	99.7	1	1	2
	wvi	35.9	17.0	2.0	63.2	0.24	989.8	2.23	2.10	23.3	99.3	1	1	2
0.85	**novi**	35.9	8.5	2.0	50.5	0.12	1687.5	5.34	5.47	22.3	99.5	0	0	0
	vimip	35.9	8.5	2.0	50.5	0.12	1529.3	2.19	2.31	22.3	99.7	0	0	0
	wvi	35.9	8.5	2.0	50.5	0.12	1687.5	5.32	5.44	22.3	99.5	0	0	0
0.90	**novi**	35.9	7.0	2.0	47.5	0.11	3164.7	8.41	8.52	20.3	99.4	0	0	0
	vimip	35.9	7.0	2.0	47.5	0.11	2284.8	3.31	3.42	20.3	99.7	0	0	0
	wvi	35.9	7.0	2.0	47.5	0.10	3164.7	8.32	8.42	20.3	99.5	0	0	0
0.95	**novi**	35.9	6.3	2.0	46.6	0.10	5317.0	18.30	18.40	20.3	99.2	0	0	0
	vimip	35.9	6.3	2.0	46.6	0.10	4350.0	6.33	6.43	20.3	99.7	0	0	0
	wvi	35.9	6.3	2.0	46.6	0.09	5317.0	18.29	18.38	20.3	99.2	0	0	0

Table C.8 – Fixing singles, o10, fo10, fo11, vc10, ab20 (optimal solutions not known), ε=5%, #ABV=304, IGB=1471%

t	VI?	INT0 (%)	#ITER	FIX/ITER (#)	FIX (%)	CPU_LP	#NODE	CPU_TBB	CPU_Σ	LPGAP (%)	#LP-INF	#TBB-INF	#INF (over 5)
0.75	**novi**	26.5	111.8	1.9	93.8	11.37	2490.0	0.14	11.42	NA	3	2	5
	vimip	26.5	111.8	1.9	93.8	11.37	2490.0	NA	11.37	NA	3	2	5
	wvi	26.5	111.8	1.9	93.8	11.13	3877.5	0.14	11.19	NA	3	2	5
0.80	**novi**	26.5	101.6	2.0	82.5	10.96	50308.0	303.27	253.57	1314.4	1	2	3
	vimip	26.5	101.6	2.0	82.5	10.96	37903.0	92.15	47.82	1314.4	1	2	3
	wvi	26.5	101.6	2.0	82.5	10.93	51558.8	304.32	254.38	1314.4	1	2	3
0.85	**novi**	26.5	70.2	2.0	54.3	9.58	43049.8	281.55	291.13	969.4	0	1	1
	vimip	26.5	70.2	2.0	54.3	9.58	33098.2	55.69	54.14	969.4	0	1	1
	wvi	26.5	70.2	2.0	54.3	9.58	43052.8	278.89	288.47	969.4	0	1	1
0.90	**novi**	26.5	45.6	2.0	47.9	5.89	59390.8	1716.63	1722.52	1695.5	0	0	0
	vimip	26.5	45.6	2.0	47.9	5.89	106478.2	1483.90	1489.79	1629.1	0	0	0
	wvi	26.5	45.6	2.0	47.9	5.89	59354.2	1716.34	1722.23	1695.5	0	0	0
0.95	**novi**	26.5	37.4	2.0	43.8	4.88	82261.8	1883.75	1888.63	1618.7	0	0	0
	vimip	26.5	37.4	2.0	43.8	4.88	95390.2	1507.78	1512.66	1597.8	0	0	0
	wvi	26.5	37.4	2.0	43.8	4.88	82213.0	1883.73	1888.61	1618.7	0	0	0

Table C.9 – Fixing multiples, fo7, fo8, fo9, fo11, o7, o8, o9 (optimal solutions known), ε=1%, #ABV=129

t	VI?	INT0 (%)	#ITER	FIX/ITER (%)	FIX (%)	CPU_LP	#NODE	CPU_TBB	CPU_Σ	IGAP (%)	CPUIMP (%)	#LP-INF	#TBB-INF	#INF (over 7)
0.75	**novi**	40.1	14.3	4.1	88.7	0.26	1554.3	0.10	0.31	NA	NA	3	4	7
	vimip	40.1	14.3	4.1	88.7	0.26	1554.3	NA	0.26	NA	NA	3	4	7
	wvi	40.1	14.3	4.1	88.7	0.26	1553.3	0.10	0.31	NA	NA	3	4	7
0.80	**novi**	40.1	14.3	4.1	88.7	0.26	1521.0	0.09	0.31	NA	NA	3	4	7
	vimip	40.1	14.3	4.1	88.7	0.26	1521.0	NA	0.26	NA	NA	3	4	7
	wvi	40.1	14.3	4.1	88.7	0.26	1529.3	0.09	0.31	NA	NA	3	4	7
0.85	**novi**	40.1	10.1	4.5	72.8	0.20	800.9	0.95	1.15	14.0	98.9	0	5	5
	vimip	40.1	10.1	4.5	72.8	0.20	790.4	0.53	0.35	14.0	99.4	0	5	5
	wvi	40.1	10.1	4.5	72.8	0.19	876.4	0.96	1.15	14.0	98.9	0	5	5
0.90	**novi**	40.1	4.1	4.1	50.8	0.10	2087.9	15.71	15.82	25.9	99.3	0	1	1
	vimip	40.1	4.1	4.1	50.8	0.10	1723.0	3.03	2.70	25.9	99.7	0	1	1
	wvi	40.1	4.1	4.1	50.8	0.10	2042.4	15.92	16.01	25.9	99.3	0	1	1
0.95	**novi**	40.1	3.0	4.7	47.5	0.08	2945.3	37.48	37.56	21.0	99.1	0	1	1
	vimip	40.1	3.0	4.7	47.5	0.08	2599.9	4.61	4.03	21.0	99.6	0	1	1
	wvi	40.1	3.0	4.7	47.5	0.08	2934.0	37.93	38.01	21.0	99.1	0	1	1

Table C.10 – Fixing multiples, o10, fo10, vc10, ab20 (optimal solutions not known), ε=1%, #ABV=129, IGB=1926%

t	VI?	INT0 (%)	#ITER	FIX/ITER (%)	FIX (%)	CPU_LP	#NODE	CPU_TBB	CPU_Σ	LPGAP (%)	#LP-INF	#TBB-INF	#INF (over 4)
0.75	**novi**	22.5	41.8	2.4	92.2	5.86	4960.0	0.14	5.97	1294.4	1	2	3
	vimip	22.5	41.8	2.4	92.2	5.86	4959.7	1.78	6.31	1294.4	1	2	3
	wvi	22.5	41.8	2.4	92.2	5.88	85778.0	0.14	5.98	1294.4	1	2	3
0.80	**novi**	22.5	37.3	2.6	92.0	4.70	103611.8	0.25	4.95	1294.4	0	3	3
	vimip	22.5	37.3	2.6	92.0	4.70	103611.5	1.52	5.08	1294.4	0	3	3
	wvi	22.5	41.3	2.5	91.6	5.31	1382.7	0.19	5.45	1294.4	1	2	3
0.85	**novi**	22.5	28.8	2.9	73.6	4.05	104404.3	19.89	23.95	1018.2	0	2	2
	vimip	22.5	28.8	2.9	73.6	4.05	103322.5	6.90	7.50	1018.2	0	2	2
	wvi	22.5	31.3	2.9	73.0	4.36	24436.8	20.66	25.01	1018.2	0	2	2
0.90	**novi**	22.5	14.8	3.6	47.8	2.93	20268.3	1945.18	1948.10	928.6	0	1	1
	vimip	22.5	14.8	3.6	47.8	2.93	13355.0	36.38	30.21	928.6	0	1	1
	wvi	22.5	9.5	3.9	46.2	1.97	35193.0	1946.34	1948.30	2228.0	0	0	0
0.95	**novi**	22.5	8.3	4.5	44.4	1.73	75318.8	2205.88	2207.61	2327.9	0	0	0
	vimip	22.5	8.3	4.5	44.4	1.73	106552.5	1887.93	1889.66	2214.3	0	0	0
	wvi	22.5	8.3	4.5	44.4	1.73	75285.8	2206.02	2207.75	2327.9	0	0	0

Table C.11 – Fixing singles, fo7, fo8, fo9, fo11, o7, o8, o9 (optimal solutions known), ε=1%, #ABV=129

t	VI?	INT0 (%)	#ITER	FIX/ITER (#)	FIX (%)	CPU_LP	#NODE	CPU_TBB	CPU_Σ	IGAP (%)	CPUIMP (%)	#LP-INF	#TBB-INF	#INF (over 7)
0.75	**novi**	40.1	33.9	2.0	88.1	0.59	1540.0	0.13	0.67	NA	NA	3	4	7
	vimip	40.1	33.9	2.0	88.1	0.59	1540.0	NA	0.59	NA	NA	3	4	7
	wvi	40.1	33.9	2.0	88.1	0.59	2291.3	0.14	0.67	NA	NA	3	4	7
0.80	**novi**	40.1	32.4	2.0	86.1	0.57	1357.6	0.17	0.69	NA	NA	2	5	7
	vimip	40.1	32.4	2.0	86.1	0.57	1357.6	NA	0.57	NA	NA	2	5	7
	wvi	40.1	32.4	2.0	86.1	0.57	1984.0	0.18	0.69	NA	NA	2	5	7
0.85	**novi**	40.1	25.6	2.0	72.2	0.48	483.1	1.03	1.51	14.0	98.9	0	5	5
	vimip	40.1	25.6	2.0	72.2	0.48	458.0	0.53	0.64	14.0	99.3	0	5	5
	wvi	40.1	25.6	2.0	72.2	0.47	797.9	1.04	1.50	14.0	98.9	0	5	5
0.90	**novi**	40.1	8.7	2.0	50.8	0.18	2048.0	17.56	17.73	25.9	99.3	0	1	1
	vimip	40.1	8.7	2.0	50.8	0.18	1723.0	3.00	2.75	25.9	99.7	0	1	1
	wvi	40.1	8.7	2.0	50.8	0.17	2037.4	17.49	17.65	25.9	99.3	0	1	1
0.95	**novi**	40.1	6.3	2.0	47.5	0.13	2837.4	40.25	40.38	21.0	99.1	0	1	1
	vimip	40.1	6.3	2.0	47.5	0.13	2599.9	4.60	4.08	21.0	99.6	0	1	1
	wvi	40.1	6.3	2.0	47.5	0.13	2866.7	40.23	40.36	21.0	99.1	0	1	1

Table C.12 – Fixing singles, o10, fo10, vc10, ab20 (optimal solutions not known), ε=1%, #ABV=325, IGB=1926%

t	VI?	INT0 (%)	#ITER	FIX/ITER (#)	FIX (%)	CPU_LP	#NODE	CPU_TBB	CPU_Σ	LPGAP (%)	#LP-INF	#TBB-INF	#INF (over 4)
0.75	**novi**	22.5	129.8	2.0	94.5	14.30	27477.0	0.35	14.65	1294.4	0	3	3
	vimip	22.5	129.8	2.0	94.5	14.30	27476.5	1.77	14.75	1294.4	0	3	3
	wvi	22.5	129.8	2.0	94.5	14.11	12981.3	0.34	14.45	1294.4	0	3	3
0.80	**novi**	22.5	119.3	2.0	83.5	13.78	2865.0	5.42	19.20	1066.5	0	3	3
	vimip	22.5	119.3	2.0	83.5	13.78	3005.3	5.67	15.20	1066.5	0	3	3
	wvi	22.5	119.3	2.0	83.5	13.77	4131.5	5.42	19.19	1066.5	0	3	3
0.85	**novi**	22.5	102.5	2.0	71.9	12.39	2806.0	19.35	31.74	1082.5	0	2	2
	vimip	22.5	102.5	2.0	71.9	12.39	2601.3	9.00	16.89	1082.5	0	2	2
	wvi	22.5	102.5	2.0	71.9	12.35	3040.3	18.92	31.27	1082.5	0	2	2
0.90	**novi**	22.5	62.3	2.0	48.8	8.62	22181.5	1945.81	1954.43	928.6	0	1	1
	vimip	22.5	62.3	2.0	48.8	8.62	16009.0	37.33	36.62	928.6	0	1	1
	wvi	22.5	62.3	2.0	48.8	8.55	41814.0	1946.89	1955.43	928.6	0	1	1
0.95	**novi**	22.5	52.5	2.0	44.5	7.32	68388.8	2197.18	2204.50	2101.1	0	0	0
	vimip	22.5	52.5	2.0	44.5	7.32	90405.0	1888.67	1896.00	2112.0	0	0	0
	wvi	22.5	52.5	2.0	44.5	7.32	68397.5	2201.00	2208.32	2101.1	0	0	0

Table C.13 – Fixing multiples, o7, o8, o9, fo7, fo8, fo9, fo10 (optimal solutions known), ε=0.5%, #ABV=123

t	VI?	INT0 (%)	#ITER	FIX/ITER (%)	FIX (%)	CPU_LP	#NODE	CPU_TBB	CPU_Σ	IGAP (%)	CPUIMP (%)	#LP-INF	#TBB-INF	#INF (over 7)
0.75	**novi**	42.7	7.3	466.3	63.9	0.15	2623.6	7.46	5.48	17.1	99.4	2	1	3
	vimip	42.7	7.3	466.3	63.9	0.15	2398.6	1.47	0.99	17.1	99.7	2	1	3
	wvi	42.7	7.3	466.3	63.9	0.15	3208.6	7.53	5.54	17.1	99.4	2	1	3
0.80	**novi**	42.7	7.1	474.8	63.7	0.15	2027.0	6.08	5.36	17.1	99.4	1	2	3
	vimip	42.7	7.1	474.8	63.7	0.15	1839.5	1.46	0.99	17.1	99.7	1	2	3
	wvi	42.7	7.1	474.8	63.7	0.15	1903.8	6.11	5.38	17.1	99.4	1	2	3
0.85	**novi**	42.7	3.4	407.9	51.2	0.08	1768.1	9.21	9.29	22.0	99.5	0	1	1
	vimip	42.7	3.4	407.9	51.2	0.08	1598.3	2.34	2.08	22.0	99.7	0	1	1
	wvi	42.7	3.4	407.9	51.2	0.08	1742.4	9.23	9.31	22.0	99.5	0	1	1
0.90	**novi**	42.7	3.4	407.9	51.2	0.07	1521.1	9.59	9.67	22.0	99.5	0	1	1
	vimip	42.7	3.4	407.9	51.2	0.07	1351.3	2.39	2.12	22.0	99.7	0	1	1
	wvi	42.7	3.4	407.9	51.2	0.07	1475.7	9.68	9.75	22.0	99.5	0	1	1
0.95	**novi**	42.7	2.4	430.0	48.1	0.06	2740.6	11.85	11.92	19.8	98.7	0	0	0
	vimip	42.7	2.4	430.0	48.1	0.06	2492.4	3.69	3.75	19.8	99.5	0	0	0
	wvi	42.7	2.4	430.0	48.1	0.06	2740.6	11.90	11.96	19.8	98.7	0	0	0

Table C.14 – Fixing multiples, fo11, o10, vc10, ab20 (optimal solutions not known), ε=0.5%, #ABV=335, IGB=1721%

t	VI?	INT0 (%)	#ITER	FIX/ITER (%)	FIX (%)	CPU_LP	#NODE	CPU_TBB	CPU_Σ	LPGAP (%)	#LP-INF	#TBB-INF	#INF (over 4)
0.75	**novi**	20.7	33.5	2.3	77.1	4.64	17452.5	137.70	73.49	1489.9	2	0	2
	vimip	20.7	33.5	2.3	77.1	4.64	18203.5	38.42	23.85	1489.9	2	0	2
	wvi	20.7	33.5	2.3	77.1	4.62	17452.5	139.59	74.42	1489.9	2	0	2
0.80	**novi**	20.7	32.5	2.4	75.8	4.22	13170.3	88.93	70.92	1302.3	1	1	2
	vimip	20.7	32.5	2.4	75.8	4.22	13670.7	38.24	23.34	1302.3	1	1	2
	wvi	20.7	32.5	2.4	75.8	4.21	12984.0	88.11	70.30	1302.3	1	1	2
0.85	**novi**	20.7	28.5	2.5	59.6	4.12	15674.3	118.60	93.06	1132.0	1	0	1
	vimip	20.7	28.5	2.5	59.6	4.12	15423.3	32.79	28.71	1132.0	1	0	1
	wvi	20.7	28.5	2.5	59.6	4.10	15674.3	119.56	93.77	1132.0	1	0	1
0.90	**novi**	20.7	17.0	2.8	45.8	2.80	46636.5	2043.46	2046.26	2191.5	0	0	0
	vimip	20.7	17.0	2.8	45.8	2.80	107070.3	1857.45	1860.25	2130.0	0	0	0
	wvi	20.7	22.0	2.8	52.1	3.84	33127.5	248.91	252.75	919.2	0	1	1
0.95	**novi**	20.7	11.3	2.9	37.7	1.96	56446.0	2086.26	2088.21	1875.7	0	0	0
	vimip	20.7	11.3	2.9	37.7	1.96	81418.8	1876.67	1878.63	1744.9	0	0	0
	wvi	20.7	11.3	2.9	37.7	2.00	56376.8	2086.34	2088.34	1875.7	0	0	0

Table C.15 – Fixing singles, o7, o8, o9, fo7, fo8, fo9, fo10 (optimal solutions known), ε=0.5%, #ABV=123

t	VI?	INT0 (%)	#ITER	FIX/ITER (#)	FIX (%)	CPU_LP	#NODE	CPU_TBB	CPU_Σ	IGAP (%)	CPUIMP (%)	#LP-INF	#TBB-INF	#INF (over 7)
0.75	**novi**	42.7	16.6	2.0	64.2	0.31	2618.6	10.08	7.51	17.1	99.3	2	1	3
	vimip	42.7	16.6	2.0	64.2	0.31	2398.6	1.46	1.15	17.1	99.7	2	1	3
	wvi	42.7	16.6	2.0	64.2	0.31	3203.6	10.20	7.60	17.1	99.3	2	1	3
0.80	**novi**	42.7	16.6	2.0	64.2	0.31	1136.6	10.11	7.53	17.1	99.3	2	1	3
	vimip	42.7	16.6	2.0	64.2	0.31	916.6	1.46	1.14	17.1	99.7	2	1	3
	wvi	42.7	16.6	2.0	64.2	0.30	1141.4	10.17	7.56	17.1	99.3	2	1	3
0.85	**novi**	42.7	6.0	2.0	51.2	0.11	1432.1	11.52	11.63	22.0	99.4	0	1	1
	vimip	42.7	6.0	2.0	51.2	0.11	1265.9	2.38	2.15	22.0	99.7	0	1	1
	wvi	42.7	6.0	2.0	51.2	0.11	1407.7	11.50	11.61	22.0	99.4	0	1	1
0.90	**novi**	42.7	6.0	2.0	51.2	0.11	1517.6	11.53	11.64	22.0	99.4	0	1	1
	vimip	42.7	6.0	2.0	51.2	0.11	1351.3	2.38	2.15	22.0	99.7	0	1	1
	wvi	42.7	6.0	2.0	51.2	0.11	1507.0	11.52	11.63	22.0	99.4	0	1	1
0.95	**novi**	42.7	4.1	2.0	48.1	0.09	2988.3	13.14	13.23	19.8	98.7	0	0	0
	vimip	42.7	4.1	2.0	48.1	0.09	2492.4	3.68	3.77	19.8	99.5	0	0	0
	wvi	42.7	4.1	2.0	48.1	0.09	2988.3	13.12	13.21	19.8	98.7	0	0	0

Table C.16 – Fixing singles, fo11, o10, vc10, ab20 (optimal solutions not known), ε=0.5%, #ABV=335, IGB=1721%

t	VI?	INT0 (%)	#ITER	FIX/ITER (#)	FIX (%)	CPU_LP	#NODE	CPU_TBB	CPU_Σ	LPGAP (%)	#LP-INF	#TBB-INF	#INF (over 4)
0.75	**novi**	20.7	106.8	2.0	76.8	11.66	15366.0	117.77	70.54	1348.0	2	0	2
	vimip	20.7	106.8	2.0	76.8	11.66	18190.5	38.30	30.81	1348.0	2	0	2
	wvi	20.7	106.3	2.0	76.3	11.62	15429.0	117.17	70.21	1348.0	2	0	2
0.80	**novi**	20.7	105.3	2.0	75.2	11.73	11877.0	78.03	70.25	1191.8	1	1	2
	vimip	20.7	105.3	2.0	75.2	11.73	13741.3	38.38	30.92	1191.8	1	1	2
	wvi	20.7	105.3	2.0	75.2	11.61	12637.3	78.28	70.32	1191.8	1	1	2
0.85	**novi**	20.7	85.3	2.0	56.0	11.25	23381.3	185.52	150.39	996.1	1	0	1
	vimip	20.7	85.3	2.0	56.0	11.25	21482.0	44.93	44.95	996.1	1	0	1
	wvi	20.7	85.3	2.0	56.0	11.09	23381.3	183.47	148.69	996.1	1	0	1
0.90	**novi**	20.7	55.5	2.0	46.0	7.09	21149.5	1949.43	1956.51	919.2	0	1	1
	vimip	20.7	55.5	2.0	46.0	7.09	30187.8	76.78	64.67	919.2	0	1	1
	wvi	20.7	55.5	2.0	46.0	7.11	40782.0	1949.41	1956.52	919.2	0	1	1
0.95	**novi**	20.7	42.5	2.0	38.9	5.62	49751.5	2073.54	2079.15	1747.7	0	0	0
	vimip	20.7	42.5	2.0	38.9	5.62	92366.8	1877.62	1883.23	1722.8	0	0	0
	wvi	20.7	42.5	2.0	38.9	5.63	49717.0	2072.88	2078.51	1747.7	0	0	0

APPENDIX D

RESULTS OBTAINED FOR THE IMPROVEMENT STEP OF LP ROUNDING HEURISTIC

The following abbreviations are used in Tables D.1-D.16:

ε	Accuracy level of the area linearization
#ABV	Number of binary variables used in the model averaged for the instances in question
IGB	Averages of the gaps of the best known solutions from the LP relaxation solution value for the problems under consideration
t	Threshold value
VI?	How valid inequalities are treated
CPU	Runtime of the improvement step of LP rounding heuristic in CPU seconds
IGAP	Gap of the solution found with respect to the optimal solution value
LPGAP	Gap of the solution found with respect to the LP relaxation solution value
%IMP	Improvement obtained in the objective function with respect to the LP rounding step
#INF (over x)	Total # of infeasible instances, x being the number of instances under consideration

Table D.1 – Improvement on fixing multiples, fo7, fo8, fo9, fo10, fo11, o7, o8, o9 (optimal solutions known), ε=10%, #ABV=135

t	0.75			0.80			0.85			0.90			0.95		
VI?	**novi**	**vimip**	**wvi**	**novi**	**vimip**	**wvi**	**novi**	**vimip**	**wvi**	**novi**	**vimip**	**wvi**	**novi**	**vimip**	**wvi**
#INF (over 8)	7	7	7	7	7	7	3	3	2	0	0	0	0	0	0
IGAP	9.4%	9.5%	9.4%	9.4%	9.5%	9.4%	10.6%	10.1%	15.4%	12.2%	11.9%	12.2%	12.2%	12.3%	12.2%
CPU	4.41	1.49	4.53	4.30	1.47	4.42	1.52	1.17	5.13	1.72	0.90	1.71	1.44	0.97	1.37
%IMP	3.4%	3.3%	3.4%	3.4%	3.3%	3.4%	6.7%	7.2%	8.7%	6.2%	6.5%	6.2%	5.2%	5.2%	5.2%

Table D.2 – Improvement on fixing multiples, o10, vc10, ab20 (optimal solutions not known), ε=10%, #ABV=373, IGB=1960%

t	0.75			0.80			0.85			0.90			0.95		
VI?	**novi**	**vimip**	**wvi**	**novi**	**vimip**	**wvi**	**novi**	**vimip**	**wvi**	**novi**	**vimip**	**wvi**	**Novi**	**vimip**	**wvi**
#INF (over 3)	3	3	3	2	2	2	2	2	2	0	0	0	0	0	0
LPGAP	NA	NA	NA	703.6%	703.6%	703.6%	679.7%	679.7%	679.7%	1665.1%	1686.9%	1665.1%	1758.6%	1706.8%	1758.6%
RT	NA	NA	NA	31.47	5.58	32.23	0.25	0.08	0.25	17.41	1.51	17.43	6.17	0.96	6.16
%IMP	NA	NA	NA	27.9%	27.9%	27.9%	0.2%	0.2%	0.2%	3.4%	1.5%	3.4%	0.9%	0.8%	0.9%

Table D.3 – Improvement on fixing singles, fo7, fo8, fo9, fo10, fo11, o7, o8, o9 (optimal solutions known), ε=10%, #ABV=135

t	0.75			0.80			0.85			0.90			0.95		
Method	**novi**	**vimip**	**wvi**	**novi**	**vimip**	**wvi**	**novi**	**vimip**	**wvi**	**novi**	**vimip**	**wvi**	**novi**	**vimip**	**wvi**
#INF (over 8)	7	7	7	7	7	7	3	3	3	0	0	0	0	0	0
IMP_IP-GAP	9.4%	9.5%	9.4%	9.4%	9.5%	9.4%	11.0%	10.1%	11.0%	12.4%	11.9%	12.4%	12.5%	12.3%	12.5%
RT	4.34	1.47	4.42	4.34	1.47	4.50	1.59	1.18	1.60	1.78	0.90	1.78	1.38	0.92	1.39
%IMP_OBJ	3.4%	3.3%	3.4%	3.4%	3.3%	3.4%	6.4%	7.2%	6.4%	6.0%	6.5%	6.0%	5.0%	5.2%	5.0%

Table D.4 – Improvement on fixing singles, o10, vc10, ab20 (optimal solutions not known), ε=10%, #ABV=373, IGB=1960%

t	0.75			0.80			0.85			0.90			0.95		
Method	**novi**	**vimip**	**wvi**	**novi**	**vimip**	**wvi**	**novi**	**vimip**	**wvi**	**novi**	**vimip**	**wvi**	**novi**	**vimip**	**wvi**
#INF (over 3)	3	3	3	2	2	2	2	2	2	1	1	1	0	0	0
IMP_LP-GAP	NA	NA	NA	941.1%	941.1%	941.1%	941.1%	941.1%	941.1%	783.8%	783.8%	783.8%	1866.7%	1708.2%	1866.7%
RT	NA	NA	NA	6.53	1.94	6.64	6.66	1.97	6.64	2.65	0.70	2.63	23.91	4.85	23.97
%IMP_OBJ	NA	NA	NA	10.3%	10.3%	10.3%	10.3%	10.3%	10.3%	1.2%	1.2%	1.2%	1.3%	1.0%	1.3%

Table D.5 – Improvement on fixing multiples, o7, o8, o9, fo7, fo8, fo9 (optimal solutions known), ε=5%, #ABV=113

t	0.75			0.80			0.85			0.90			0.95		
VI?	**novi**	**vimip**	**wvi**	**novi**	**vimip**	**wvi**	**novi**	**vimip**	**wvi**	**novi**	**vimip**	**wvi**	**novi**	**vimip**	**wvi**
#INF (over 6)	2	2	2	1	1	1	0	0	0	0	0	0	0	0	0
IMP_IP-GAP	21.8%	21.8%	21.8%	17.9%	16.0%	17.9%	18.4%	16.5%	18.4%	19.2%	19.3%	19.2%	19.2%	19.3%	19.2%
RT	1.05	0.41	1.08	2.11	0.86	2.15	2.41	0.87	2.46	1.24	0.55	1.26	1.10	0.48	1.05
%IMP_OBJ	2.3%	2.3%	2.3%	4.2%	5.7%	4.2%	3.3%	4.9%	3.3%	0.9%	0.8%	0.9%	0.9%	0.8%	0.9%

Table D.6 – Improvement on fixing multiples, o10, fo10, fo11, vc10, ab20 (optimal solutions not known), ε=5%, #ABV=304, IGB=1471%

t	0.75			0.80			0.85			0.90			0.95		
VI?	**novi**	**vimip**	**wvi**	**novi**	**vimip**	**wvi**	**novi**	**vimip**	**wvi**	**novi**	**vimip**	**wvi**	**novi**	**vimip**	**wvi**
#INF (over 5)	5	5	5	2	2	2	0	0	0	1	1	1	0	0	0
IMP_LP-GAP	NA	NA	NA	1953%	1844%	1953%	1472%	1477%	1472%	921%	921%	921%	1505%	1439%	1505%
RT	NA	NA	NA	2570.21	2475.48	2624.42	1450.15	1442.56	1449.21	13.51	3.20	13.51	7.02	1.07	6.97
%IMP_OBJ	NA	NA	NA	22.0%	25.2%	22.0%	7.2%	7.1%	7.2%	4.3%	4.3%	4.3%	4.5%	4.5%	4.5%

Table D.7 – Improvement on fixing singles, o7, o8, o9, fo7, fo8, fo9 (optimal solutions known), ε=5%, #ABV=113

t	0.75			0.80			0.85			0.90			0.95		
VI?	**novi**	**vimip**	**wvi**	**novi**	**vimip**	**wvi**	**novi**	**vimip**	**Wvi**	**novi**	**vimip**	**wvi**	**novi**	**vimip**	**wvi**
#INF (over 6)	4	4	4	2	2	2	0	0	0	0	0	0	0	0	0
IMP_IP-GAP	21.8%	21.8%	21.8%	17.9%	17.7%	17.9%	18.9%	17.1%	18.9%	19.3%	19.3%	19.3%	19.2%	19.3%	19.2%
RT	1.12	0.42	1.12	1.34	0.45	1.34	2.64	0.51	2.64	1.50	0.56	1.50	1.26	0.44	1.25
%IMP_OBJ	2.3%	2.3%	2.3%	4.5%	4.6%	4.5%	2.9%	4.4%	2.9%	0.8%	0.8%	0.8%	0.9%	0.8%	0.9%

Table D.8 – Improvement on fixing singles, o10, fo10, fo11, vc10, ab20 (optimal solutions not known), ε=5%, #ABV=304, IGB=1471%

t	0.75			0.80			0.85			0.90			0.95		
VI?	**novi**	**vimip**	**wvi**	**novi**	**vimip**	**wvi**	**novi**	**vimip**	**wvi**	**novi**	**vimip**	**wvi**	**novi**	**vimip**	**wvi**
#INF (over 5)	5	5	5	3	3	3	1	1	1	0	0	0	0	0	0
IMP_LP-GAP	NA	NA	NA	750%	750%	736%	921%	921%	921%	1564%	1543%	1564%	1555%	1497%	1555%
RT	NA	NA	NA	386.53	122.79	726.01	7.64	2.18	7.29	1445.99	11.28	1445.93	7.55	1.30	7.59
%IMP_OBJ	NA	NA	NA	32.5%	32.5%	33.3%	4.3%	4.3%	4.3%	5.4%	4.5%	5.4%	4.0%	4.8%	4.0%

Table D.9 – Improvement on fixing multiples, fo7, fo8, fo9, fo11, o7, o8, o9 (optimal solutions known), ε=1%, #ABV=129

t	0.75			0.80			0.85			0.90			0.95		
VI?	**novi**	**vimip**	**wvi**	**novi**	**vimip**	**wvi**	**novi**	**vimip**	**wvi**	**novi**	**vimip**	**wvi**	**novi**	**vimip**	**wvi**
#INF (over 7)	7	7	7	7	7	7	5	5	5	1	1	1	1	1	1
IMP_IP-GAP	NA	NA	NA	NA	NA	NA	12.5%	12.7%	12.5%	19.7%	21.0%	19.7%	17.7%	17.7%	17.7%
RT	NA	NA	NA	NA	NA	NA	1.48	1.06	1.48	2.76	0.86	2.74	1.53	0.71	1.49
%IMP_OBJ	NA	NA	NA	NA	NA	NA	1.3%	1.1%	1.3%	4.7%	3.6%	4.7%	2.7%	2.7%	2.7%

Table D.10 – Improvement on fixing multiples, o10, fo10, vc10, ab20 (optimal solutions not known), ε=1%, #ABV=129, IGB=1926%

t	0.75			0.80			0.85			0.90			0.95		
VI?	**novi**	**vimip**	**wvi**	**novi**	**vimip**	**wvi**	**novi**	**vimip**	**wvi**	**novi**	**vimip**	**wvi**	**novi**	**vimip**	**wvi**
#INF (over 4)	3	3	3	3	3	3	2	2	2	1	1	0	0	0	0
IMP_LP-GAP	566%	518%	566%	564%	518%	511%	811%	810%	811%	878%	882%	2158%	2188%	2190%	2189%
RT	7200.03	7200.05	7200.05	7200.03	7200.06	7200.03	30.05	9.24	27.40	4.79	0.54	7.14	14.77	1.11	15.16
%IMP_OBJ	52.2%	55.7%	52.2%	52.4%	55.7%	56.2%	18.8%	18.9%	18.8%	4.5%	4.1%	3.9%	3.5%	2.3%	3.5%

Table D.11 – Improvement on fixing singles, fo7, fo8, fo9, fo11, o7, o8, o9 (optimal solutions known), ε=1%, #ABV=129

t	0.75			0.8			0.85			0.9			0.95		
VI?	**novi**	**vimip**	**wvi**	**novi**	**vimip**	**wvi**	**novi**	**vimip**	**wvi**	**novi**	**vimip**	**wvi**	**novi**	**vimip**	**wvi**
#INF (over 7)	7	7	7	7	7	7	5	5	5	1	1	1	1	1	1
IMP_IP-GAP	NA	NA	NA	NA	NA	NA	11.8%	12.7%	11.8%	20.6%	21.0%	20.6%	17.4%	17.7%	17.4%
RT	NA	NA	NA	NA	NA	NA	1.90	1.20	1.90	3.33	0.84	3.32	1.72	0.67	1.71
%IMP_OBJ	NA	NA	NA	NA	NA	NA	1.9%	1.1%	1.9%	4.0%	3.6%	4.0%	2.9%	2.7%	2.9%

Table D.12 – Improvement on fixing singles, o10, fo10, vc10, ab20 (optimal solutions not known), ε=1%, #ABV=325, IGB=1926%

t	0.75			0.8			0.85			0.9			0.95		
VI?	**novi**	**vimip**	**wvi**	**novi**	**vimip**	**wvi**	**novi**	**vimip**	**wvi**	**novi**	**vimip**	**wvi**	**novi**	**vimip**	**wvi**
#INF (over 4)	3	3	3	3	3	3	2	2	2	1	1	1	0	0	0
IMP_LP-GAP	510.5%	537.1%	510.5%	792.1%	706.3%	792.1%	851.0%	808.1%	851.0%	856.2%	882.0%	856.2%	1904.0%	1914.3%	1904.0%
RT	7200.02	7200.06	7200.05	6.11	4.44	6.17	7.03	3.07	6.83	2.39	0.53	2.37	611.42	30.15	613.87
%IMP_OBJ	56.2%	54.3%	56.2%	23.5%	30.9%	23.5%	19.6%	23.3%	19.6%	6.3%	4.1%	6.3%	4.5%	5.3%	4.5%

Table D.13 – Improvement on fixing multiples, o7, o8, o9, fo7, fo8, fo9, fo10 (optimal solutions known), ε=0.5%, #ABV=123

t	0.75			0.8			0.85			0.9			0.95		
VI?	**novi**	**vimip**	**wvi**	**novi**	**vimip**	**wvi**	**novi**	**vimip**	**wvi**	**novi**	**vimip**	**wvi**	**novi**	**vimip**	**wvi**
#INF (over 7)	3	3	3	3	3	3	1	1	1	1	1	1	0	0	0
IMP_IP-GAP	13.2%	12.7%	13.2%	13.2%	12.7%	13.2%	18.5%	18.2%	18.5%	18.5%	18.2%	18.5%	16.7%	16.4%	16.7%
RT	11.79	4.05	11.99	11.59	3.94	11.66	10.19	3.31	10.11	10.57	3.39	10.69	7.69	2.81	7.64
%IMP_OBJ	3.3%	3.8%	3.3%	3.3%	3.8%	3.3%	2.9%	3.2%	2.9%	2.9%	3.2%	2.9%	2.6%	2.9%	2.6%

Table D.14 – Improvement on fixing multiples, fo11, o10, vc10, ab20 (optimal solutions not known), ε=0.5%, #ABV=335, IGB=1721%

t	0.75			0.8			0.85			0.9			0.95		
VI?	**novi**	**vimip**	**wvi**	**novi**	**vimip**	**wvi**	**novi**	**vimip**	**wvi**	**novi**	**vimip**	**wvi**	**novi**	**vimip**	**wvi**
#INF (over 4)	2	2	2	2	2	2	1	1	1	0	0	1	0	0	0
IMP_LP-GAP	849%	784%	849%	1044%	1041%	1044%	923%	896%	923%	2068%	2017%	867%	1865%	1531%	1865%
RT	800.17	14.25	799.81	770.08	175.51	777.06	9.98	2.95	10.01	282.19	4.88	9.80	7.56	1.65	7.52
%IMP_OBJ	34.8%	38.2%	34.8%	17.7%	17.9%	17.7%	14.9%	16.7%	14.9%	4.7%	4.6%	4.4%	1.0%	5.6%	1.0%

Table D.15 – Improvement on fixing singles, o7, o8, o9, fo7, fo8, fo9, fo10 (optimal solutions known), ε=0.5%, #ABV=123

t	0.75			0.8			0.85			0.9			0.95		
VI?	**novi**	**vimip**	**wvi**	**novi**	**vimip**	**wvi**	**novi**	**vimip**	**wvi**	**novi**	**vimip**	**wvi**	**novi**	**vimip**	**wvi**
#INF (over 7)	3	3	3	3	3	3	1	1	1	1	1	1	0	0	0
IMP_IP-GAP	13.2%	12.7%	13.2%	13.2%	12.7%	13.2%	18.5%	18.2%	18.5%	18.5%	18.2%	18.5%	16.7%	16.4%	16.7%
RT	11.37	3.92	11.45	11.35	3.96	11.40	10.21	3.36	10.18	10.34	3.35	10.23	7.64	2.77	7.75
%IMP_OBJ	3.3%	3.8%	3.3%	3.3%	3.8%	3.3%	2.9%	3.2%	2.9%	2.9%	3.2%	2.9%	2.6%	2.9%	2.6%

Table D.16 – Improvement on fixing singles, fo11, o10, vc10, ab20 (optimal solutions not known), ε=0.5%, #ABV=335, IGB=1721%

t	0.75			0.8			0.85			0.9			0.95		
VI?	**novi**	**vimip**	**wvi**	**novi**	**vimip**	**wvi**	**novi**	**vimip**	**wvi**	**novi**	**vimip**	**wvi**	**novi**	**vimip**	**wvi**
#INF (over 4)	2	2	2	2	2	2	1	1	1	1	1	1	0	0	0
IMP_LP-GAP	768.1%	767.2%	768.1%	821.7%	820.8%	821.7%	943.1%	942.5%	943.1%	866.1%	865.5%	866.1%	1685.1%	1659.5%	1685.1%
RT	362.59	95.13	98.56	53.13	9.81	50.97	14.23	2.68	14.16	14.03	2.62	13.68	19.53	1.30	19.52
%IMP_OBJ	36.0%	36.1%	36.0%	27.5%	27.6%	27.5%	4.5%	4.5%	4.5%	4.5%	4.5%	4.5%	2.1%	2.2%	2.1%

APPENDIX E

THE PARENT REPLACEMENT SCHEME TO TEST A MUTATION OPERATOR

bestfeas : the material handling cost of the best feasible solution among the solutions generated by mutation
bestinfeas : the material handling cost of the best infeasible solution among the solutions generated by mutation
parent : the material handling cost of the parent

replace:
If the parent is feasible
    If *bestfeas* exists and *bestinfeas* does not exist
        If (*bestfeas*<*parent*)
            Replace parent with the best feasible solution
        Else
            Replace the infeasible solution with the greatest number of flags in the population with the best feasible solution (break ties randomly)
    If *bestfeas* does not exist and *bestinfeas* exists
        Call **noreplace**
    If *bestfeas* and *bestinfeas* both exist
        If (*bestfeas*<*parent*<*bestinfeas*) or (*bestfeas*<*bestinfeas*<*parent*) or (*bestinfeas*<*bestfeas*<*parent*)
            Replace parent with the best feasible solution
        Else if (*bestinfeas*<*parent*<*bestfeas*)
            Replace the infeasible solution with the greatest number of flags in the population with the best feasible solution (break ties randomly)
        Else if (*parent*<*bestfeas*<*bestinfeas*) or (*parent*<*bestinfeas*<*bestfeas*)
            Call **noreplace**
Else if the parent is infeasible
    If *bestfeas* exists and *bestinfeas* does not exist
        Replace parent with the best feasible solution
    If *bestfeas* does not exist and *bestinfeas* exists
        If (*bestinfeas*<*parent*)
            Replace parent with the best infeasible solution
        Else
            Call **noreplace**
    If *bestfeas* and *bestinfeas* both exist
        If (*bestfeas*<*parent*<*bestinfeas*) or (*bestfeas*<*bestinfeas*<*parent*) or (*bestinfeas*<*bestfeas*<*parent*)
            Replace parent with the best feasible solution
        Else if (*bestinfeas*<*parent*<*bestfeas*)
            Replace parent with the best feasible solution

Else if (*parent*<*bestfeas*<*bestinfeas*) or (*parent*<*bestinfeas*<*bestfeas*)
Call **noreplace**

noreplace:
If (# of flagged departments of the parent < # of departments – 1)
Flag the related department in the parent and discard the offspring
Else
Retire the parent (save if better than the current best retired solution) and discard the offspring