GENETIC ALGORITHM BASED RE-CONFIGURATION OF PLANT LAYOUTS

A CASE STUDY ON AKAKI SPARE PARTS AND HAND TOOLS FACTORY

By

Tesfaye Tadesse

A thesis Submitted to the School of Graduate Studies of Addis Ababa University in Partial Fulfillment of the Requirement of the Degree of Master's of Science in Mechanical Engineering

(Industrial Engineering Stream)

Advisor: Dr. Subhash Chandra

ADDIS ABABA UNIVERSITY, SCHOOL OF GRADUATE STUDIES,
FACULTY OF TECHNOLOGY, MECHANICAL ENGINEERING
DEPARTMENET

July, 2007

ADDIS ABABA UNIVERSITY
SCHOOL OF GRADUATE STUDIES
DEPARTMENT OF MECHANICAL ENGINEERING

Genetic Algorithm Based Re- configuration of Plant Layouts

A case study on Akaki Spare Parts and Hand Tools Factory

By Tesfaye Tadesse

Approved by Board of Examiners

Ato Hamssasew Moges _____
Chairman, Department

Dr. S. Chandra _____
Advisor

Ato Lalisa Daba _____
External Examiner

Dr.Ing. Daniel Kitaw _____
Internal Examiner

Candidate's Declaration

I hereby declare that the work which is being presented in this thesis entitled "Genetic Algorithm Based Re-configuration of Plant Layouts, A case study on Akaki Spare Parts and Hand Tools Factory" is original work of my own, hasn't been presented for a degree in any other university and that all sources of materials used for the thesis have been duly acknowledged.

Tesfaye Tadesse — Date

(Candidate)

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

Dr. S. Chandra — Date

(Thesis Advisor)

Acknowledgments

My first and heartily acknowledgment goes to my professor, Dr. Subhash Chandra, for his valuable help and contribution in and out of office time during the preparation of this research paper. In my Industrial Engineering educational carrier, I strongly want to appreciate our great model Dr.Ing. Daniel Kitaw, a man of values and beliefs who is teaching his students not only in a class session but also in the way of life he is living practically.

I also want to acknowledge the engineers and maintenance crew at Akaki Spare Parts and Hand tools Share Company. My Acknowledgment also goes to all my friends who helped me in the coding of VB program, in reading and commenting on my paper.

Last but not least I would like to thank my beloved families who are always behind me with full support in my entire educational carrier.

These all are nothing without my Lord, so my greatest and heartfelt thanks go to the almighty GOD who gave me not only what I asked him but also what I should have.

Table of Contents

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

This paper describes a novel method based on genetic algorithms (GA) to solve the facilities re–configuration problem. Developing a proper floor layout is an important step in designing manufacturing facilities due to the impact of the layout to material handling cost and time, and its direct consequence on the overall productivity of the shop floor.

Poor layout would result in having more parts spending longer time in moving from one facility to the other, and results in increased material handling costs. In contrast to the block layout the objective of facilities re-configuration is to find the appropriate placement of facilities with in the existing boundary of the factory floor for the new product coming in to the manufacturing system.

The genetic algorithm based method developed to solve this uses the objective of minimizing the movement of materials being processed in the factory.

This thesis investigates the potential of genetic algorithms in re-configuring factory floor facilities and presents a computer automated system for performing facilities re-configuration task of an existing factory floor. The system integrates the powerful graphical capabilities of AutoCAD and the intricate search and optimization abilities of genetic algorithms for the purpose of solving facilities re-configuration problem.

The computer automated system is implemented via Visual Basic 6.0. The interface features of Visual Basic and AutoCAD are utilized to capture the geometrical details of the existing factory floor layout and to represent the final solution graphically.

Akaki Spare Parts and Hand Tools factory is selected as one of the factory which is suffering from re-configuration of its factory floor facilities. The final part of this thesis shall validate the systems performance taking ASPSC's hand tools and cutlery division facilities as a case which is mostly dedicated to produce one type of product at a time.

CHAPTER-1

INTRODUCTION

1.1 Background

There is an emerging consensus that existing layout configurations do not meet the needs of the multi-product enterprises [9, 16] and that there is a need for a new generation of factory layouts that are more flexible, modular, and more easily reconfigurable. Flexibility, modularity, and re-configurability could save factories the need to redesign their layouts each time their production requirements change. Re-layout can be highly expensive and disruptive, especially when the entire factory has to be shut down and production stopped. For factories that operate in volatile environments or produce a high variety of products, shutting down each time when demand changes or a new product is introduced, is simply not an option. In fact, plant managers may prefer to live with the inefficiencies of an existing layout rather than suffer through a costly re-layout, which in turn could become quickly obsolete. Most companies, ranging from big to small, encountered mounting frustration with the existing layout choices. This is particularly acute in companies that continuously introduce and offer a wide range of products whose demands are variable and lifecycles are short. For these companies, being able to design a layout that can either retain its usefulness over a wide range of product mixes and volumes or be easily reconfigured is extremely valuable. Equally important is designing layouts that can support the need for increased customer responsiveness in the form of shorter lead times, lower inventories, and higher product customization.

The current choices of layouts, such as *product*, *process*, and *cellular* layouts do not adequately address the above needs because they tend to be designed for a specific product mix and production volume, both assumed to last for a sufficiently long period (e.g. 3-5 years). [9, 16]

The design criterion that routinely used in most layout design procedures is a measure of long-term material handling efficiency which fails to capture the priorities of the flexible factory (e.g. responsiveness is more important than cost, and re-configurability is more important than efficiency).

As a result, layout performance tends to deteriorate significantly with fluctuation in product volumes, mix, or routings [9, 16]. Using a static measure of material handling efficiency also fails to capture the impact of layout configuration on operational performance, such as work-in-process, accumulation, queue times at processing departments, and throughput rates. Consequently, layouts that improve material handling often result in inefficiencies elsewhere in the form of long lead times or large in-process inventories [9].

Some of the limitations underlying many of the tools and methods used to design and evaluate factory layouts, making them less effective in factories with high product variety or short lifecycles are:

- ***Use of the travel chart as input data***: The traditional input data for layout design has been the *Travel Chart* [9]. However, this chart aggregates the routings and production quantities of all the products produced in a facility. Being a simple graph, it prevents machine duplication analysis. Thereby, it limits the facilities planner to the design of mostly a single type of layout – the functional layout. An alternative could be a Multi-Product Process Chart that captures the unique routing of each product. Such a chart would be essentially a hyper-graph representation of the facility that treats each routing as a hyper-edge connecting a sequence of departments in the layout. With routing information embedded in the layout, the design of layout configurations, other than the functional layout, becomes possible because partitioning the edge list allows duplication of machines in several locations in the facility [9, 16].
- ***Number of part samples and sampling criteria used to design a layout***: A common practice in industry is to use the 80-20 rule (or ABC Analysis) to select one sample of products in designing the entire layout [9]. However, a single sample is rarely an accurate representation for a facility with high product variety or a changing product mix. This problem is compounded by the use of "production volume" as the criterion in selecting a sample. Although a volume-based criterion tends to minimize material handling costs by minimizing material travel distances,

it ignores important factors such as revenue generated by each product, frequency of product ordering, and variability in order sizes.

- ***The phase in the life cycle of a facility when most models and methods are used***: In industry, the dominant use of facility layout design methods tends to be in the midlife or later life of a facility. In other words, facility planners are often engaged in evaluating an existing layout and proposing improvements to it. Clearly, there is considerable opportunity for application of layout algorithms at the conceptual design phase of planning the layout of a facility. Since production data for the entire life of a layout is not known at the initial design stage, there is a need for layout design methods that can work with fuzzy or incomplete data on product mix, routings and production quantities.

Hence, there is a need for a new class of layouts that are more flexible and responsive. There is also a need for alternative evaluation criteria for layout design that explicitly account for flexibility and responsiveness.

1.2 Problem Statement

Problems of layout develop when needed to start a new product, to change the product design or to reduce the cost [3]. The resulting problems involve planning a completely new plant, re-arranging a presently installed layout or making adjustments to the existing layout. The relation ship between the causes and classes of layout problems is shown in the figure1.1 [3].

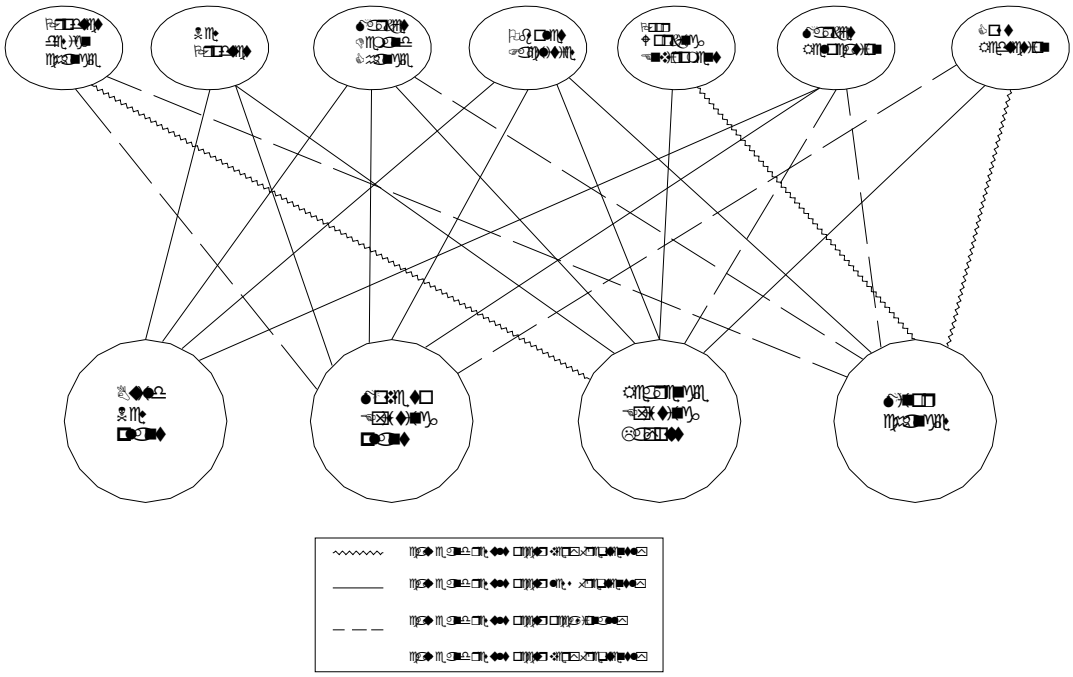


Figure 1.1 Relations between the causes and classes of layout problems

Generally facility layout problem can be broadly classified in to static facility layout models and dynamic layout models. Static facility layout models deal with the initial design of a layout for a single period, whereas dynamic facility layout models are used for designing variable layouts depending on the product mix variation provided that the product mixes and volumes are known prior to the commencement of the design activity as project phases [2].

As we see from the two models of floor layout design mentioned above, it is clearly seen that both models fail to meet the needs of variable floor layout design in the absence of prior information regarding the product mix and volume. Manufacturing factories like the Akaki Spare Parts and hand tools factory don't have enough information about the product- mix as well as volume they are going to produce for the coming year.

They mostly are going to produce a certain product when an order comes from the customer. This would have complicated the layout designing process using the dynamic layout model at the initial stage.

Due to fluctuations in the product demand and changes in the product mix material handling cost fluctuates and often increases. In addition, introduction of a new product or

discontinuing the existing products can lead to changes in material handling requirements. These changes render the existing layout inefficient and often come up with an increase in material handling costs which necessitates a re-configuration of facilities in the existing layout [8].

To make the existing layout compatible with the new product mix and volume, a partial re-configuration of facilities (which can be dismantled and relocated with a lower cost) can solve factory floor layout problems initially designed with a static layout model [8].

For example, the existing layout may be designed with a static layout model in the manner that P.D-1 (process department-1) and P.D-2 to be close to each other. But at some time in the future the factory may have a product mix which don't need the closeness of P.D-1 and P.D-2 or the closeness of those departments may come up with an increase in material handling cost. In such cases a certain rearrangements of facilities (which can be easily dismantled and relocated) from one position to other can address the problem on the manufacturing process due to facilities layout, and this is what re-configuration of facilities in the existing factory mean.

So to fill the gap between the two models mentioned above a partial reconfiguration of facilities in the existing factory floor is suggested [8].

This approach will make the existing factory floor layout to meet the demands of the new product mix and volume requirements.

1.3 Research Objectives

The efforts put into this research aim to develop an automated system for generating a new optimum layout of facilities by re-locating them with in the existing boundary of the factory.

To achieve this, the following sub-objectives are considered:

- Investigate the potential for using genetic algorithms in solving the re-configurable floor layout planning problem. The use of genetic algorithms as complex function optimizers has been for long acknowledged. Recently some research in the plant layout planning domain has been focused on using these evolutionary techniques to solve the layout planning problem.

- Formulate an automated system for reconfiguring facilities with in the existing layout boundary using Genetic Algorithm. Due to the geometric nature of the problem at hand, the researcher incorporates a CAD input/output media to facilitate the use of the system. The approach used aims to integrate the powerful graphical capabilities of CAD systems with the intricate search and optimization abilities of genetic algorithms for the purpose of solving the factory layout problem.
- Formulate a decision model for reconfiguring the existing layout of facilities. Before going to decide on reconfiguring the existing floor layout of the factory, the managers should have a decision model based on material handling costs of the existing layout for the given product-mix and re-layout costs.
- Test and validate the system. A carefully selected case study is chosen to test and validate the systems' performance.

1.4 Methodology

The proposed system primarily consists of an optimization engine and a geometric input / output interface. The link between the optimization engine and the geometrical data contained in AutoCAD drawings has been made possible through automation of AutoCAD-Visual Basic Applications. The programmable features of AutoCAD enable all the existing factory related geometrical data to be detected as an orthogonal 2-D grid. The optimization engine utilizes this grid in its execution.

To perform the optimization task, the proposed system utilizes genetic algorithms as the main optimization engine. Due to the linearity of the objective function to be optimized, the various constraints to be considered in traditional operations research techniques were found to be suitable.

1.5 Thesis Organization

This thesis is organized into 7 chapters. Chapter 2 provides a thorough review of previous studies related to plant layout planning and the applications of genetic algorithms in factory layout problems. The research approach adopted and theoretical background for this study is presented in chapter 3.

Chapter 4 presents functional details of the various components of the reconfigurable plant layout planning system as well as details concerning their integration to perform the required task. Details of the physical structure of the program are provided in chapter 5 using an illustrated example. In chapter 6, validation of the Automated Layout Re-configuration Program is performed using Akaki spare parts and hand tools manufacturing factory as a case study.

Finally in chapter 7, a strong conclusion will be forwarded to materialize this research paper and also recommendations for future enhancements are presented.

CHAPTER-2
LITERATURE REVIEW

2.1 Definition

Many researchers have attempted to define the factory layout planning process. One of the more crisp and generic definitions was given by [3].

"The task of plant layout consists of identifying the facilities needed to support manufacturing process, determining their size and shape and positioning them within the boundaries of the available area."

Examples of these facilities include process departments, cells, lay down areas, maintenance areas, offices, parking lots, warehouses, buildings etc.

Continuing from the above definition, we can say the floor facility layout reconfiguration is concerned with the location and arrangements of process departments, cells or machines in the existing factory floor so that the existing floor layout can be optimum for the new product mix or change in the product demands.

2.2 Applications in Other Domains

The layout re-configuration problem may be considered as a sub-domain of the greater "Facility Layout Problem". This problem is common in other specialization areas besides its use in industrial engineering. Facility layout has been utilized extensively in the domains of construction and electrical engineering.

2.2.1 Electrical Engineering

In electrical Engineering the facility layout problem is utilized in the physical design of VLSI (very large scale integrated) microchips in a task named "macro-cell layout generation" [2]. In this task the circuit is partitioned and the components are grouped into functional units or micro cells. These cells can be described as rectangular blocks with terminals (pins) along their borders. These terminals have to be connected by signal nets, along which power or signals (e.g. clock ticks) are transmitted between the various units of the chips. A net can connect two or more terminals and some nets must be routed to pads at the outer border of the layout, since they are involved in the I/O of the chip. The goal to attain in the layout process is the minimization of the total chip area which is greatly influenced by the area between the cells occupied by the signals net wiring. [2]

2.2.2 Construction Engineering

In Construction Engineering the facility layout problem is utilized in the physical design of positioning temporary facilities in a multi phase construction projects [2]. Here the temporary facilities of construction projects like trailers, lifting equipments, garages, parking lots, storage areas etc has to be reconfigured from phase to phase to minimize the movement of materials between facilities. So facilities reconfiguration is necessary in the different phases of the construction project.

2.3 Problem Approaches

In this section, the paper tries to describe the alternative and novel layout configuration methods for factories that must deal with high product variety or high volatility in their production requirements and go for the better option for factories in our country.

The paper focuses on three approaches to layout design that address three distinct needs of the flexible factory, namely *distributed*, *modular* and *reconfigurable* layouts.

2.3.1 Distributed layouts

The distributed layout concept is based on the notion that *disaggregating large functional departments into smaller sub-departments and distributing them throughout the plant floor can be a useful strategy in highly volatile environments.* [9,16] Having duplicates of the same departments, which can be strategically located in different areas of the factory floor is desirable in a variable environment since it allows a facility to hedge against future fluctuations in job flow patterns and volumes. The distribution of similar departments throughout the factory floor increases the accessibility to these departments from different regions of the layout. In turn, this improves the material travel distances of a larger number of product sequences. As a result, efficient flows can be more easily found for a larger set of product volumes and mixes. Examples of departments with varying degrees of department desegregation and distribution are shown in Figure 2.1. Such a procedure is especially appealing in environments where the frequency with which product demand fluctuation occurs is too high for a re-layout of the plant to be feasible after each change. Thus, a fixed layout that can perform well over the entire set of possible demand scenarios is desirable.

Disaggregating functional departments and placing the resulting smaller sub-departments in non adjoining areas of the layout poses several important design challenges. For example, how should the sub-departments is created? How many should be created? How much capacity should be assigned to each sub-department? Where each sub-department should should be placed? How should workload be allocated among similar sub-departments? There are also questions regarding the impact of department disaggregating and distribution on operational performance. For example, how would material handling times, work-in-process, and queuing times be affected? How should material flow be managed, how does a greater routing flexibility be achieved? How should the competing needs for material handling of similar sub-departments be coordinated? There are also important questions regarding what performance measure is appropriate when designing distributed layouts. Should we use a measure of expected material handling cost over the set of possible demand scenarios, or should we use a measure of robustness that guarantees a minimum level of performance under each scenario. More importantly, how sensitive are the final layouts to the adopted performance measure?

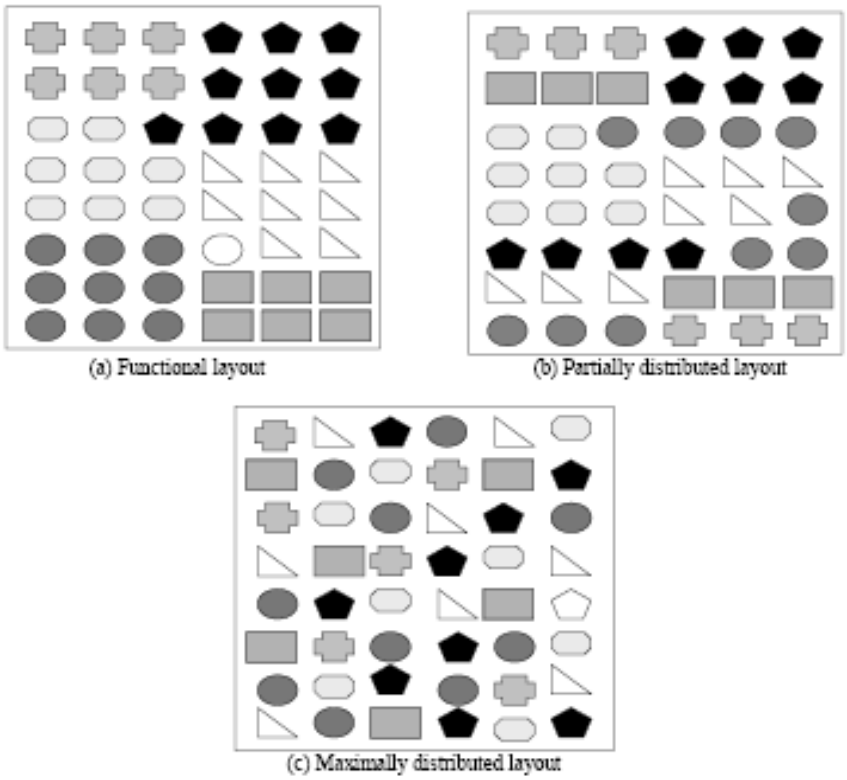


Figure 2.1**:** Layouts with varying degrees of distribution

The need for disaggregating and distributing the large functional departments throughout the factory was initially adopted to reduce the large distances that must be traveled by in-process material.

However, it was soon discovered that when coupled with effective workload allocation, this distribution resulted in significantly lower material handling costs and shorter material handling times even when demand variability was high.

Some of the above questions are explored in [9, 16].

2.3.2 Modular Layouts

The focus of this approach is on design of customized layouts for facilities with multiple products.

It considers a novel approach based on the idea that *layouts can be constructed as a network of basic modules* [16]. Here, it assumes that at least in the short term, the product mix is known and demand is relatively stable. As the product mix evolves and demand changes, certain layout modules will be eliminated and others added. The use of modules is motivated by the fact that none of the prevailing layout configurations (functional, flow line, and cellular) can individually describe the complex material flow network in a multi-product manufacturing facility. Preliminary research on this topic was undertaken and has recently been reported in [9, 16]. The research sought to answer the following fundamentally new questions: Could an alternative layout other than the three traditional layouts be a better fit for the material flow network in a multi-product manufacturing facility? And, could this alternative layout be a combination of the three traditional layouts? The proposed concept of designing any facility layout as a network of layout modules provides a meta-structure for the design of multi-product manufacturing facilities. The proposed concept uses the idea of grouping and arranging the machines required for subsets of operations in different routings into a specific (traditional) layout configuration that minimizes total flow distances or costs.

Figure 2.2 shows an example of a new layout configuration that is being proposed for multi-product manufacturing facilities. [16]

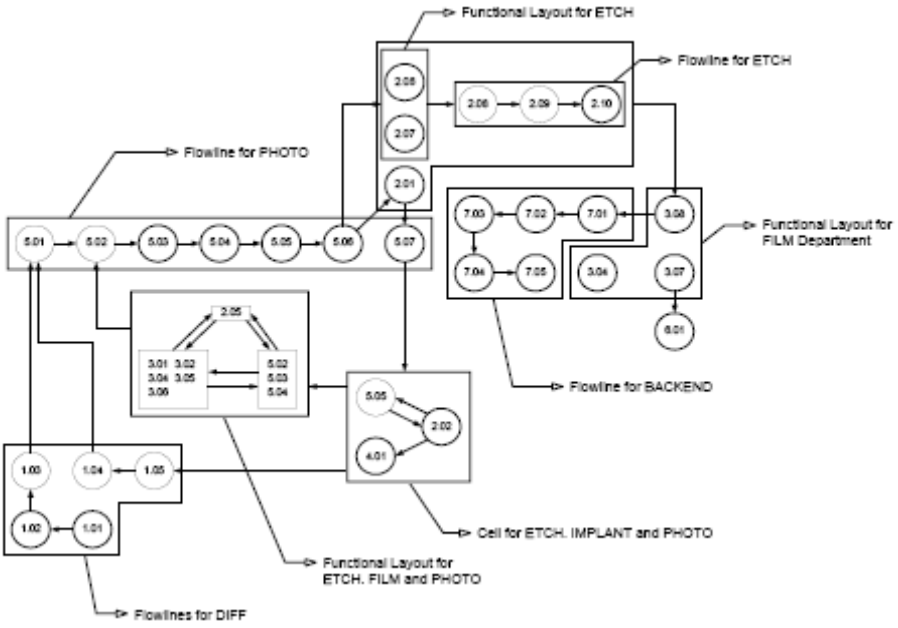


Figure 2.2: Example of facility layout designed using layout modules

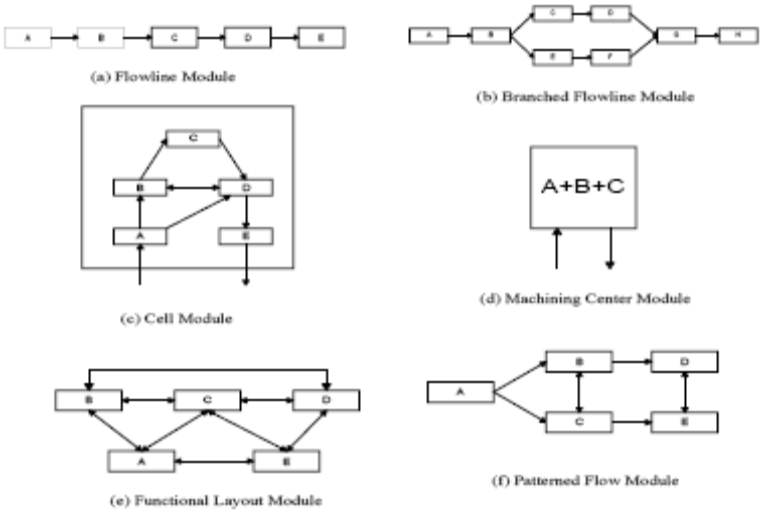


Figure 2.3: Example of layout modules

2.3.3 Reconfigurable Layouts

The third focus is on design of reconfigurable layouts. Here, it considers the case where *resources can be easily moved around so that frequent relocation of departments is feasible.* [16] This is motivated by the fact that in many industries (e.g. consumer electronics, home

appliances, garment manufacturing, etc) fabrication and assembly workstations are light and can be easily relocated. [16] In fact, even in the metal cutting industry recent advances in materials and processing technology are making it easier for manufacturing facilities to be configured and reconfigured on a more frequent basis. For example, many discrete manufactured components are made of composite materials that are light in weight and have much better mechanical properties (e.g. vibration absorption properties). Aluminum composites, for instance, can now replace cast iron parts [9, 16] and phenolics are replacing aluminum parts among the others. Newer processing technologies such as electron beam hardenings, molecular nano-technology and laser cutting is resulting in lighter weight machining equipment. Permanent magnetic chucks that carry their own energy source, that do not obstruct machining and do not magnetize the cutting tool are also being developed. With these developments in materials and processing technology, we are moving towards processing technologies which employ light weight machine tools and can process light weight parts. It is possible to envision facilities where these light weight equipment are mounted on wheels and are easily moved along suitably designed tracks embedded in the shop-floor.[16] As a result, it may not be too far fetched to say that the layout will be changed several times a year. In fact, through a workshop survey, the committee on Visionary Manufacturing Challenges for 2020 has identified *adaptable processes and equipment* and *reconfiguration of manufacturing operations* as two key enabling technologies that will help companies overcome two of the six grand challenges or fundamental goals to remain productive and profitable in the year 2020.[16] These grand challenges are to "achieve concurrency in all operations" and to "reconfigure manufacturing enterprises rapidly in response to changing needs and opportunities".[16]

When frequent re-layout is feasible, the layout design problem can be significantly simplified even when product demand and product mix are highly variable. It becomes possible to focus only on the immediate product mix and the immediate production volumes. However, since we would typically incur:

(1) Some loss in production capacity during the relocation process, and

(2) A relocation cost associated with the physical movement of resources (e.g., labor cost, dismantling and reconstruction costs, rewiring costs, and startup/setup costs), we must

account for these costs when deciding whether it is beneficial to remove a resource or leave in its current location [2, 16].

The objective function of the model consists of two terms: a material handling cost term and a relocation cost term. The magnitude of the relocation costs determine whether a re-layout is carried out or not [2, 16]. In the extreme, where re-layout costs are insignificant, an entirely new layout can be generated during each period. On the other hand, if re-layout costs are prohibitive, the existing layout would be retained. In practice, the two extreme scenarios would be unlikely to occur. Instead it would be desirable to relocate some of the resources during each period. The layout would then evolve gradually over time as flow patterns evolve. The cost of re-layout could be reduced if investments in infrastructure that facilitates re-layout are made during the initial design of the factory. For example, the facility may be designed so that it has embedded tracks that help decrease the cost of moving equipment. It may also be possible to design all interface devices for control systems so that they are interchangeable and open. In such a case, "plug and play" features may be implemented at the workstation level. Support services such as compressed gas, water or coolant lines, and waste disposal may have to be suitably designed for the concept.

A primary advantage of reconfiguring a layout when warranted by changes in product mix and volume is that material handling cost can be minimized because equipment can be reconfigured to suit the new production mix and volume. Of course, this cost must more than the cost of moving equipment from its current location to a new one. In addition, due to the short term life of a given layout and production data availability for this time period, it is possible to consider optimizing operational performance measures such as minimizing part cycle times, work in process inventory, or throughput.

The potential to frequently alter layouts, therefore, transforms the modern layout problem from a strategic problem in which only long term material handling costs are considered to a tactical problem in which operational performance measures such as reduction of product flow times, work in process inventories, and maximizing throughput rate are considered in addition to material handling and machine relocation costs when changing from one layout configuration to the next.

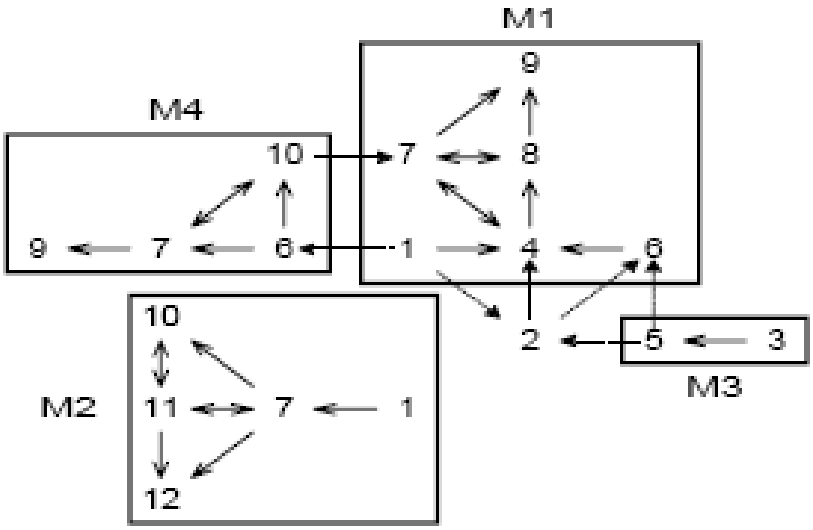


Figure 2.4: Reconfigurable Facility

Reconfiguration of floor facilities in a factory of dynamic environment also works for existing factory layouts designed with a static layout model [2; 16], so this method can best handle the existing factory floor layout problems that we have in our country.

2.4 Optimization Method Based On Genetic Algorithm

For three decades, many mathematical programming methods have been developed to solve optimization problems. However, until now there has not been a single totally efficient and robust method to cover all optimization problems that arise in the different engineering fields.

Most engineering application design problems involve the choice of design variable values that better describe the behavior of a system [1].

In design problems, the variables are discreet from the mathematical point of view. However, most mathematical optimization applications are focused and developed for continuous variables [1].

Given the shortcomings of the calculus based techniques and the numerical ones the random methods have increased their popularity.

The methods of random search are known as evolutionary algorithms. The evolutionary techniques are parallel and globally robust optimization methods. They are based on the principles of natural selection of Darwin [11]. The application of evolutionary techniques as abstractions of the natural evolution has been broadly proven [11].

In general, all recursive approaches based on population, which use selection and random variation to generate new solutions, can be seen as evolutionary techniques. The genetic algorithm is an example of a search procedure that uses random selection for optimization of a function by means of the parameters space coding. The genetic algorithms were developed by Holland [11].

In the a very broad sense genetic algorithms or GA's, are search algorithms based on the mechanics of natural selection and natural genetics. They combine survival of the fittest among string structures with seemingly randomized information exchange to form a search algorithm with some of the innovative flair of human search. In every generation, a new set of artificial creatures (strings) is created using bits and pieces of the fittest of the old; an occasional new part is tried for good measure. While randomized, genetic algorithms are no simple random walk. They efficiently exploit historical information to speculate on new search points with expected improved performance [11].

The genetic algorithms have been proven successful for robust searches in complex spaces. For these reasons Genetic Algorithms are broadly used in daily activities, as much in scientific applications as in business and engineering circles.

The genetic algorithms (G.A.) are typically characterized by the following aspects:

- The G.A. work with the base in the code of the variables group (artificial genetic strings) and not with the variables in themselves.
- The G.A. work with a set of potential solutions (population) instead of trying to improve a single solution.
- The G.A. do not use information obtained directly from the object function, of its derivatives, or of any other auxiliary knowledge of the same one.
- The G.A. applies probabilistic transition rules, not deterministic rules.

2.4.1 GA terminologies

The parallelism between GA's for biological evolution and natural selection is evident. Goldberg (1989) summarized this parallel nature as follows [11].

Natural Terminology	GA Terminology
Chromosome	String
Gene	Feature, character
Allele	Feature value
Locus	String position
Genotype	Structure
Phenotype	Parameter list

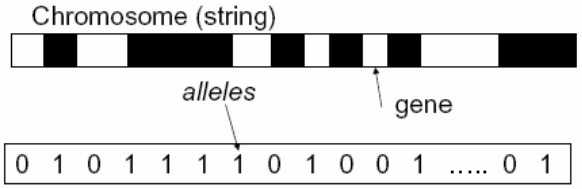


Figure 2.5: GA terminologies

Each chromosome represents a certain solution often using strings of 0's and 1's. Each bit typically corresponds to a gene and the values for the given gene are alleles. A chromosome in isolation is meaningless; we need to decode it to phenotypic value.

2.4.2 The coding and decoding schemes in Genetic Algorithms

As indicated before, the essential characteristic of genetic algorithms is the coding of the variables that describe the problem. The common coding method is to transform the variables to specific length binary strings [11]. For a problem depending on more than one variable the coding involves linking with each variable code. The following figures show the coding and decoding schemes in GA.

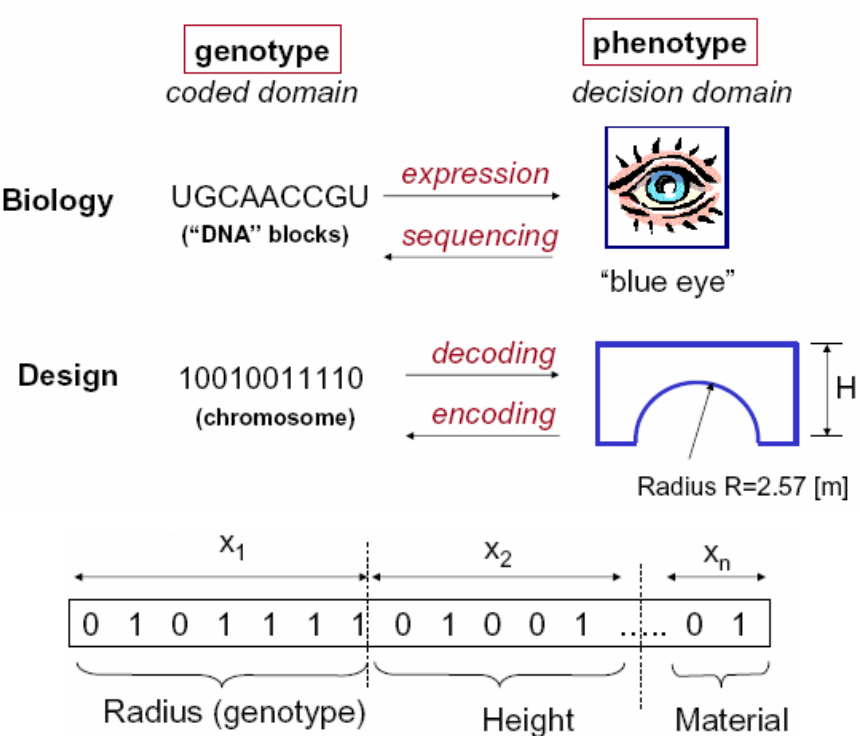


Figure 2.6: The coding and decoding schemes in GA

On [11], Goldberg presented two main principles for choosing GA encoding:

1) Principal of meaningful building blocks: "the user should select a coding so that short, low order schemata are relevant to the underlying problem and relatively unrelated to schemata over other fixed positions"
2) Principal of meaningful alphabets: "The user should select the smallest alphabet that permits a natural expression of the problem"

Some GA encodings, that have been successfully used in practice are the binary, permutation and value encoding [2, 11]. Each type is discussed in detail below.

1. Binary Encoding: The first works of GA's used this type of encoding. Each gene in a chromosome assumes either the value of 1 or 0. Binary encoding gives many possible chromosomes even with a small number of alleles. On the other hand, this encoding is often not natural for many problems and sometimes corrections must be made after crossover and/or mutation.

Chromosome A	1	1	0	1	0	0	0	1	0	0
Chromosome B	0	1	1	1	1	0	1	0	1	1

Figure 2.7: Example of a chromosome with a 10-bit binary encoding

2. Permutation Encoding: Every chromosome is a string of numbers, which represents numbers in a sequence. This type of encoding is only useful for ordering problems.

Chromosome A	1	4	2	9	0	3	5	8	7	6
Chromosome B	8	7	9	2	6	4	5	0	1	3

Figure 2.8: Example of a chromosome with a 10-bit permutation encoding

3. Value Encoding: Direct value encoding can be used in problems where complicated values such as real numbers are used. Use of binary encoding for this type of problems would be very difficult. Values of the alleles can be anything related to the problem, whole numbers, real number, characters, or even objects.

Chromosome A	1.23	6.75	9.31	0.73	5.52	7.11
Chromosome B	A	C	B	C	B	D
Chromosome C	(up)	(down)	(left)	(up)	(right)	(down)

Figure 2.9: Example of a chromosome with a 6-bit value encoding

2.4.3 Generation of population using GA

According to Goldberg [12], when GA is allowed to generate several generations, there are certain steps or procedures that should be followed as shown in figure 2.6. Three of the above procedures make actions on the existing chromosomes (individuals) to generate a new off springs (children) and these procedures are called GA operators.

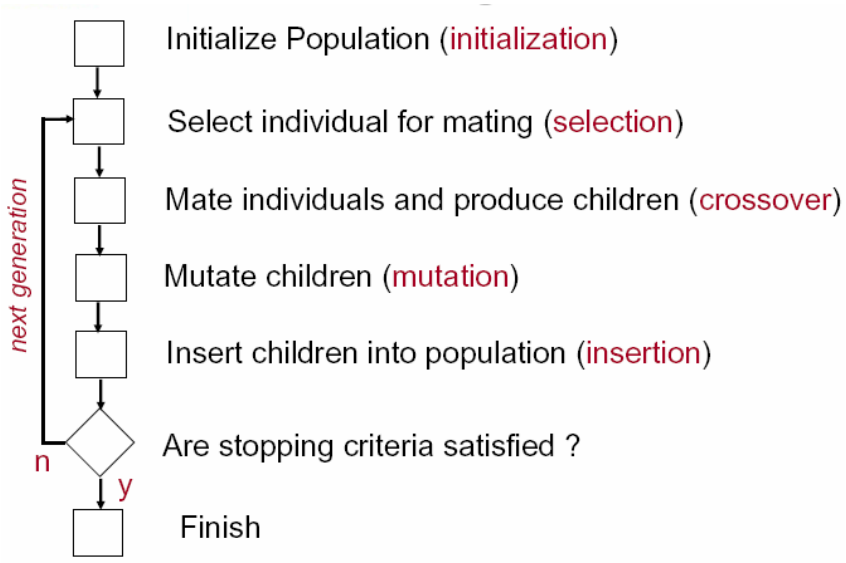


Figure 2.10: GA actions over several generations

As shown above, a basic genetic algorithm that can produce acceptable results in many practical problems is composed of three operators:

- Selection
- Crossover
- Mutation

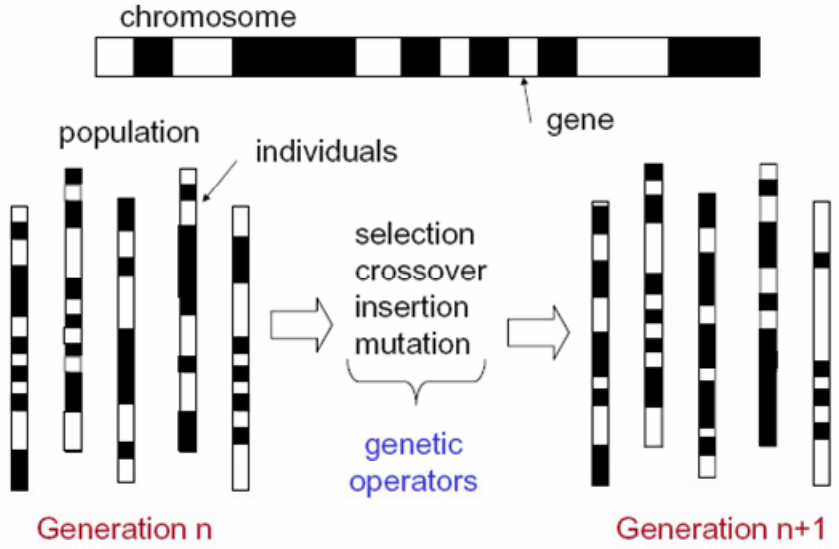


Figure 2.11: GA operators

A- Selection of the fittest

Selection of the fittest is the cornerstone of the operation of the GA. From a broader point of view it can be considered as the process of creating an intermediate population. It is this intermediate population that undergoes the genetic operations of crossover and mutation. There are different methods of performing the selection process, among these:

(1) Selection according to RANKING

Example: Let $D = \sum_{j \in P} (1/j)$
select the kth most fit member of a population
to be a parent with probability $P_k = \left(\frac{1}{k}\right) D^{-1}$

Better ranking has a higher probability of being chosen

(2) Proportional to FITNESS Value Scheme

Example: Let $\bar{F} = \sum_{j \in P} Fitness(j)$
select the kth most fit member of a population
to be a parent with probability $P_k = Fitness(k) \cdot \bar{F}^{-1}$

Chromosomes are gives a ranked fitness. The worst will have a fitness of 1, second worst 2 and so on. The best chromosome will have a fitness of N (number of chromosomes in the population).

(3) Roulette Wheel Selection

The population is mapped onto a roulette wheel, where each individual is represented by a space that proportionally corresponds to its fitness.

By repeatedly spinning the roulette wheel, individuals are chosen using "stochastic sampling with replacement".

A selection process that will more closely match the expected fitness value is "*remainder stochastic sampling*". For each string *i* where f_i / f_{avg} is greater than 1.0, the integer portion of this number indicates how many copies of that string will be copied to the intermediate population. All strings (including those with $f_i / f_{avg} < 1.0$) are then chosen with a probability corresponding to the fractional probability of f_i / f_{avg}. For example, a string with $f_i / f_{avg} = 2.3$ is chosen twice and then receives a 0.3 chance of placing a third copy.

Remainder stochastic sampling is most efficiently implemented using a method known as "*stochastic universal sampling*". Assume that the population is laid out in random order as in a pie graph, where each individual is assigned space on the pie graph in proportion to fitness. Next an outer roulette wheel is placed around the pie with N equally spaced pointers. A single spin of the roulette wheel will now simultaneously pick all X members of the intermediate population. The resulting selection is also unbiased.

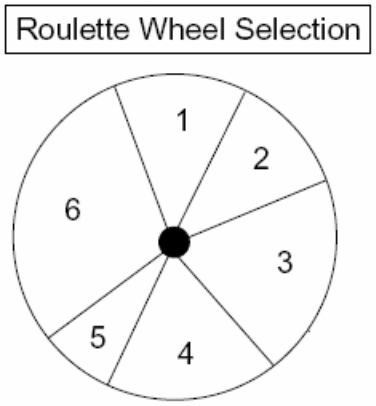


Figure 2.12: The roulette wheel

Probabilistically select individuals based on some measure of their performance.

B- Crossover

The crossover is a process by which a string is divided into segments, which are exchanged with the segments corresponding to another string. With these process two new strings different to those that produced are generated. It is necessary to clarify that the choice of strings crossed inside those that were chosen previously in the reproduction process is random. The following outline shows the different types of crossover process:

(1). Single point cross over: One crossover point is randomly selected. String form the beginning of the chromosome to the crossover point is copied from one parent; the rest is copied from the second parent.

Parent A	1	1	0	1	*0*	*0*
Parent B	*0*	*1*	*1*	*1*	1	0

Crossover Point

Offspring 1	1	1	0	1	1	0
Offspring 2	*0*	*1*	*1*	*1*	*0*	*0*

Figure 2.13: Example of a single point crossover with a binary encoding

(2).Two or more point cross over: Two crossover points are randomly selected. This method copies string from the first parent, starting from the beginning of the chromosome to the first crossover point and from the second crossover point to the end of the chromosome. The remainder is copied form the second parent.

Parent A	1	*1*	0	*1*	0	0
Parent B	*0*	1	*1*	1	*1*	*0*

Offspring 1	1	1	0	1	0	0
Offspring 2	0	1	1	1	1	0

Figure 2.14: Example of a Three-point crossover with a binary encoding

C- Mutation

As with biological systems the mutation is manifested with a small change in the genetic string of the individuals. In the case of artificial genetic strings, the mutation is equal to a change in the elementary portion (allele) of the individuals' code. The mutation takes place with characteristics different to those that the individuals had at the beginning, characteristics that didn't possibly exist in the population.

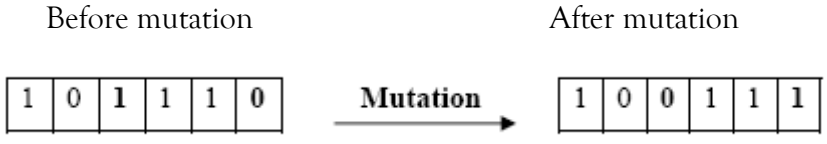


Figure 2.15: Example of mutation operator

The genetic algorithms seek their goal recurrently (by generation), evaluating each individual's aptitude in the object function which is in fact the optimization approach.

2.4.4 The Fundamental Theorem of Genetic Algorithms

A genetic algorithm is constructed by stochastic operators, and its robust search ability is based on the theorem depicted in [11, 12], which states, "short schemata of low order with aptitude above average, exponentially increase its number by generations ", this is:

$$\mathrm{m}(H,t+1) \geq \mathrm{m}(H,t)\frac{\mathrm{f}(H)}{f_{avg}}\left[1-p_c\frac{\delta(H)}{l-1}-\mathrm{O}(H)\,p_m\right]$$

where $m(H,t+1)$ and $m(H,t)$ are the schemata number H in the generation $t+1$ and t respectively, $f(H)$ is the average aptitude value of the strings that is included on the schemata H, f_{avg} is the total population's average aptitude value, l is the total string length, $\delta(H)$ is the schemata length from H, $O(H)$ is the schemata order from H, p_c is the crossover probability and p_m is the mutation probability.

2.4.5 Convergence conditions in GA

In any GA the generation process must proceed until a certain termination condition is reached. Researchers have used various convergence conditions, some of these convergence conditions are:

1- Until no further improvement in the population occurs [2, 11]

2- Until all offspring in the population are replaced [11]

3- Until there is very little variation within the population itself. [11]

2.5 Summary of Research

Studies in the field of layout planning have commenced as early as 1970's. Research has evolved from the development of heuristic models and expert systems to the formulation of analytical models having a precise optimization goal.

Various optimization tools have been used in layout planning. Traditional optimization tools like linear programming have given way to artificial intelligence techniques like neural networks and evolutionary or genetic algorithms.

CHAPTER-3

RESEARCH APPROACH & THEORETICAL BACKGROUND

In this chapter, three main topics will be discussed. The first section discusses about the greater domain of plant layout problem (PLP) and the plant facilities re-configuration problem. The next part shall see the suitability of using CAD and GA platforms for layout problems. Thirdly we move to see the theoretical background of objective functions that must be used to solve layout re-configuration problems.

3.1 The Plant Layout Problem

The solution of any size and type of plant layout problems could be facilitated by using a systematic and logical approach. An early pioneer in this area was Richard Muther, developer of Systematic layout planning (SLP) methodology. [3]

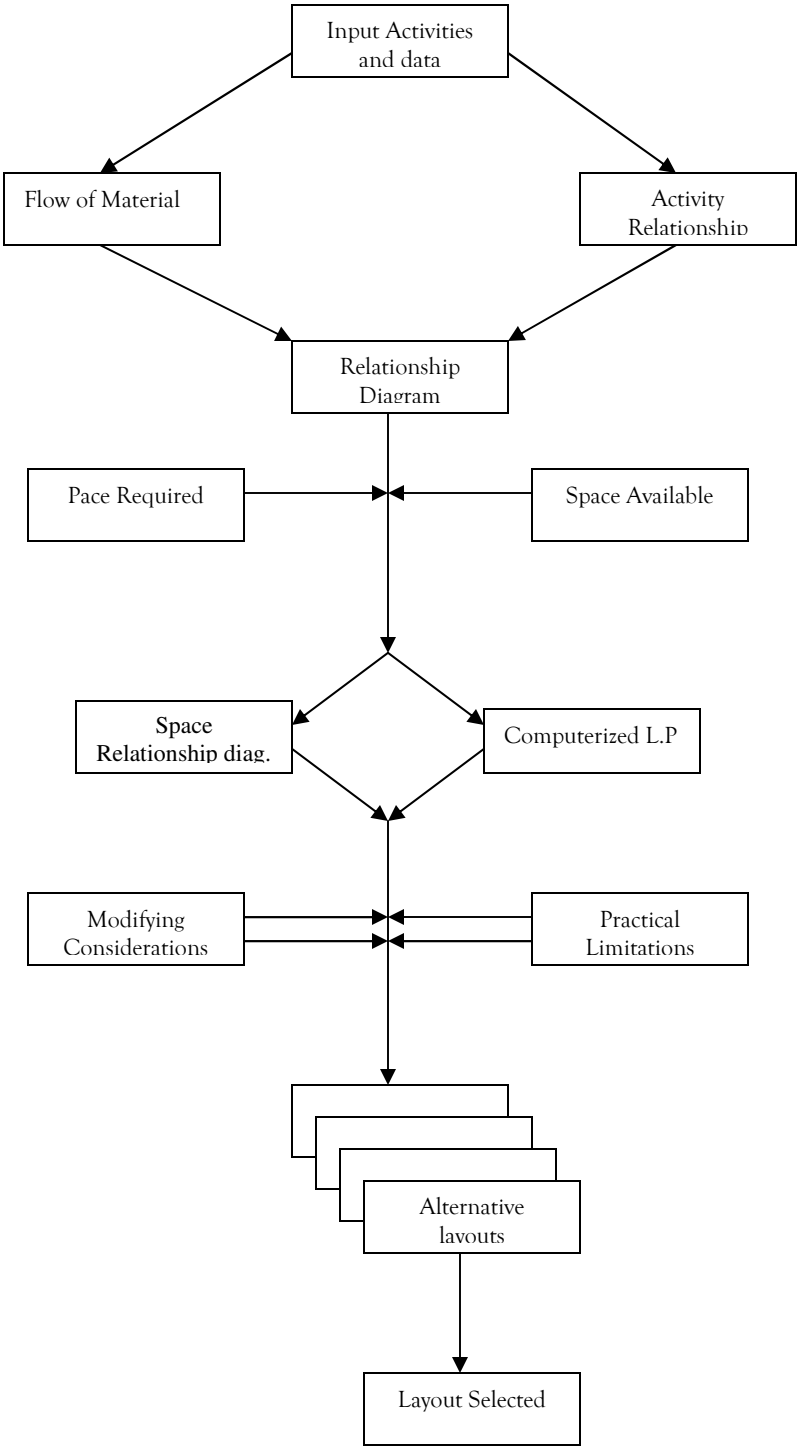


Figure 3.1: The systematic layout planning procedure

The systematic layout planning procedure can also be used for one of the domains of the plant layout problem i.e. re- configuring the presently installed layout with a slight modification. This is actually the focus area of this research paper.

As we see from figure 3.1, we normally come to the search stage where a number of alternative layouts should be considered to select the optimum layout. Here the more the number of alternative solutions (layouts) to select from, the better optimum solution we will have. It is very clear that generating more number of alternative solutions manually is very much tiresome. So the question is that, how can we generate a lot of alternative layouts and also evaluate them to have a better optimum solution?

It is here that the GA is supposed to work best i.e. in the *search and evaluate* stages of the systematic layout planning procedure. [5]

To re-configure the presently installed layout, the system should detect the available space as well as the practical limitations for assignment of facilities over the existing factory floor through AutoCAD application interfaced with VB6. [4, 10]

3.2 Suitability of AutoCAD and GA for Layout Re-configuration Problem

Computer Aided Design platforms experienced great advances during the late 80's. Their use in various engineering disciplines became inevitable. In the mechanical engineering branch, CAD software started off in use in the design stage as a drafting tool [4].

CAD could be used for plan plant layout problems, as adoption of such systems allows easy and accurate visualization of the relationship between different facilities in the factory.

The layout re-configuration problem is evidently graphical in nature. Existing factory boundaries, aisles, Reconfigurable and non reconfigurable facilities all occupy space and have distinct shape. Thus, the need to represent the relationship between all the aforementioned entities in some sort of graphical format is apparent. Tommelien & Zouein [21] were one of the earliest researchers that utilized CAD platforms for layout problems.

Conditions on the existing layout involve far more constraints, variables and uncertainties than those taken into consideration in most mathematical approaches for problem solving. Practically, the difference between optimum and near optimum solution is not that important, as even the optimum solution may require slight enhancements dictated by unforeseen layout conditions [2, 21].

The problem addressed in this paper is defined so that we want to find an optimal layout of a set of two dimensional objects with in the existing boundary such that

1. The objects should lie with in the existing boundary
2. The objects can't mutually overlap
3. The material strip has a defined length L and width W

For the sake of simple representation and manipulation of objects, it is considered only rectangular shapes which can coincide with rectangular two dimensional grid structures.

The first and most important thing in using the genetic algorithm for layout re-configuration problem is to choose the representation of the problem to be solved [5]. This is a crucial point since on the representation; other features of the algorithm will depend.

The most common way of representing potential solutions of layout problem would be a sequence of genes each coding the actual position of the individual object (facility) in terms of its coordinate values (x, y) with in the boundaries of the available space.[2, 5]

In the GA methodology, selecting an appropriate representation for a solution is termed "*coding*" [2]. It is considered one of the most important steps in formulating an accurate solution. Many approaches, like the permutation, binary and ordinal representations have been used for these types of problems. In this study, the permutation type is used, the other approaches being less suitable. For example, the string layout representation for 6 facilities can be shown as follows:

Table 3.1: String form representation of facilities

Facility	1	2	3	4	5	6
Location coordinate	(0,0)	(4,5)	(8,2)	(2,2)	(12,2)	(6,9)

The above string form representation of facilities location with in the specified bounding box (existing factory floor area) can be interpreted by the figure below as locations of each facility put on the grid area of the bounding box. Here it should be noted that the coordinate values represent the lower left corner of the facility having length L and width W. [2, 5]

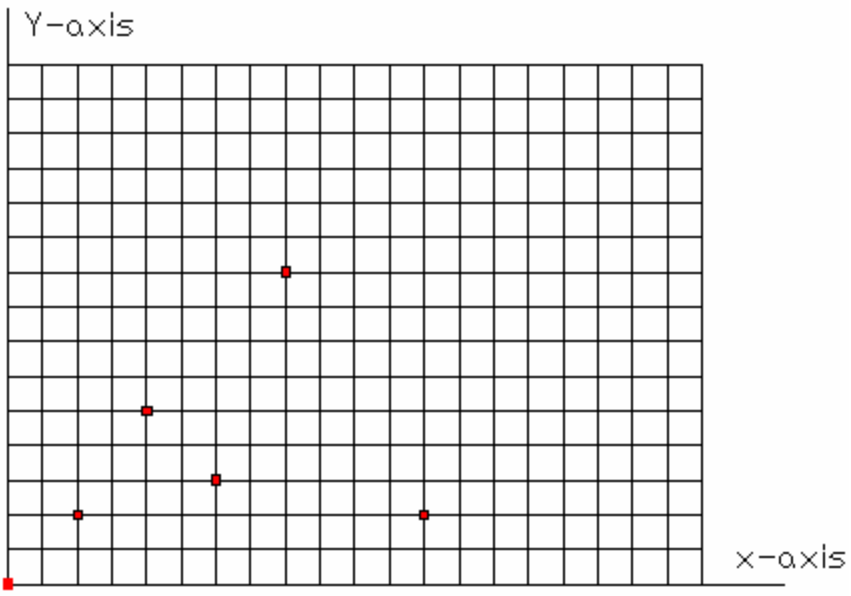


Fig 3.2: Actual representation of the facilities

The facilities have relative proximity weights between each other; this represents the level of interaction between facilities or the preference in having the facilities close to or apart from each other. The proximity weights used are *qualitative* proximity weights. These qualitative measures are then mapped to a *quantitative* weight that can be used in the optimization. [2]

The genetic algorithm generates random solutions (strings) as any GA would do. The solution is then evaluated for feasibility (for non-overlap between facilities) and the non-feasible solutions would be disregarded. [5]

3.2.1 Implementation Issues

The layout re-configuration problem as defined above can be seen as objects placement problem in which the algorithm generates sequences of objects and use some localization algorithm according to which the objects are properly placed on the grid. So the goal is to find an optimal placement of objects that will give the most optimum layout. [5]

The localization algorithm used in this paper scratches the partially covered grid for a free space where the next object or facility can be placed [5]. The algorithm starts at the lower left corner of the grid spaces (in the existing factory boundary) and randomize the coordinate points over the grid spaces for placement of the next object. If it fulfills the two

constraint conditions explained above the object will be put there, otherwise the algorithm disregards the point and starts to find another point for placement. The search goes on until any legal position for all of the objects at hand is found. [2, 5]

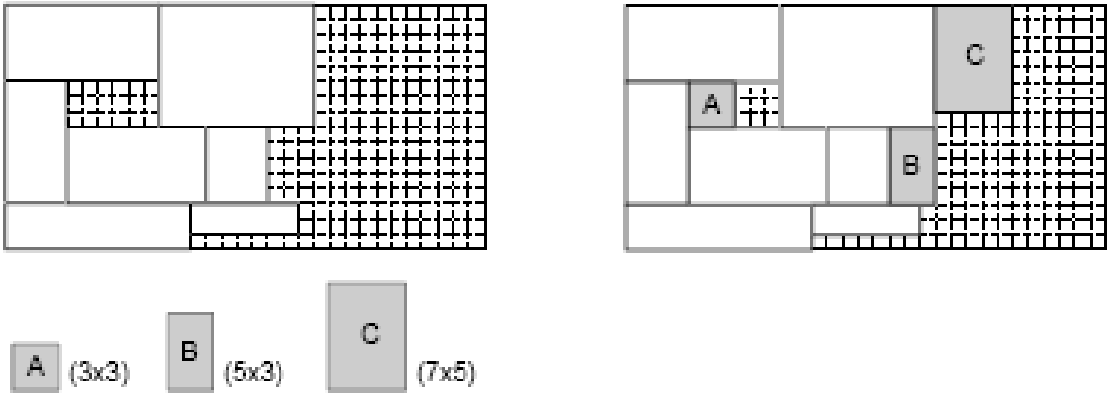


Fig 3.3: Localization algorithm

The function of the algorithm is illustrated for three different objects in the figure 3.2.

The first picture depicts the initial situation on the grid. We take all the spaces occupied by facilities to be re-configured as a free space as shown above by grid areas, where the area shown as blank is an area for fixed facilities. The second picture shows simultaneously the final states after inserting objects in their legal positions. From the example we can see the situation when it is possible to place the object A at position which is surrounded by other objects. Object B did not fit into that place, so the algorithm rejects this placement coordinate for object B and randomizes the grid coordinates to find another placement point for B. This way all the reconfigurable facilities at hand shall be put and finally we will have one candidate layout or solution space for the GA to optimize according to the objective function. [5]

In many optimization methods the GA move gingerly from a single point in the decision space to the next using some transition rule to determine the next point. This point to point method is dangerous because it is a perfect prescription for locating false peaks in multimodal search spaces. [1, 5] By contrast, when the GA is made to work from a rich database of points simultaneously, creates many peaks in parallel. GA's do not utilize gradient information. Thus, they are highly applicable to problems having non differentiable functions, as well as functions with multiple local optima. On the other

hand, if there is a specialized optimization method for a specific problem, then genetic algorithm may not be the best optimization tool for that application [1, 10].

Al-Tabtabi & Alex [1, 5] suggest that the use of GA in optimization is appropriate in the following circumstances:

1) Conventional statistical & mathematical methods are inadequate.
2) The problem is very complex, because the possible solution space is too large to analyze in finite time.
3) The additional information available to guide the search is absent or not sufficient, so conventional methods are not practical.
4) The solution to the problem can be encoded in the form of strings and characters.
5) The problem is large and poorly understood.
6) There is an urgent need for near-optimal solutions to use as starting points for conventional optimization methods.

Three of the aforementioned points make the utilization of GA in solving the plant layout problem very suitable. Firstly, when modeling a plant layout the available solution space is immense. The larger the available areas for placement and the greater number of facilities needed to be assigned the larger the feasible solution space becomes. Secondly, the solution to the problem can be encoded in the form of strings. This will be highlighted in chapter 4. Thirdly, finding a comprehensive solution to the layout problem is not as trivial as minimizing an objective function.

3.2.2 Fitness

In this case the candidate layouts are accessed according to a certain fitness measure which incorporates the requirements on maximal compactness of layout and non- overlapping of objects.[5]

3.2.3 Genetic operators

A very important roll in genetic algorithm applications is the production of off-springs of solution spaces or other candidate layouts from the existing ones by using different operators. The basic idea behind generating a new layout is given the two promising solutions; their offspring could hopefully inherit the best parts of each [5]. In this research paper the three types of recombination methods discussed in chapter 4 are used to

generate new off springs of layouts that could replace the worst two from the initial pool of population.

Here the researcher wishes to introduce the whole concepts of genetic algorithm through an example of a GA in action. The following example basically tries to illustrate the two principles of coding and selection. Suppose we are interested in finding the maximum value of the function

$y = -0.005x^2 + 1.5x + 10$ over the range $0 \le x \le 255$ as illustrated in Figure 3-4.

The first aspect to consider about a GA is its encoding, that is the transformation of the solution to a unique chromosome like structure. For simplicity we will use an 8 string binary coding to represent the solution space. Take for example the string 11001100, the binary representation for the decimal number 230.

The binary number is translated into its decimal equivalent as follows:

128	64	32	16	8	4	2	1	Coding:
1	1	0	0	1	1	0	0	= 128+64+8+4 = 230

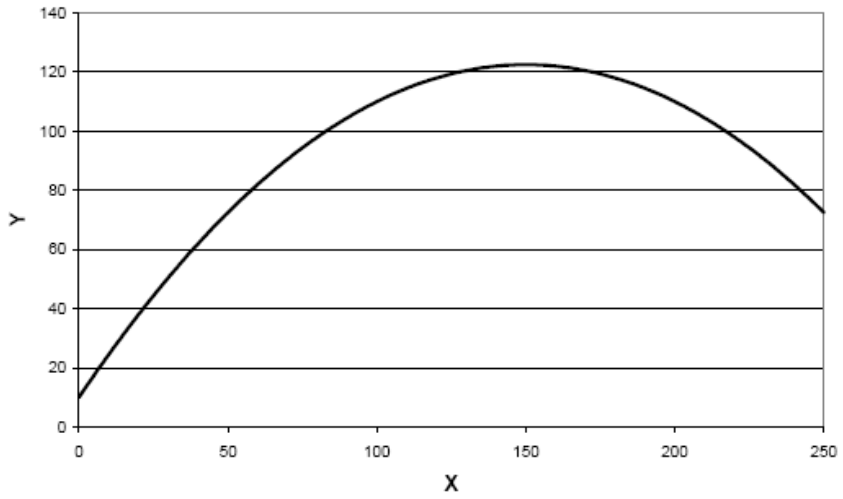


Figure 3-4: Graph of $y = -0.005x^2 + 1.5x + 10$

Any GA follows the following simple algorithm

Begin

Generate a new population of solutions

While terminating condition is not met

DO

Evaluate the solutions

Select the better solutions

Recombine solutions using genetic operators

END

Initially the GA must start with a randomized initial population of solutions. Suppose we will randomly choose 10 initial solutions. For each solution or *string* the GA evaluates its objective function. In column (5) we evaluate the probability of selection in the next population such that a solution's probability to be selected is directly proportional to the value of its objective function. P_{select} is sometimes referred to as the solution's *relative fitness*. In column (6) the expected count for the current solution is calculated by multiplying p_{select} by the number of solutions N, while in column (7) the actual count is calculated by rounding off column (6). The selection is made such that fitter solutions have a higher probability of being reselected. On the other hand, bad solutions are eliminated from the population. It is in this manner that GA's mimic the process of natural selection and survival of the fittest.

Table 3-2: A genetic algorithm by hand

Solution # (1)	String (2)	X (3)	y = f(x) (4)	$p_{select} = f_i / \sum f$ (5)	$Count_{exp} = p_{select} * N$ (6)	$Count_{act}$= (7)
1	00111001	57	79	0.11	1.1	1
2	00011110	31	52	0.073	0.73	1
3	11101111	223	95	0.132	1.3	1
4	00110001	49	71	0.099	0.99	1
5	00001001	9	23	0.032	0.32	0
6	00000110	6	19	0.02	0.2	0
7	11010101	213	103	0.144	1.44	1
8	10010110	150	122	0.17	1.7	2
9	00011010	26	45	0.06	0.6	1
10	01011111	95	107	0.15	1.5	2
Sum			717	1	10	10

As it can be seen, the objective function is evaluated several times during any one generation. Thus the evaluation process must be relatively fast. As the members of the population reproduce, their offspring must be evaluated in parallel. After the selection process the solutions are recombined using the common genetic operators, namely *crossover* and *mutation*. They will be discussed further in detail in chapter 4.

3.3 The Objective Function: Relative Weights vs. Cost Data

In almost all optimization approaches, when performing a static layout planning, the objective function to be minimized, takes the general form: [13]

$$\text{Min: } \sum_{i=1}^{p-1} \sum_{j=j+1}^{p} W_{ij} * d_{ij} * C_{ij}$$

Where:

P= Total number of fixed and temporary facilities present.

$d_{i,j}$ = Distance between facilities i and j

C_{ij} = the cost of material handling per meter per hour

The term $W_{i,j}$ can represent one of the following:

1) Material handling cost data of some sort or
2) A relative proximity weight that reflects the required closeness between any two facilities.

In all optimization problems the clear identification of a *goal* to attain is essential. Thus, the first representation has the clear objective of minimizing total material handling costs between facilities. The objective is not as apparent when using the relative weight representation. With the second representation one might argue, "What exactly are we trying to achieve?"

Several scales were adopted in order to facilitate the verbal representation of the proximity weight. The main advantage of using the relative weight representation is the great difficulty in obtaining accurate inter-facility material handling cost data. Using a relative proximity weight may be much easier for the layout designer to provide. So with this data the layout planner can have information about flow of materials and activity relationship.

One of the common proximity weight representations used in industrial facility layout planning is shown in Table 3-3. [2, 3, 5]

Table 3-3: The six value closeness relationship values used in industrial facility layout planning

Desired relationship between facilities	Proximity weight
Absolutely necessary (A)	81
Especially important (E)	37
Important (I)	9
Ordinary closeness (O)	3
Unimportant (U)	1
Undesirable (X)	0

On the other hand, when performing layout re-configuration, the objective function takes the general form [8, 16, 17]:

$$\text{Min: } \sum_{i=1}^{p-1} \sum_{j=j+1}^{p} W_{ij} * d_{ij} * C_{ij} + \text{Relocation cost}$$

Relocation costs of facilities are more easily quantifiable than inter-facility material handling costs. Formulating a relative relocation weight to be used with the proximity weight is quite complex. So in this research, the objective function to be minimized takes the general form:

Total layout Cost = Material handling Cost (T.C.) + Relocation Cost (R.C.)

To sum up the research approach, layout re-configuration problem is one of the domains of the greater plant layout problem. During floor reconfiguration problems, the space available and practical limitations on floor area of the factory will be accessed by the programmable feature of AutoCAD where as the activity relationship and flow of materials between facilities are expressed qualitatively by proximity weight and space required by each facility is assumed to be known by quantitative analysis in the abstraction phase of SLP. Having these inputs the GA shall generate alternative layout solution and evaluate them for the objective function to select the best out of them.

CHAPTER- 4

SYSTEM DEVELOPMENTS

This chapter provides details of the automated system used for the layout re-configuration of floor facilities. The chapter begins with an overview of the system structure as a whole. Afterwards the various modules comprising the system are thoroughly explained. These include the geometrical data detection module, the constraint satisfaction module and the main GA-based optimization procedure module. The coding for all modules is provided in the appendices of this thesis.

4.1 System Structure

The reconfigurable layout planning system is comprised of three main components:

- An input facility that incorporates various types of data needed for the layout planning task.
- An optimization engine based on the concepts of genetic algorithms.
- An output facility that utilizes the interface features of the VB and AutoCAD application.

The automated system utilizes some amounts of data. Data used in the system can be grouped into four major categories (Table 4-1), namely temporary facility data, existing floor geometrical data and facility cost data.

Table 4-1: Description of the main data types required in the model

Data	Description
Reconfigurable facility data	Reconfigurable facility requirements and the expected sizes of these facilities
Existing factory floor geometrical data	AutoCAD drawings representing the layout of fixed facilities.
Facility cost data	Inter-facility material handling costs along with the expected cost for relocating reconfigurable facilities.

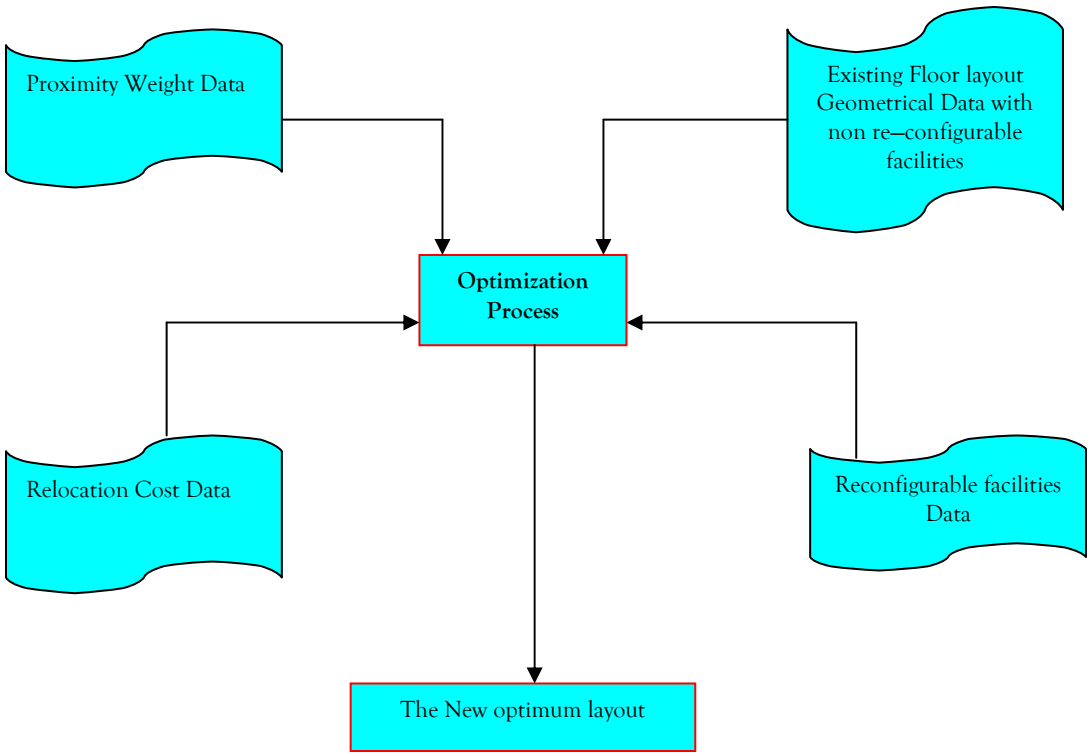


Figure 4.1: Detailed system structures

Genetic algorithms are used to perform the optimization process. Following the optimization process, the system delivers a CAD drawing depicting a particular floor layout with all reconfigurable facilities placed in their optimal positions. The detailed system architecture is illustrated in figure 4-1.

4.2 Space Identification

The functionality of the optimization engine largely depends on identifying the specifics of the CAD drawing (i.e., existing floor boundaries, fixed facilities, and obstacles/aisles). Accurately identifying the available space on factory floor for assigning the reconfigurable facilities is essential in order to yield a feasible solution. Available space on floor is detected through the algorithm explained hereafter. The main concept in space detection is that of space discretization, that is the division of space into an orthogonal two-dimensional grid. This grid is then coded, each grid cell having a unique (X, Y) coordinate as illustrated in Figure 4-2. Regardless of the factory geometry, the system is able to capture all geometrical

data lying within the factory boundaries, after which removal of space occupied by permanent facilities occurs.

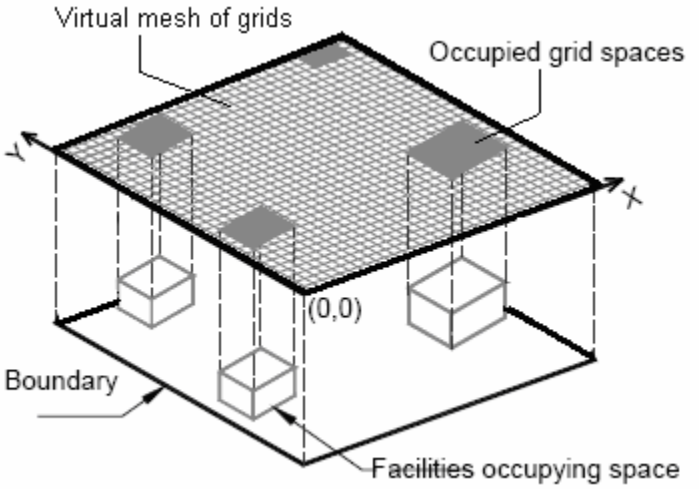


Figure 4.2: Details of space descritization

The available space in the factory floor is detected using the capabilities of the CAD application. A Visual Basic program module can be automated with AutoCAD to perform this function. Generally, two steps must be accomplished, first the factory boundaries are identified, second fixed facilities and obstacles (e.g. aisles) occupying space are identified.

4.2.1 Identification of enclosed factory floor space:

In order to perform this task the user must identify the factory floor boundaries (must be an AutoCAD lightweight poly-line) and any point lying inside the factory boundaries. Performing this task proceeds as follows:

- Using the rectilinear coordinates of the shop boundary's vertices, the VB module identifies the edges of the shop boundary as a set of equations:

  $Y = A_1$, $Y = A_2$, $X = B_1$, $X = B_2$

- Using a point inside the boundary, the VB module identifies the edges of the shop boundary as a set of inequalities:

  $Y >= A_1$, $Y <= A_2$, $X >= B_1$, $X <= B_2$

- By toggling through coordinates of all points inside the shop boundary's "Bounding Box" (Figure 4-3), the VB module chooses only those points that satisfy all linear inequalities simultaneously.

- Feasible grid squares are stored in 2 arrays: Available X ( ) and Available Y ( ) such that any point "j" has coordinates (Available X(j), Available Y(j))

4.2.2 Identification of fixed facilities and obstacles

Till this stage no account has been made for obstacles or fixed facilities.

Obstacles are defined as facilities with fixed positions having no closeness relationship with other facilities. Fixed facilities on the other hand have fixed positions but maintain closeness relationship with other facilities. In order to perform this task the user must identify all obstacles and fixed facilities present in the boundary. Performing this task proceeds as follows:

- Grid spaces occupied by the obstacles are removed from the arrays Available X and Available Y.
- Grid spaces occupied by the fixed facilities are removed from the arrays Available X and Available Y. The coordinates of their centroid are stored for future use in the optimization procedure.

After these two sub-tasks are accomplished, available space, obstacles and fixed facilities are translated into a set of X, Y coordinates. These coordinates will be utilized in the optimization procedure during the layout of the re-configurable facilities.

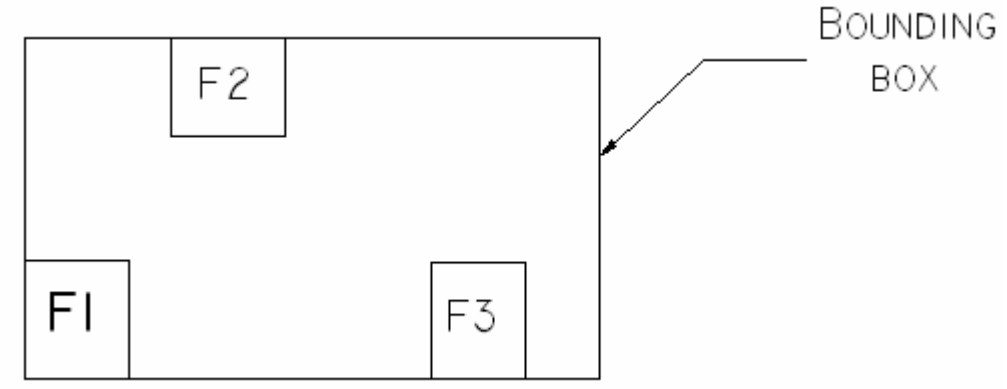


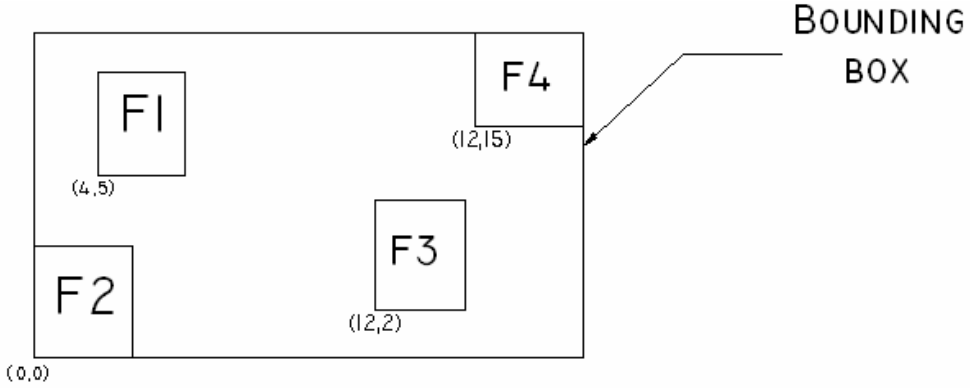
Figure 4-3: Bounding box of a polygon

4.3 GA String Coding

The optimization procedure itself depends primarily on GA string coding. As discussed in chapter 3, the coding of a solution to a string is one of the most important aspects of

success for GA optimization. The coding scheme of any GA is the essence of its success. The coding scheme used in the presented model is the one that is adopted by Hegazy & Elbeltagi (1999) [2]. As their model depended largely on utilizing a set of grid cells to represent space; each grid cell was uniquely coded based on the location of the row and column in which the cell is located. Factory facilities were then referenced through these developed codes.

The coding of a certain solution representing the assignment of four re-configurable facilities is illustrated in Figure 4-4.



Solution Encoding for the above Layout alternative

	F1	F2	F3	F4
X	4	0	12	15
Y	5	0	2	15

Figure 4-4: GA string encoding of 2D space

4.4 Constraint Satisfaction

Geometrical constraints are vital in the layout process. It is of utmost importance that re-configurable facilities be placed (1) Inside the boundaries and (2) In such a manner that no overlap occurs between any two re-configurable facilities or between re-configurable and fixed facilities.

To achieve the satisfaction of geometrical constraints, two main modules are utilized, namely *"Check-Space"* and *"Check-Overlap"*. These modules rely on 5 main variables in their operation as shown in Table 4-2.

For each re-configurable Facility

Check-Space (Xmin, Ymin, FacilityX, FacilityY)

Check-Overlap (Xmin, Ymin, FacilityX, FacilityY)

Next Facility

Where: *Xmin* and *Ymin* are gene values for current facility (coordinates of bottom left corner of facility) and *FacilityX* and *FacilityY* are the dimensions of the temporary facility in the X and Y directions respectively.

Table 4-2: Main variables required in the constraint satisfaction procedure

Variable Name	Description
Available X( )	Available X-coordinate for assigning Reconfigurable facilities
Available Y( )	Available Y-coordinate for assigning temporary facilities
Occupied X( )	Space currently occupied by reconfigurable facilities
Occupied Y( )	Space currently occupied by reconfigurable facilities
Reserved Pts	Number of grid units currently occupied by reconfigurable facilities

4.4.1 Check Space module:

This module is a built-in function that makes sure that any re-configurable facility

(1) Lies inside the boundaries and

(2) Does not overlap with any fixed facility or space obstacles.

This function requires as input 4 variables; [Xmin, Ymin, FacilityX and FacilityY]. It provides a Boolean type true/false output.

In its operation it toggles through all grid coordinate occupied by the facility and compares them with the arrays AvailableX and AvailableY.

If any (X, Y) of facility ⊄ Available(X, Y) then Check-Space = False
Else Check-Space = True

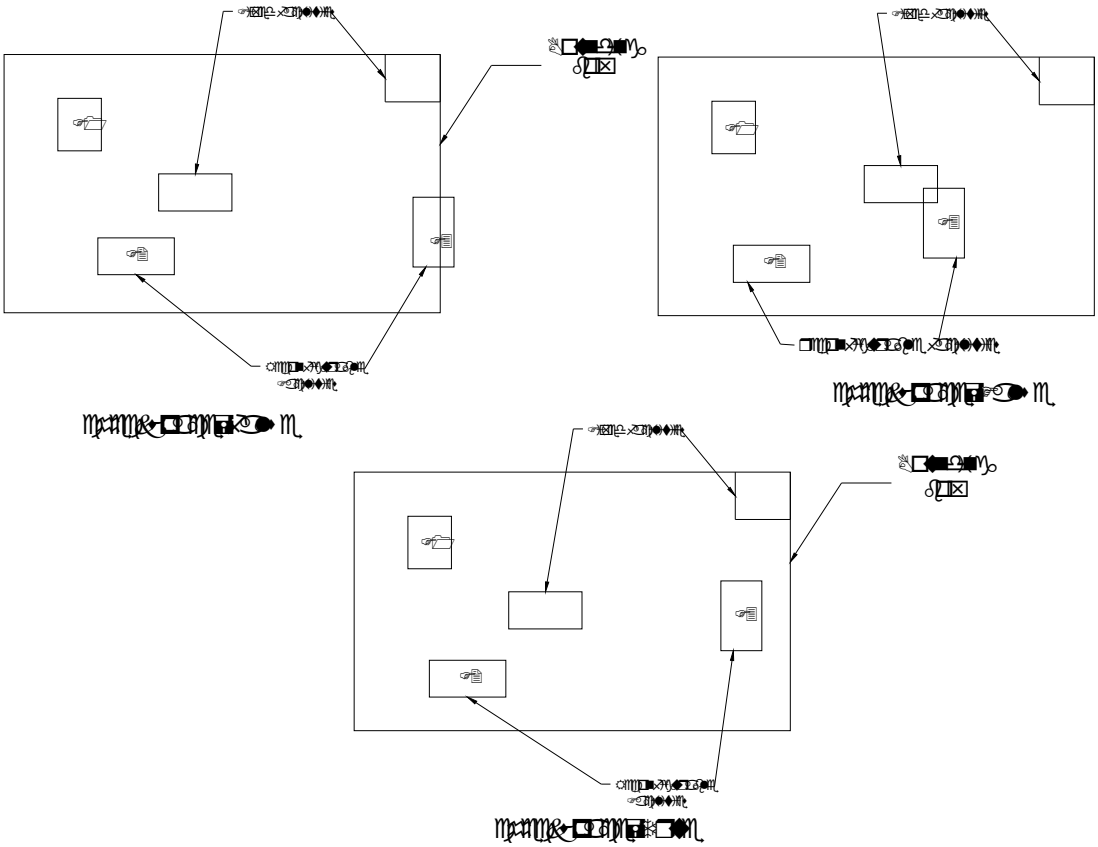


Figure 4-5: Functionality of the *Check space* module

4.4.2 Check Overlap module

This module is a built-in function that performs two sequential tasks.

a) Makes sure that the facility being checked does not occupy space already being reserved for another facility that has been assigned on site.

   If any (X, Y) of facility C Available(X, Y) then Check-Overlap = False
   Else Check-over-lap = True

b) If no overlap occurs space is reserved for the facility

   For all (X, Y) of facility, Occupied(X, Y) = Facility(X, Y)

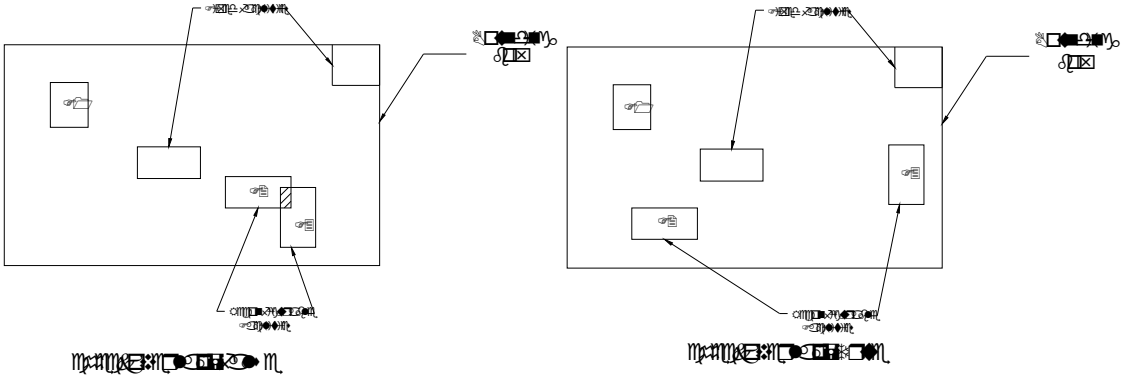


Figure 4.6: Functionality of the check overlap module

4.5 Inter-Facility Cost Matrix

The inter-facility material handling cost matrix contains the most important optimization relevant data. Using these numbers, the optimization procedure will assign facilities close to or far from one another. The proximity weight or cost data is represented in a lower triangular matrix as shown in Table 4-3 for six reconfigurable / temporary facilities and three fixed facilities. As shown in the table, no inter-facility costs data exists for fixed-to-fixed facilities.

Table 4-3: Inter-facility cost matrix for 6 re-configurable facilities and 3 fixed facilities

Temporary Facilities	F1								
	F2	W_{12}							
	F3	W_{13}	W_{23}						
	F4	W_{14}	W_{24}	W_{34}					
	F5	W_{15}	W_{25}	W_{35}	W_{45}				
	F6	W_{16}	W_{26}	W_{36}	W_{46}	W_{56}			
Fixed Facilities	F7	W_{17}	W_{27}	W_{37}	W_{47}	W_{57}	W_{67}		
	F8	W_{18}	W_{28}	W_{38}	W_{48}	W_{58}	W_{68}	N/A	
	F9	W_{19}	W_{29}	W_{39}	W_{49}	W_{59}	W_{69}	N/A	N/A
		F1	F2	F3	F4	F5	F6	F7	F8	F9

Temporary Facilities — Fixed Facilities

4.6 Optimization Procedure

Details of the objective function to be optimized and the GA procedure are fully described in the following section. The flowchart for the GA is illustrated in Figure 4-8.

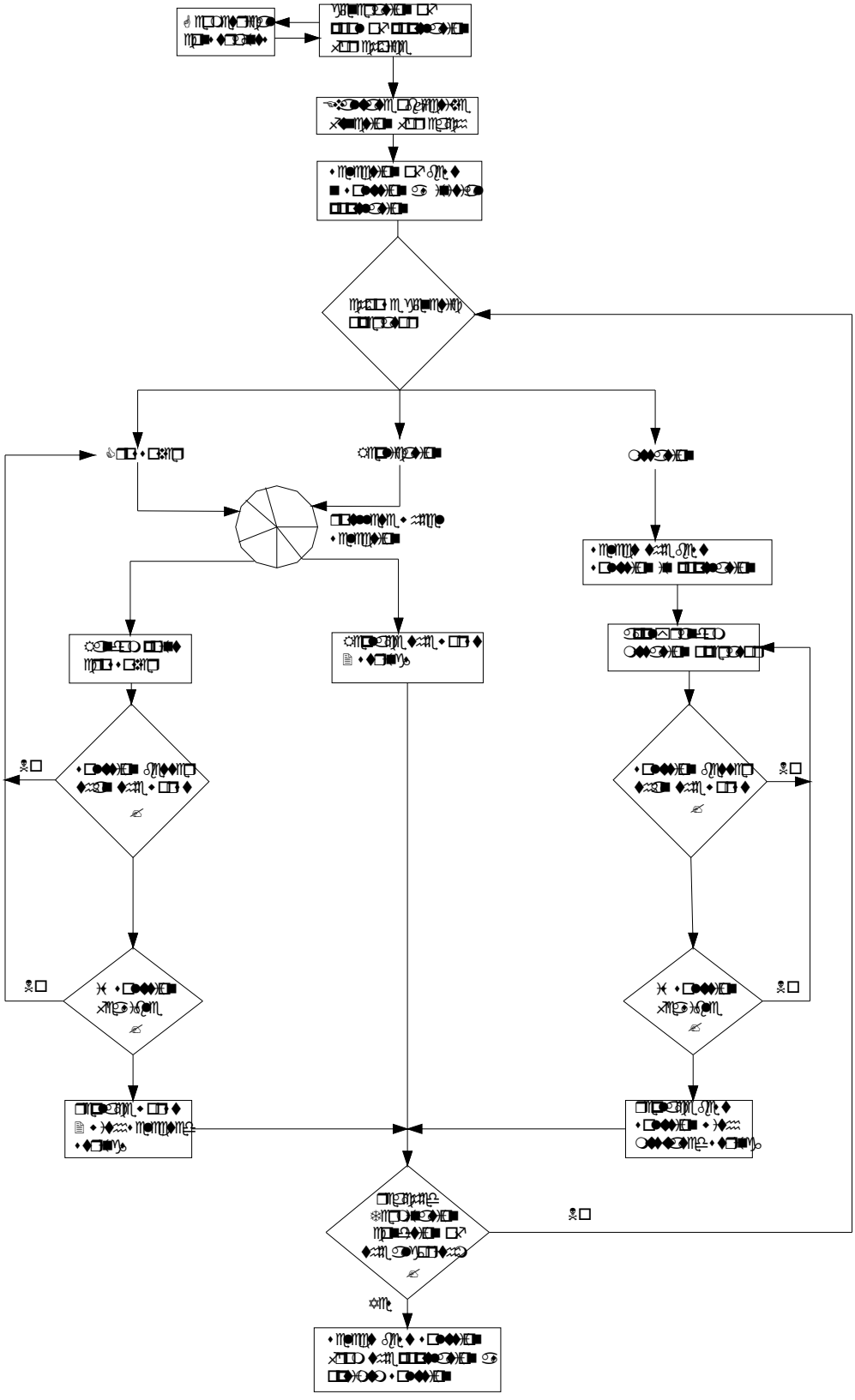


Figure 4-7: Genetic Algorithm Flowcharts

As shown above, first the generation of pool of population is performed by taking the available grid areas. This is made by a visual basic technique to randomize a given points to locate a coordinate for all the reconfigurable facilities with in the existing area. The generation of the initial set of population is made to have N number of candidate layouts for evaluation with the objective function.

After the evaluation of the initial set of population, the GA will move to select the best n solutions out of n such that n<N by genetic recombination (GA operators). As The GA operators used in this research are crossover, mutation and replication. These operators will take an action on a selected group of two individuals step by step to produce an off spring hoping that the children of the parent candidate layouts may have a better solution to the objective function. If it is so the children will replace the two of the worst from the initial set of population and if it is not, they will be discarded. A selection procedure code is made to select on of the operators to act at a time. In the crossover and replication operators the roulette wheel selection principle is used to select two strings but in the mutation operator the selection of the best solutions from the populations is done based on the value of the objective function. In the mean time, the feasibility of the new solutions is checked (for check space and check overlap modules) before replacements of the parents have taken place.

Finally when the termination condition of the recombination process is reached, the selection of the optimum layout from the current set of population is made by comparing the objective function values.

4.6.1 Initialization of Population

Any GA procedure starts with an initial population of solutions. The number of initial solutions generated influences the GA. It is known that increasing the population size has the following effects on the GA:

1- Increases the time required for generating a new population.

2- Causes a very slow convergence rate.

3- Causes the GA to reach more optimum solutions.

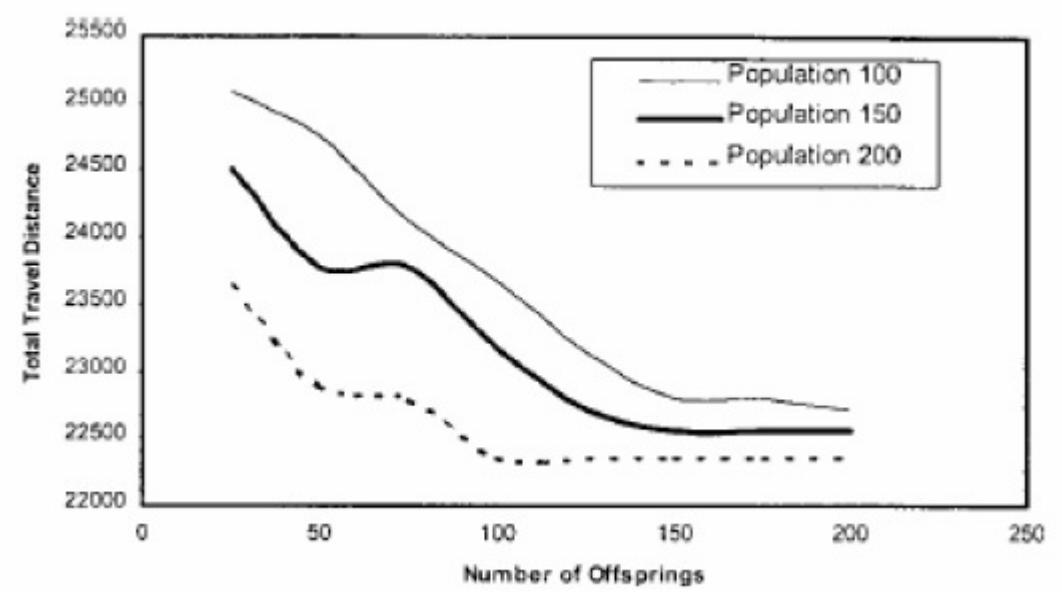


Figure 4-8: Effect of population size on optimum solution

In order to assist the GA in its blind search, a slight enhancement has been proposed. Instead of working with a very large population throughout the GA, the initial population is selected as the best 'n' solutions from an initial pool of 'N' solutions where n < N. Before running the GA, both the initial pool "N" and the required population size "n". Thus the GA benefits from the presence of a large initial population that assists its random search without paying the large computational penalty posed by dealing with a large population at each generation.

4.6.2 Objective function

The objective function that is evaluated in the optimization process is

Total Layout Cost =Material Handling Cost (MH.C.) + Relocation Cost (R.C.)

$$\text{MH.C} = \sum_{i=1}^{p-1} \sum_{j=j+1}^{p} W_{ij} * d_{ij} * C_{ij}$$

$$\text{R.C} = \sum_{i=1}^{p} R_i * d_{ij}$$

Where:

P = Total number of Fixed and Reconfigurable facilities present.

*W*i,j = Proximity weight between facility i and j

d i,j= Distance between facilities i and j

C_{ij} = the cost of material handling per meter per hour

R_i = Cost of relocating facility i

Generally, relocation cost of an industrial facility can be represented by the following equation:

R.C. = Fixed Cost + Variable Cost

Where the fixed cost represents those costs spent on dismantling, re-installment, delay and providing an alternative facility. Variable costs are usually attributed to hauling, and are a direct function of hauling distance. The cost of relocating a facility is not greatly dependant on the distance of relocation but on whether relocation has taken place or not. When considering factory floor facilities, fixed costs are far larger than variable costs. Accordingly, the following representation for relocation cost can be used:

$$\text{R.C} = \sum_{i=1}^{p} R_i * C_i$$

Where C_i is a binary variable that takes only 0 or 1

If facility i has been relocated $C_i = 1$

If no relocation has occurred $C_i = 0$

From the objective function we can simply put the decision model for re-configuration of floor layout facilities as follows:

If material handling cost of the existing layout >Total layout cost of the new layout then do the re-configuration.

Else use the existing layout.

4.6.3 GA Generations

The generation process proposed in this research is the traditional GA generation which moves from generation (i) to generation (i+1) via the generation of a new population. This approach moves from one generation to the other via the introduction of two new

offspring that would replace the worst two solutions in the population. In case the new offspring were not better than the worst solutions, they were discarded and two other offspring were chosen.

Elimination of the worst offspring meant that as each new solution is introduced, the population as a whole would improve. It also meant a chance for randomly bred offspring to even outperform the best solution in the population.

Generation of new offspring involves the three traditional genetic operators:

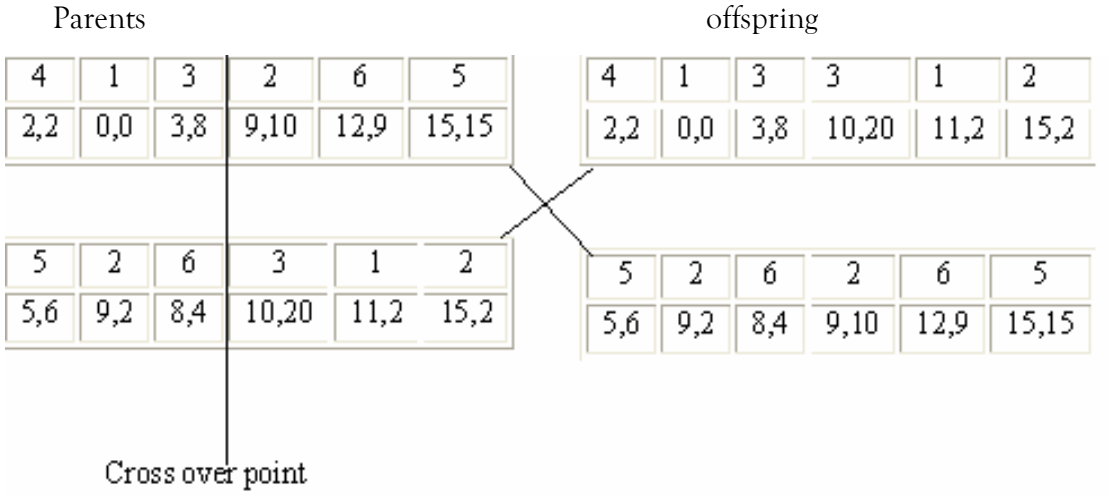
1- Replication

Roulette wheel selection is performed based on the fitness value for individual solutions. Two random solutions are chosen to replace the worst two solutions in the population. There is no need to check for the feasibility of the solutions, as no modifications were performed (crossover, mutation).

2- Crossover:

The same selection procedure is applied to select the parents that will be crossed.

Simple single-point crossover was used so as to minimize the disruption of the schemata.



After the crossing two checks are performed on the offspring:

1- Constraint satisfaction: To verify the feasibility of the new solutions.

2- Objective function improvement: To verify that the new solutions are not worse that those being replaced.

3- Mutation:

Mutation is used mainly to break current stagnation in improvement by introducing new genetic information into the population. It was noticed during the development of the system, that performing the GA without mutation led to solutions that required slight refinements to reach more optimum solutions. These refinements usually involved very small movements of one or more facilities in a specific direction.

The mutation operator randomly chooses the following:

1- The facility to be moved.

2- Whether the movement should be in the X or Y direction.

3- Whether the movement should be in the +ve or –ve direction.

And then applies a movement of 1 unit to the facility chosen and in the direction chosen. Similarly, the new solution is checked for constraint satisfaction and objective function improvement. If violated, the mutation procedure is repeated. The mutation operator flowchart is shown in Figure 4-9.

4.6.4 Convergence Condition

In this system, the generation process continues until the following convergence condition is reached:

$\Delta < Convergence$

Where:

$$\Delta = \frac{Max - Min}{Max}$$

Convergence: User specified tolerance (usually 5-10%).

Min: solution with a minimum objective function in current population

Max: solution with a maximum objective function in current population

4.7 Solution Representation

The optimum solution must be represented in a user-comprehendible form. The final step in our system is the physical representation of the optimum solution.

The system, via the *draw facility* module automatically opens AutoCAD application and draws the reconfigurable facilities in their optimum positions as depicted by the genetic

algorithm. Having the output in graphical form allows the designer to easily visualize the relationship between the fixed structures and reconfigurable facilities with in the factory floor space. Graphical solution representation proceeds as follows

Open CAD file representing the floor layout

For all reconfigurable facilities

Draw temporary facility in optimum position

Next temporary facility

Close CAD file

CHAPTER-5
THE AUTOMATED SYSTEM AND AN ILLUSTRATED EXAMPLE

In this chapter, the automated system, ALREP (Automated Layout Re-configuration Program) is presented along with an illustrated example that will demonstrate the various features of the automated system.

The system has been implemented through Microsoft visual basic 6.0. The AutoCAD and MS Excel interfaces with VB have been made possible through ActiveX automation. COM (Component Object Model) automation is used to communicate the visual basic program with the different objects in the AutoCAD and MS Excel environment.

The codes for various modules in the VB environment can be found in the appendices of this thesis.

5.1 System Input

The previous chapter discussed about the types of the data required as an input to the ALREP. These data can be broadly classified as:

- The re-configurable facilities data
- The existing factory geometrical data
- The inter facility transport cost data and
- The relocation cost data of reconfigurable facilities

In the automated system, the reconfigurable facilities data is input through the user window as shown in the figure 5.2. The user can make an input of reconfigurable facilities and their dimensions in the combo box provided for the names as well as for length and widths of these facilities step by step. The user can also remove and update the reconfigurable facilities from the list and input continues until all the facilities are entered.

The windows for all the data inputs will be accessed from the main system's window menu bar called data as shown in the figure 5.1.

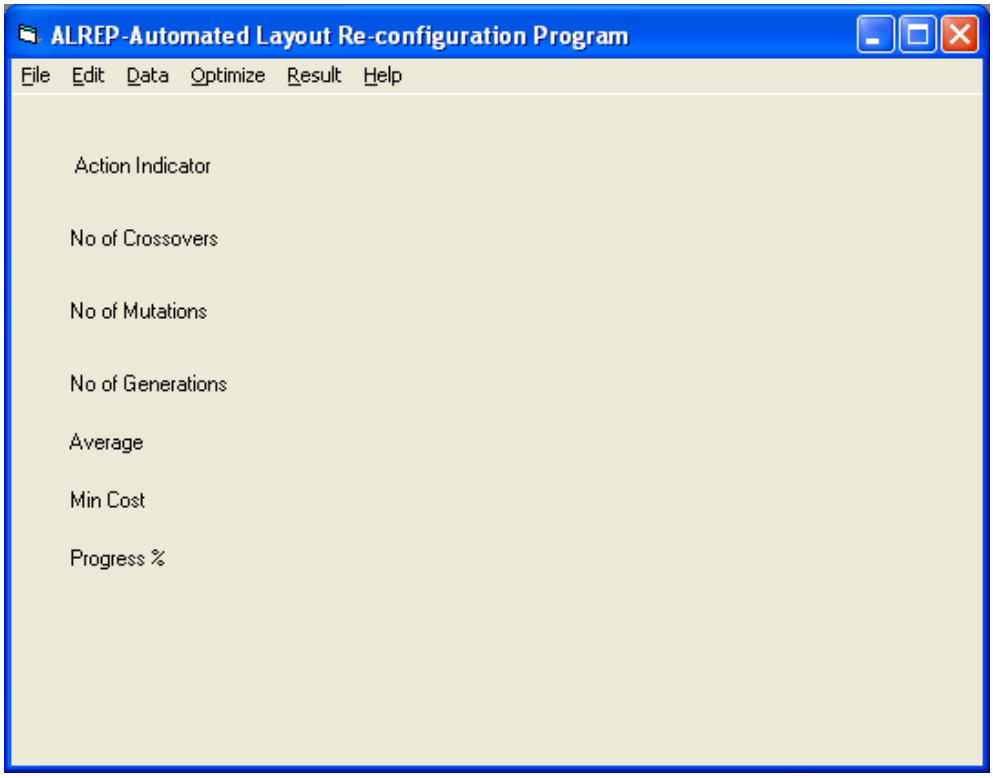


Figure 5.1: The system's main window

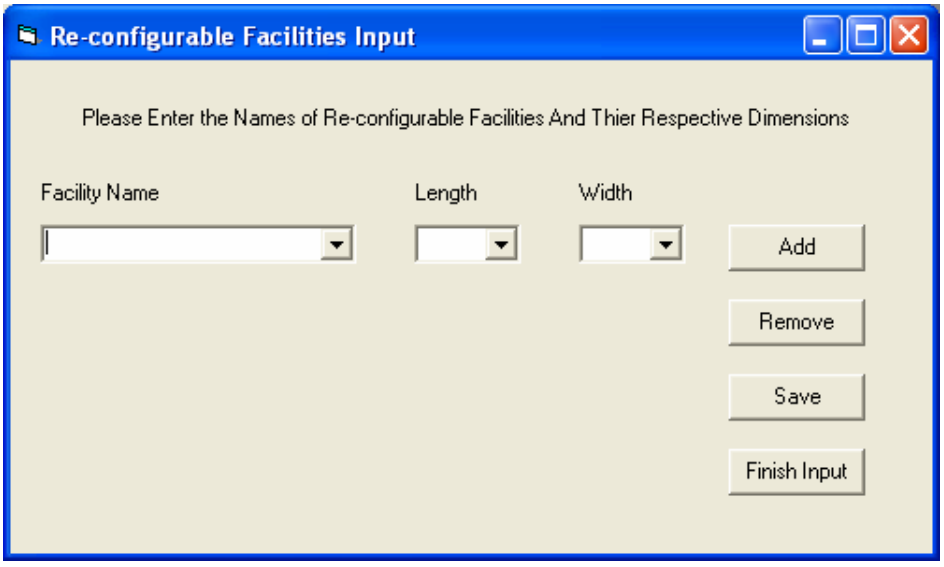


Fig 5.2: Window for input of reconfigurable facilities

The existing factory geometrical data input is made possible through the visual basic-AutoCAD referencing capabilities.

The automated system is capable of opening the existing CAD drawings that are done before to represent the factory boundary, fixed facilities and some obstacles. The system

detects the available factory space for reconfigurable facilities to be put on. The AutoCAD drawing should be appropriately prepared before the automated system began to process it. Fig 5.3 shows the AutoCAD interactive capability present in the automated system. Fig 5.4 shows the procedure for space detection operating on the Auto CAD environment.

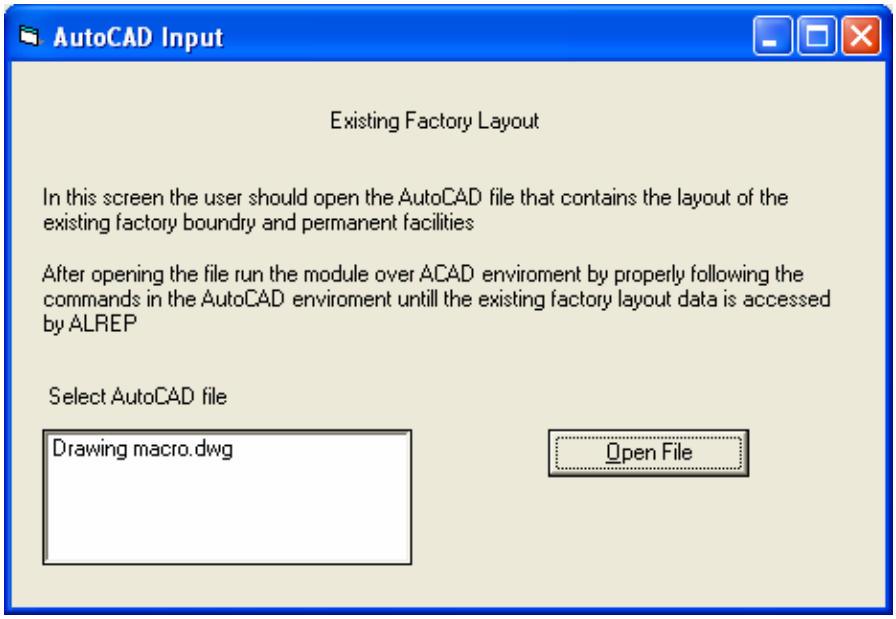


Fig 5.3: The System's window to interact with AutoCAD application

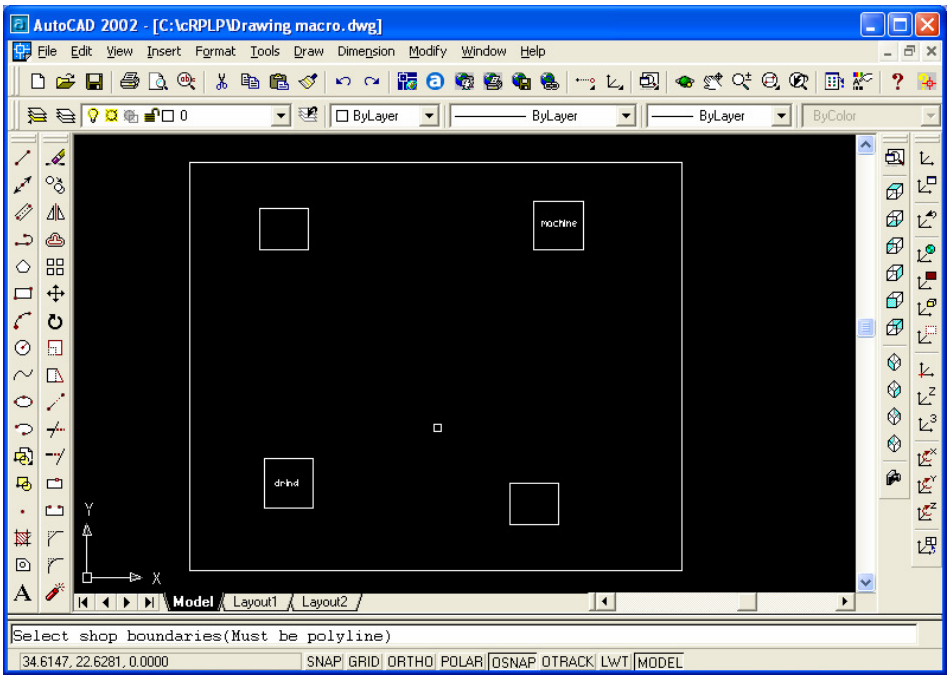


Fig 5.4: The AutoCAD environment being run by the system

The inter facility proximity cost data is made input through the visual basic- MS Excel referencing capabilities. Here also the automated system is capable of opening the existing MS Excel work sheet that contains the proximity weight between each facility including the fixed facilities.

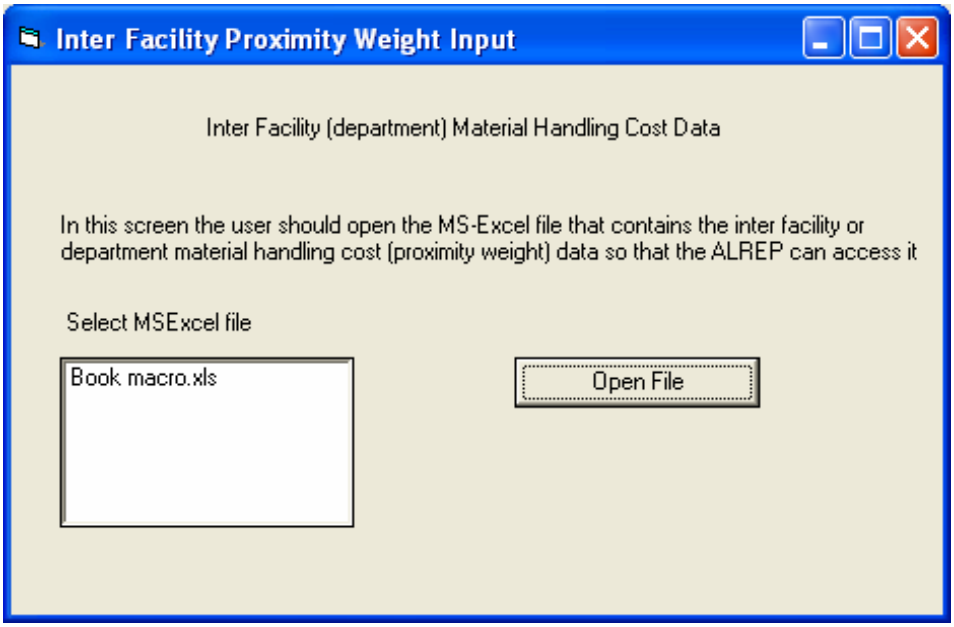
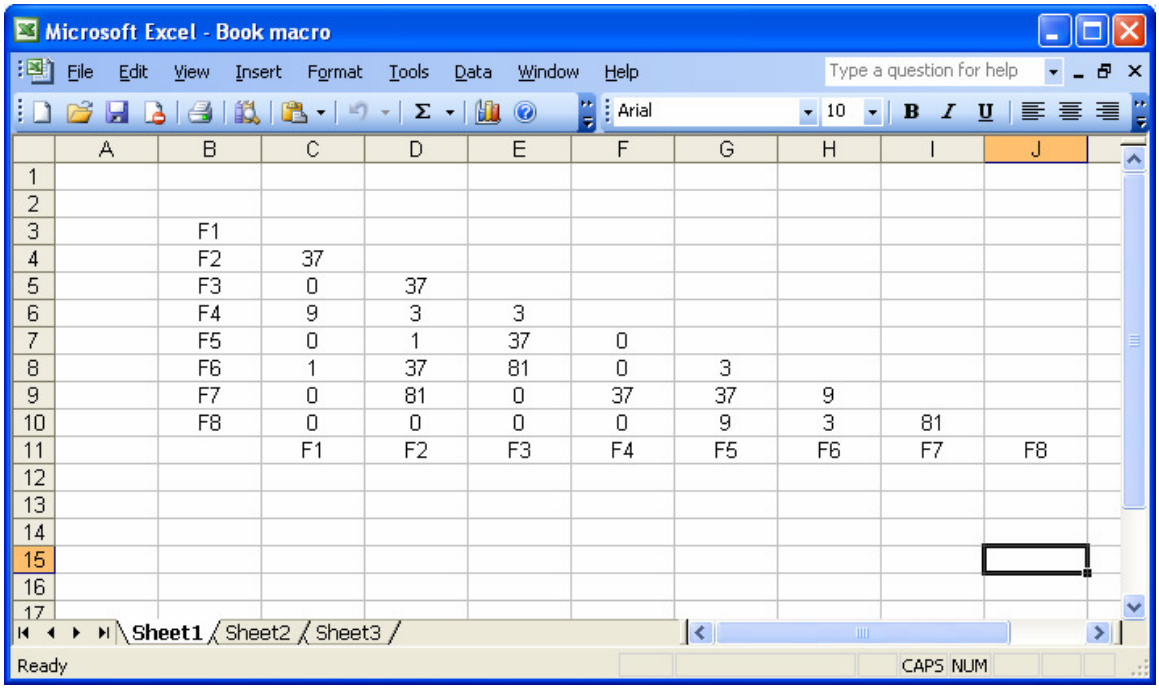


Fig 5.5: the System's window to interact with MS Excel application



	A	B	C	D	E	F	G	H	I	J
1										
2										
3		F1								
4		F2	37							
5		F3	0	37						
6		F4	9	3	3					
7		F5	0	1	37	0				
8		F6	1	37	81	0	3			
9		F7	0	81	0	37	37	9		
10		F8	0	0	0	0	9	3	81	
11			F1	F2	F3	F4	F5	F6	F7	F8
12										
13										
14										
15										
16										
17										

Fig 5.6: The MS-Excel environment being run by the system

The last data which is the cost per facility required for relocating a facility from one location to the other is made possible through the user window shown in the fig 5.7.

As it is shown the user can add the names of the facilities and the corresponding relocation costs for the reconfigurable facilities.

Here the user should follow the same order of inputting the facilities as he has followed in the reconfigurable facilities data and proximity weight data.

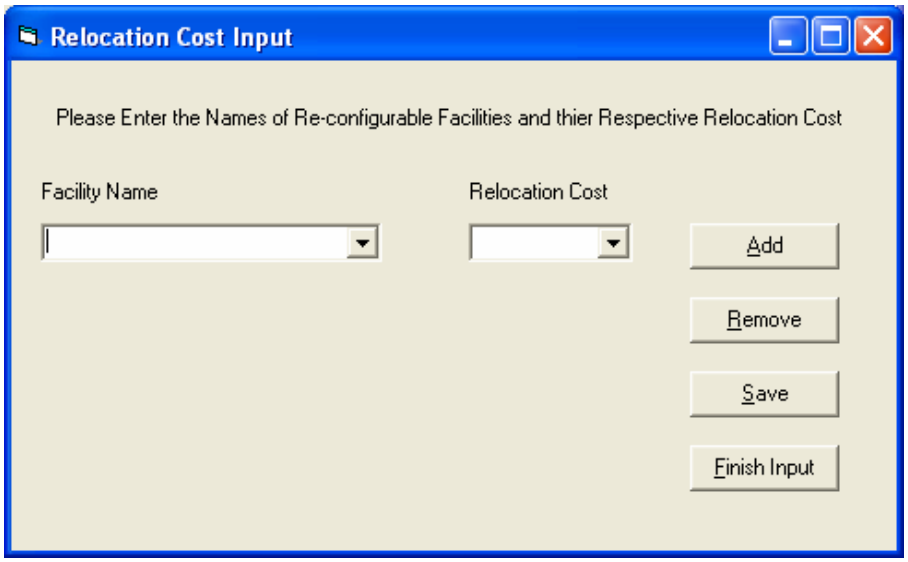


Fig 5.7: The system's window to input the relocation costs of reconfigurable facilities

5.2 System Optimization

After the above four types of data are properly made input to the system, the user shall turn to the system's main window and can start the optimization process by clicking over the optimize button in the menu bar.

The system's main window will show the different characteristics of the optimization process including the progress rate in percent.

5.3 System output

The result menu in the system's menu bar provides two distinct types of outputs, namely

1) A graphical out put and
2) A cost summary

The graphical output is presented through the AutoCAD environment.

The minimum material handling cost and the total cost which is the sum of material handling and relocation costs will be presented on the cost summary window.

5.4 Illustrated Example

The following simple example is used to demonstrate the systems capability and evaluate the output.

An ideal factory used for this example purpose has got the following characteristics with regards to the data requirement to ALREP.

A- The reconfigurable facilities and their dimension(m)

No	Facility Name	Length	Width
1	Cutting Department	6	6
2	Lathe machines Department	6	4
3	Polishing Machines Department	4	2
4	Inspection	6	4

And the fixed facilities

No	Facility Name	Length	Width
1	Press machines Department	6	4
2	Surface Treatment	6	6

B- The geometrical data or the existing factory /drawing

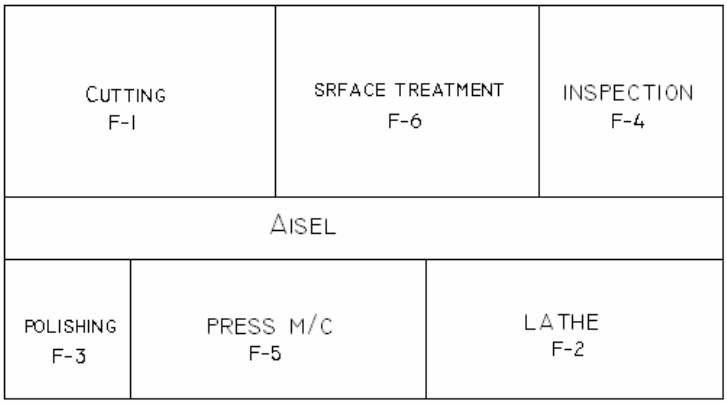


Fig 5.9: The existing factory layout

C- The inter facility transport cost data or proximity weight data

	F-1	F-2	F-3	F-4	F-5	F-6
F-1						
F-2	37					
F-3	9	0				
F-4	81	3	0			
F-5	0	81	0	9		
F-6	0	0	81	3	0	

D- The relocation cost data for reconfigurable facilities

No	Facility Name	Relocation cost
1	Cutting machines Dpt	600
2	Lathe Machines Dpt	800
3	Polishing	300
4	Inspection	100

5.5 Optimization Result

The output for the example presented above is shown in two ways as explained previously. For instance, the graphical representation is shown through AutoCAD environment like the one shown in the Figure 5.10.

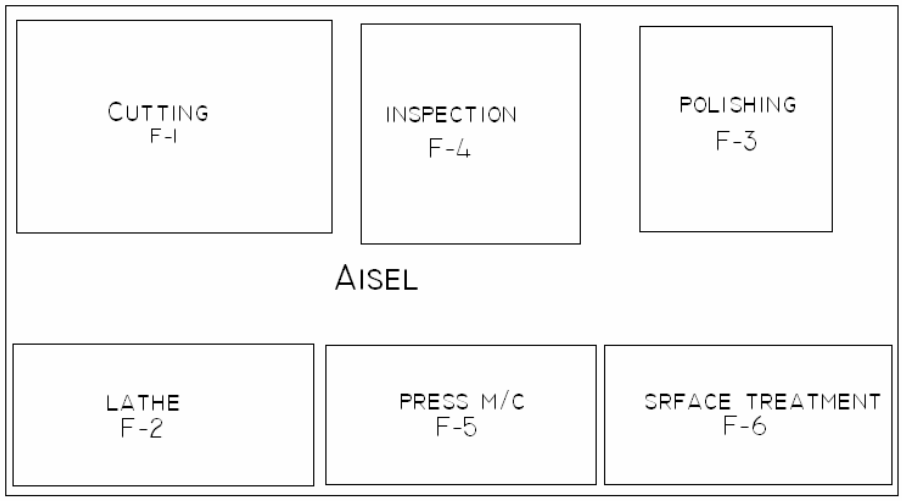


Fig 5.10: Graphical representation of the result

CHAPTER 6

CASE STUDY

In this chapter the performance of the automated system (ALREP) is validated through a case taken from Akaki spare parts and hand tools manufacturing industry. Based on the result a comparison between the existing layout and the layout generated by the automated system is conducted.

6.1 Background of the Case

Akaki spare parts and hand tools factory is one of the biggest spare parts and hand tools manufacturing industry in Ethiopia. This factory can be taken as a case where a variety of products are introduced to the manufacturing system from time to time. This is creating different problems related with floor layout of facilities that I have mentioned in my introductory part. Among the manufacturing shops of this factory such problems are mostly pronounced in the biggest or machine shop of ASPF.

Industrial engineers working in this factory often used to manipulate the effect of the existing layout of the machineries to the new product mix if the floor facilities layout is decided to be rearranged. These practices were made solely using past experience and no formal approach was exploited.

The effect of the existing layout has even bigger effect when the volume of the product is more. In my frequent visits to this manufacturing industry most of the mass production of the factory is carried out by the hand tools and cutlery division of the machine shop.

Due to the expansion of remote electrification program by the Ethiopian Electric Power Corporation, Akaki spare parts and hand tools factory is highly engaged in production of large amounts of power transmission equipments. Some of these products with their respective volume per a month period of time are shown in table 6.1.

For each of these products the hand tools and cutlery division will be dedicated at a time example for one month.

Table 6.1: Power transmission equipments and their volumes

Equipment type	Volume per 1 month
Power transmission hooks	300,000pcs
33kV pin	310,0000pcs
Big collar	20,000pcs
Cross arm	30,000pcs
Smaller collar	40,000pcs
Tie strip (15kV and 33kV)	140,000pcs

Entertaining the production of one of these products with in the existing layout of hand tools and cutlery division is coming up with a very high material handling costs and is creating congestion in the manufacturing environment.

As I have seen most of the facilities or machines in this division can easily be dismantled and relocated as per the requirement of the new product mix.

So, as a case I took the manufacturing of 33kV pin in the hand tools and cutlery division having the following part drawing and process flow of the product considered with in the existing area.

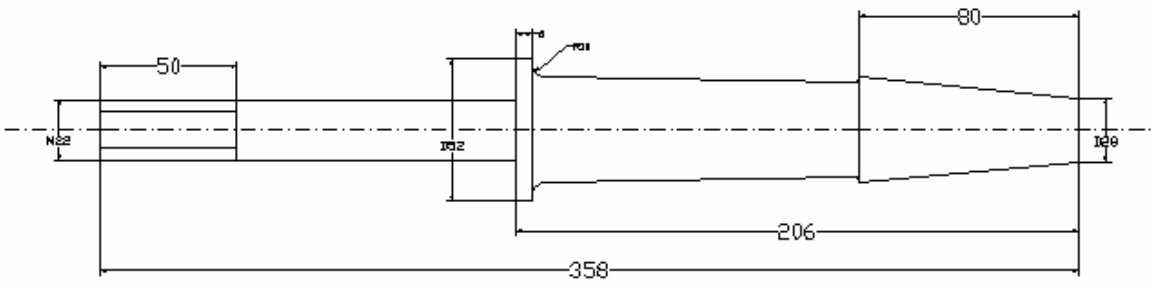


Fig 6.1: 33kV pin part drawing

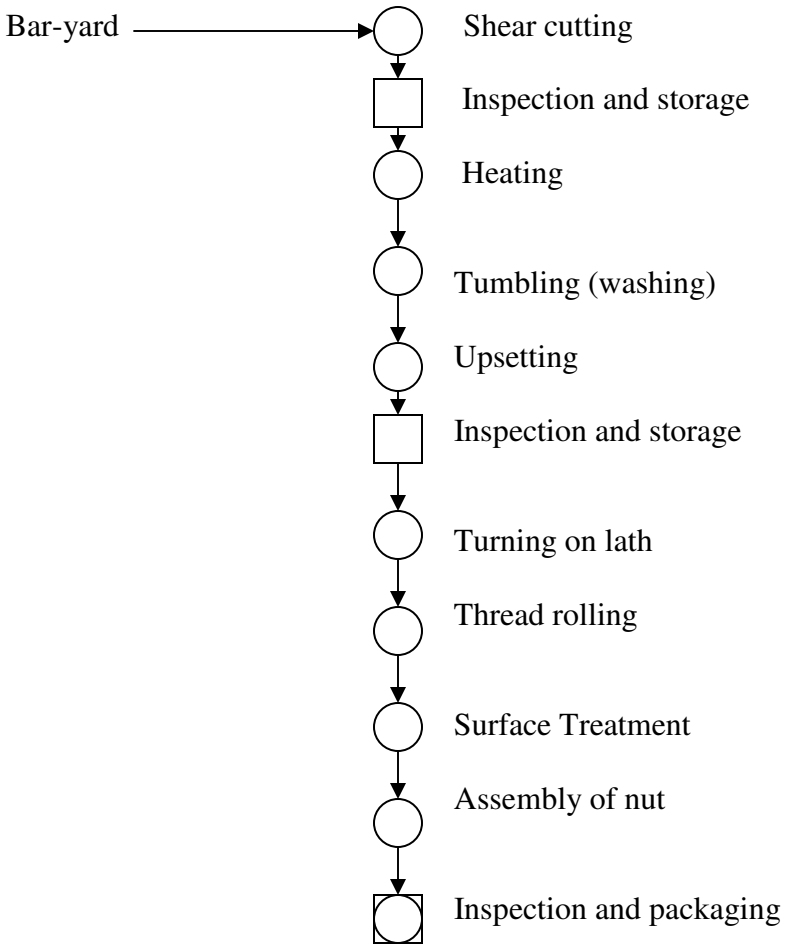


Fig 6.2: The process flow chart of 33kV pin

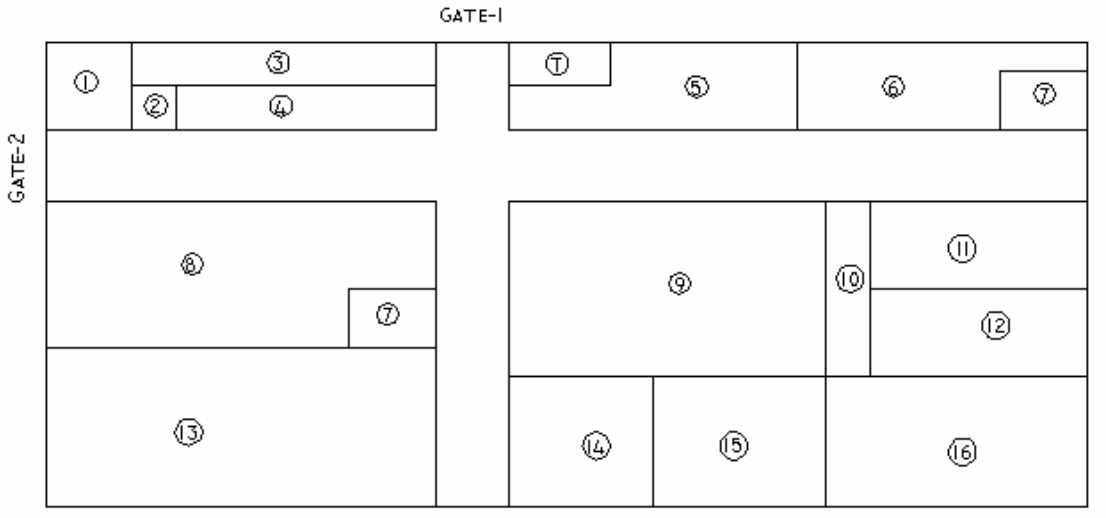


Fig 6.3: Existing layout of hand tools and cutlery division

The names of the different machines numbered above are shown in table 6.2.

Table 6.2 names of facilities in the existing hand tools and cutlery area

No	Names	No	Names
1	Hydraulic presses	9	Lathe m/cs
2	Furnace	10	Pneumatic hammers
3	Grinders	11	Plastic injection m/cs
4	Cutting m/cs	12	Free area
5	Bending m/cs	13	Free area
6	Simple presses	14	Shear cutting
7	Thread rollers	15	Free area
8	Vertical Broaching m/cs	16	Power hack saws

6.2 Case Data Inputs

During my visit to Akaki spare parts and hand tools factory, the following information's are collected to be inputs for the developed automated system.

1. The facilities required for the manufacturing of 33kV pin.
2. The actual layout of facilities in the hand tools and cutlery division including their dimensions.
3. The inter facility closeness relationships and
4. The facility relocation costs

6.2.1 Facilities Data

The facilities (places and machine sets) required to produce the 33kv pin are listed below

1. The bar yard including the gate from the yard
2. The shear cutting machine
3. Furnaces for preheating of the bars
4. Washing machines to unrest the bars
5. Press machines for upsetting the bars
6. Lathe machines for tuning

7. Thread rolling machines for thread making
8. Galvanization process for surface treatment
9. Assembly area for assembling nuts on the pin
10. The testing, inspection and packaging area

Among these facilities the researcher assumed the cutting and shearing machine, the hydraulic press machines and the galvanization process areas as non reconfigurable facilities for they are somehow huge to relocate as we want and takes time as well as money compared to the others. The rest of the above facilities assumed to be reconfigured in the available area.

6.2.2 The Geometrical Data

The geometrical data of the hand tools and cutlery division is shown in the figure 6.4. For the sake of simplicity the floor spaces occupied by non functional facilities or machines in this division are taken to be free. Spaces currently occupied by unnecessary facilities for the case product considered are taken to be obstacles which have no closeness relationship with other facilities.

As shown on the Fig 6.4 all the re-configurable facilities are removed from their prior locations (from the boundary area) and the boundary area with its fixed facilities and obstacles are considered as the geometrical data.

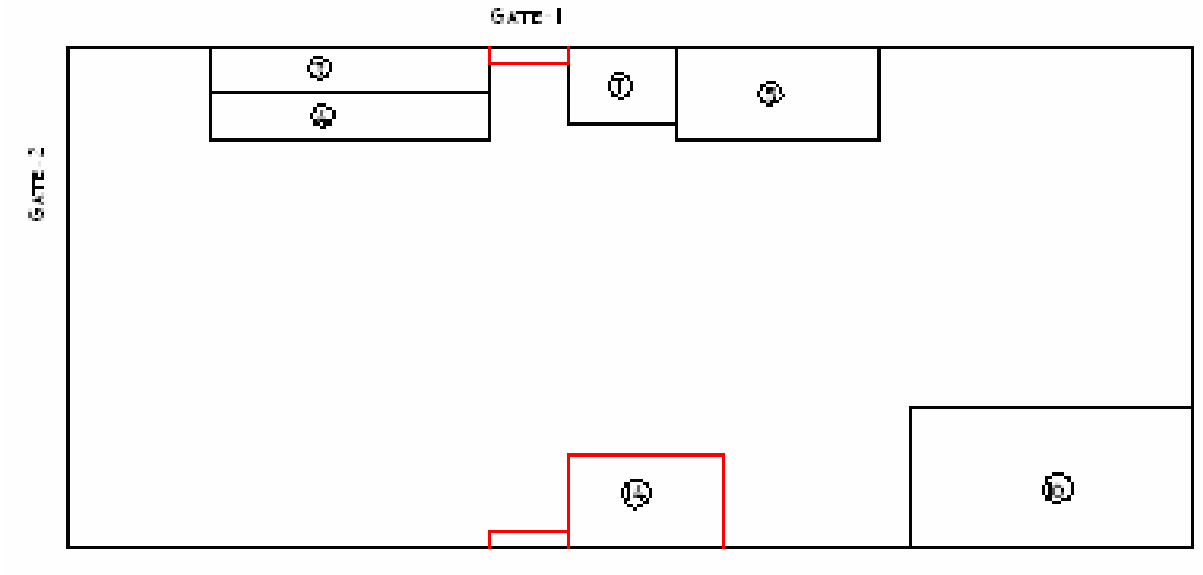


Fig 6.4: Geometrical data of the existing HC division

The dimensions of the reconfigurable as well as the non reconfigurable facilities are taken directly measuring from the actual area and it is shown in table 6.3

Table 6.3: Dimensions of the facilities considered (in meters)

- Reconfigurable facilities

No	Facility Name	Length	Width
F-1	Inspection and packaging area	16	4
F-2	Furnace	4	4
F-3	Washing Machines	12	6
F-4	Press Machines	6	6
F-5	Lath Machines	22	12
F-6	Thread Rolling	12	8
F-7	Storage and Assembly Area	22	10

- Fixed facilities

No	Facilities Name	Length	Width
F-8	Bar yard+ Gate-1	4	2
F-9	Shear cutting Machines	10	6
F-10	Surface Treatment	4	2

6.2.3 Facilities Closeness Relationships Data

The inter facility closeness relation ships of the above facilities are given a value based on the process flow shown above and it is taken as shown in table 6.4.

Table 6.4: Proximity weights of the facilities

	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10
F1										
F2	37									
F3	0	81								
F4	81	37	81							
F5	81	1	3	37						
F6	9	0	0	9	81					
F7	37	0	0	0	9	3				
F8	0	0	3	0	0	0	0			
F9	37	1	0	0	3	37	0	N/A		
F10	0	0	9	0	0	81	81	N/A	N/A	

6.2.4 Relocation Cost Data

Finding the relocation cost of the facilities is one of the challenges faced during the data collection process, because they don't have enough record of information regarding the costs associated with dismantling and relocating a certain facility. For this case the writer has taken a cost estimation made by an engineer from the production control and planning department as shown in table 6.5 for the facilities being considered.

Basically they used to pay 13 birr per hour per person during relocation practices. The time required for relocating facilities depends on the type and method of installation. For e.g. most light weight machines can simply be put on the level area by just checking the water level and area requirement for maintainability and hence they require 3 to 8hrs based on their number. Those facilities which require anchorage to the ground may take up to two days. Based on the estimated time requirement by the facilities the researcher considered, the relocation costs of all the re-configurable facilities are shown in Table 6.5.

Table 6.5: Relocation cost (in Birr) data

No	Facilities	Relocation Cost
F-1	Inspection and packaging area	100
F-2	Furnace	208
F-3	Washing Machines	832
F-4	Press Machines	832
F-5	Lath Machines	1248
F-6	Thread Rolling	832
F-7	Storage and Assembly Area	100

6.3 The Automated System (ALREP) Output

The system performed the GA based optimization of the above data. A termination condition of Δ = 10% was used along with the following genetic parameters Pmutuation = 0.05, Pcrossover = 0.7, initial pool of populations = 50 and the population size = 25.

The final result of the optimum layout including its minimum material handling cost is shown below.

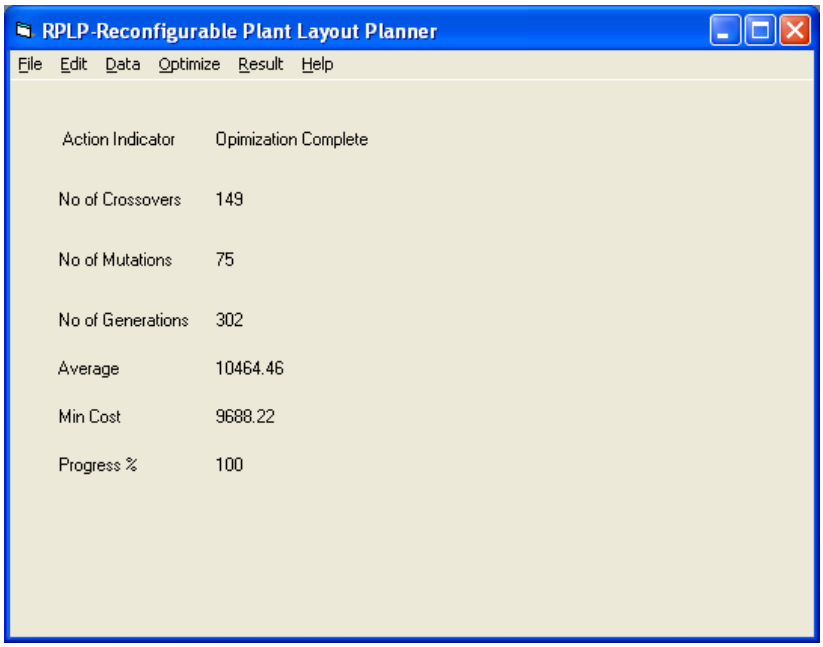


Figure 6.5: The final window after optimization process is completed

And the generated layout based on inter facility material handling cost is shown in fig 6.6.

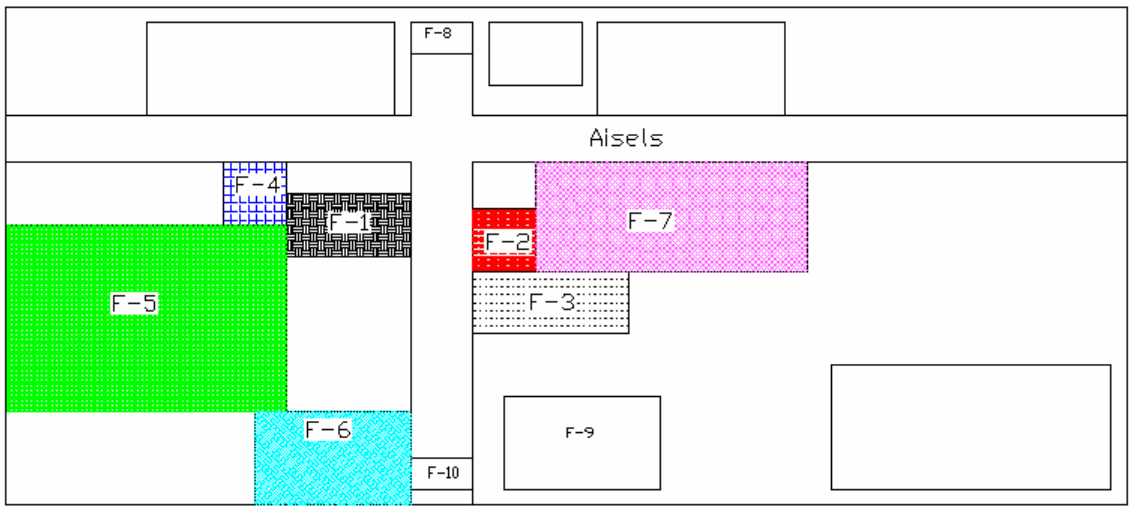


Figure 6.6: The automated system generated layout

We can see from the graph shown in the fig 6.7, the layout cost reduces as the number of generation of population increases. So it is better to have a more number of generations to get a better optimum layout.

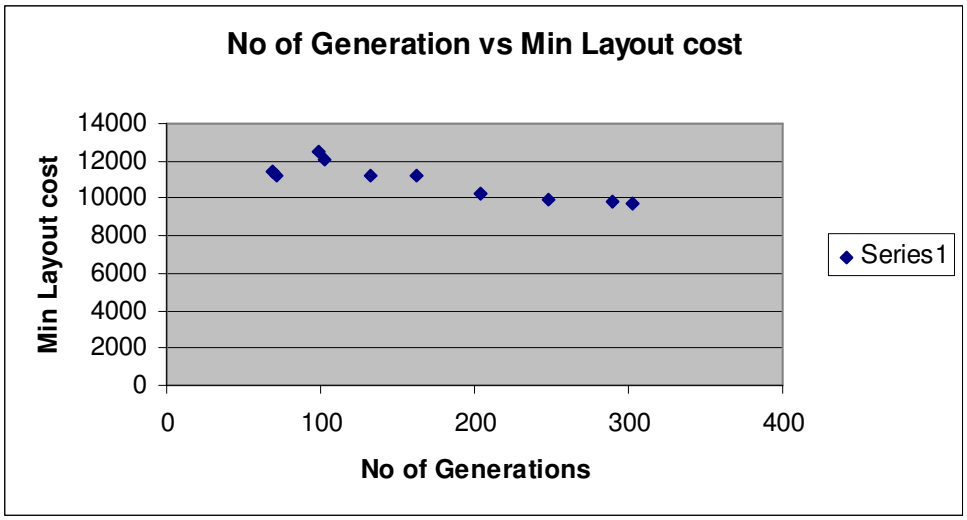


Figure 6.7: No of generation vs material handling cost

6.4 Problems with the Existing Floor Layout

The actual floor layout of the hand tools and cutlery division is shown in fig 6.3 and during my brief interview with a design engineer as well as my visit to the manufacturing system; I came up with the following problems pertaining to the existing layout.

- Due to the absence of a space area for buffer storage and assembly process, they normally used to practice assembling and storage over the aisles and this is actually creating a very high congestion in the manufacturing area and also become an obstacle for movement of forklifts through the aisles.
- Sine the washing machines are not with in the main machine shop, the movement of bars from and to these facilities is increasing the material handling cost as well as the cycle time of production tremendously.
- Inspection and packaging operations are being performed randomly where it seems to be free; there is no formal and equipped facility for these purposes. This is also creating congestion in the manufacturing area thereby a bad working environment.
- There are some machines which are not functional and very old. The presence of these machines occupied spaces which is unnecessarily creating congestion.

- The operation of thread rolling is being performed in two areas with out any valid reason. One of them is far away from the next operation area which is actually increasing the material handling cost and also the cycle time.
- Generally the working mood of most labors is not good, which is actually decreasing the productivity of the manufacturing system. During my informal interview with some of the machine operators, I recognized that they are not feeling good about the congestion created due to the existing layout.

6.5 Comparison with the Existing Layout

- One can easily see that the closeness of facilities that have got a higher closeness relationship is much better than the existing layout. For instance, a high volume flow rate occurs between lathe turning and thread rolling facilities. This is very much facilitated by the automated system generated layout than the existing one.
- When tumbling or washing to unrest the bars is required the forklifts will be very much engaged with moving materials to and from washing machine facilities which is around 55m away from hand tools and cutlery division. This is very costly as compared to the availability of these facilities with in the hand tools and cutlery.
- The distance between the thread rolling and surface treatment facilities in the existing layout is very much reduced by the automated system generated layout and hence will have a direct impact on the cost of material handing equipments.
- The availability of a proper buffer storage, assembly and inspection area will improve the congestion created in the existing layout due to the lack of these facilities and also can improves the working environment of the manufacturing system.
- I tried to estimate the cost of material handling or transport per cycle of movement of forklifts. Normally they do have three types of forklifts with a capacity of 2, 7 and 13Tones. The forklift with a 2Tone capacity is mostly used to transport 33kV pin and it costs 80 Birr per hr. when it is changed to cost per meter it become about 0.018 = 0.02Birr per meter. The approximate production cycle distance for the existing layout =330m and it costs about 6.6 Birr. For the automated system

generated layout the production cycle distance approximately reduces to 127m and it costs about 2.54Birr. From this, we can say that the automated system generated layout can reduce about (6.6 – 2.54) which is equal to 4.06 Birr/cycle.

- Akaki spare parts and hand tools factory is now providing each unit of 33kV pin by 26 Birr to EEPCO; however if they use a layout generated by the automated system depicted above, they can provide each unit with a cost of 19 Birr. These will make them very competitive in cost wise.

CHAPTER-7

CONCLUSIONS AND RECOMENDATIONS

7.1 Conclusions

- Genetic algorithms can satisfactorily be used to solve layout re-configuration problem: When tested with an actual case study, the Automated Layout Re-configuration Program yielded a highly optimum solutions with nearly 40% savings in production cycle material handling cost.

- The integration between CAD and genetic algorithms was successful in performing layout optimization process: the integrated system managed to benefit from the intricate search and optimization abilities of genetic algorithms and at the same time utilize the powerful graphical capabilities of CAD systems. Optimization results were very satisfactory as previously mentioned. Geometrical constraints dictated by the space geometry were strictly followed by the CAD platform as no overlap or out-of-boundary assignment of re-configurable facilities occurred.

- Assigning facilities with in the existing boundary of a factory should also consider the relocation costs of the facilities to be re-configured. Therefore, plant layout re-configuration problem should be based solely on material handling and relocation costs.

- It is very crucial for factories which frequently need to manipulate their facilities to use an automated tool like the one developed by this thesis. This will help them in reducing the non value adding cost in the manufacturing process which is the material transport cost.

7.2 Recommendations for Further Work

1. Provide the non rectangular bounding box identification module: the automated system is only able to identify rectangular bounding boxes of the existing factory geometry. Here I recommend a further work to be done on incorporating a module to identify factory bounding geometries with non rectangular shapes.

2. Enhance the system to be able to deal with irregular facility shapes instead of rectangular shapes: The present system is able to deal with rectangular facility shapes. Although most facilities tend to have rectangular shapes, fixed facilities may assume any irregular shape. Enhancing the system to be able to recognize irregular spaces occupied by facilities will further increase its practicality for use.

3. Enhance the system to be able to deal with actual travel distances between facilities instead of rectilinear distances: The present system utilizes rectilinear distances between facilities. These distances may not necessarily represent the realistic distances to travel inside the manufacturing area. Especially in congested environment, maneuvering around facilities may be required during travel. Using CAD features, actual travel distances around facilities can be estimated.

4. Develop a system that is capable of dealing with three dimensional spatial aspects. (3-D layout planning): The present system performs layout planning at the 2-D level. Some projects (e.g. High-rise machines) require re-configurable support facilities to be placed inside the building whilst construction. Enhancing the system to deal with the three dimensions will make it more suitable for these types of factories, further increasing its practicality for use.

5. Enhance the computer automated system to take care of different errors made by the user and also to be more users friendly. The present system

fails to handle errors in the input procedures especially in the AutoCAD environment. Further works are recommended to incorporate message boxes and other features when proper inputs are not given to the system.

Genetic Algorithm Based Reconfiguration of Plant Facilities

ABSTRACT

This paper describes a novel method based on genetic algorithms (GA) to solve the facilities re–configuration problem. Developing a proper floor layout is an important step in designing manufacturing facilities due to the impact of the layout to material handling cost and time, and its direct consequence on the overall productivity of the shop floor.

The genetic algorithm based method developed to solve this uses the objective of minimizing the movement of materials being processed in the factory.

This thesis investigates the potential of genetic algorithms in re-configuring factory floor facilities and presents a computer automated system for performing facilities re-configuration task of an existing factory floor. The system integrates the powerful graphical capabilities of AutoCAD and the intricate search and optimization abilities of genetic algorithms for the purpose of solving facilities re-configuration problem.

The computer automated system is implemented via Visual Basic 6.0. The interface features of Visual Basic and AutoCAD are utilized to capture the geometrical details of the existing factory floor layout and to represent the final solution graphically.

Introduction

There is an emerging consensus that existing layout configurations do not meet the needs of the multi-product enterprises [9, 16] and that there is a need for a new generation of factory layouts that are more flexible, modular, and more easily reconfigurable. Flexibility, modularity, and re-configurability could save factories the need to redesign their layouts each time their production requirements change. Re-layout can be highly expensive and disruptive, especially when the entire factory has to be shut down and production stopped. For factories that operate in volatile environments or produce a high variety of products, shutting down each time when demand changes or a new product is introduced, is simply not an option. In fact, plant managers may prefer to live with the inefficiencies of an existing layout rather than suffer through a costly re-layout, which in turn could become quickly obsolete.

The current choices of layouts, such as *product*, *process*, and *cellular* layouts do not adequately address the above needs because they tend to be designed for a specific product mix and production volume, both assumed to last for a sufficiently long period (e.g. 3-5 years)[9, 16]. The design criterion that routinely used in most layout design procedures is a measure of long-term material handling efficiency which fails to capture the priorities of the flexible factory. As a result, layout performance tends to deteriorate significantly with fluctuation in product volumes, mix, or routings [9, 16].

Normally Functional (Process) types of layouts are used where product variety is high and product volume is small. But Functional layout is vulnerable to changes in the product mix and/or routings resulting in a costly complete re-layout of the plant.

Generally facility layout problem can be broadly classified in to static and dynamic layout models. Static facility layout models deal with the initial design of a layout for a single period, whereas dynamic facility layout models are used for designing variable layouts depending on the product mix variation provided that

the product mixes and volumes are known prior to the commencement of the design activity as project phases [2].

As we see from the two models of floor layout design mentioned above, it is clearly seen that both models fail to meet the needs of variable floor layout design in the absence of prior information regarding the product mix and volume. This paper is basically going to fill the gap between static and dynamic facility layout models.

The task of floor facility layout re-configuration is concerned with the location and arrangements of process departments, cells or machines in the existing factory floor so that the existing floor layout will be optimum for the new product mix at hand.

This approach is based on the idea that considers the case where resources can be easily moved around so that frequent relocation of facilities is feasible. This is motivated by the fact that in many industries (e.g. consumer electronics, home appliances, garment manufacturing, etc) fabrication and assembly workstations are light and can be easily relocated. [16] In fact, even in the metal cutting industry recent advances in materials and processing technology are making it easier for manufacturing facilities to be configured and reconfigured on a more frequent basis. For example, many discrete manufactured components are made of composite materials that are light in weight and have much better mechanical properties. Newer processing technologies such as electron beam hardenings, molecular nano-technology and laser cutting is resulting in lighter weight machining equipment. Permanent magnetic chucks that carry their own energy source, that do not obstruct machining and do not magnetize the cutting tool are also being developed. With these developments in materials and processing technology, we are moving towards processing technologies which employ light weight machine tools and can process light weight parts. It is possible to envision facilities where these light weight equipment are mounted on wheels and are easily moved along suitably designed tracks embedded in the shop-floor. [16] As a result, it may not be too far fetched to say that the layout will be changed several times a year. In fact, through a workshop survey, the committee on Visionary Manufacturing Challenges for 2020 has identified *adaptable processes and equipment* and *reconfiguration of manufacturing operations* as two key enabling technologies that will help companies overcome two of the six grand challenges or fundamental goals to remain productive and profitable in the year 2020.[16] These grand challenges are to "achieve concurrency in all operations" and to "reconfigure manufacturing enterprises rapidly in response to changing needs and opportunities".[16]

When frequent re-layout is feasible, the layout design problem can be significantly simplified even when product demand and product mix are highly variable. It becomes possible to focus only on the immediate product mix and the immediate production volumes. However, since we would typically incur:

(1) Some loss in production capacity during the relocation process, and

(2) A relocation cost associated with the physical movement of resources (e.g., labor cost, dismantling and reconstruction costs, rewiring costs, and startup/setup costs), we must account for these costs when deciding whether it is beneficial to remove a resource or leave in its current location [2, 16].

The objective function of the model consists of two terms: a material handling cost term and a relocation cost term. The magnitude of the relocation costs determine whether a re-layout is carried out or not [2, 16]. In the

extreme, where re-layout costs are insignificant, an entirely new layout can be generated during each period. On the other hand, if re-layout costs are prohibitive, the existing layout would be retained. In practice, the two extreme scenarios would be unlikely to occur. Instead it would be desirable to relocate some of the resources during each period. The layout would then evolve gradually over time as flow patterns evolve. The cost of re-layout could be reduced if investments in infrastructure that facilitates re-layout are made during the initial design of the factory. For example, the facility may be designed so that it has embedded tracks that help decrease the cost of moving equipment. It may also be possible to design all interface devices for control systems so that they are interchangeable and open. In such a case, "plug and play" features may be implemented at the workstation level. Support services such as compressed gas, water or coolant lines, and waste disposal may have to be suitably designed for the concept.

A primary advantage of reconfiguring a layout when warranted by changes in product mix and volume is that material handling cost can be minimized because equipment can be reconfigured to suit the new production mix and volume. Of course, this cost must more than the cost of moving equipment from its current location to a new one. In addition, due to the short term life of a given layout and production data availability for this time period, it is possible to consider optimizing operational performance measures such as minimizing part cycle times, work in process inventory, or throughput.

The potential to frequently alter layouts, therefore, transforms the modern layout problem from a strategic problem in which only long term material handling costs are considered to a tactical problem in which operational performance measures such as reduction of product flow times, work in process inventories, and maximizing throughput rate are considered in addition to material handling and machine relocation costs when changing from one layout configuration to the next.

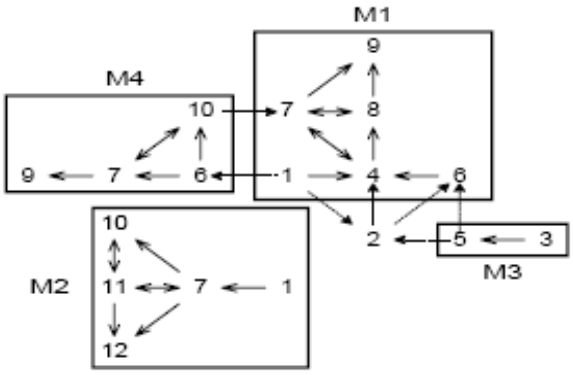


Figure 2.4: Reconfigurable Facility

Reconfiguration of floor facilities in a factory of dynamic environment also works for existing factory layouts designed with a static layout model [2; 16], so this method can best handle the existing factory floor layout problems that we have in our country.

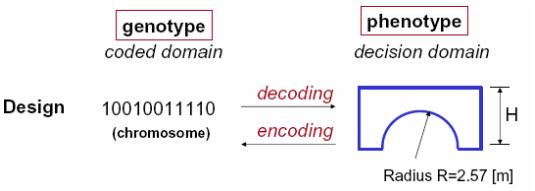
Re-configurable layout problems can best be approached by one of the search techniques called genetic Algorithms.

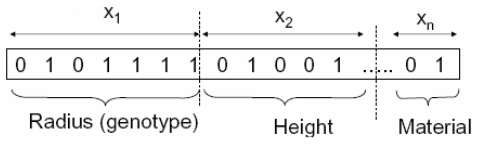
Genetic Algorithms are a search and optimization algorithm inspired by the idea of natural selection, genetics and evolution. It basically works after representing (coding) design variables by a set meaningful string. A population of strings will compete for a certain fitness value according to the objective function. Generation

of better population will be achieved by genetic operators. The generation and search capabilities of genetic algorithms make them very suitable for layout problems where a lot of candidate layouts and evaluation of them for an objective function is there. We can see the comparison between the genetic algorithm and the natural selection theory as follows:

The Coding and Decoding Schemes in Genetic Algorithms

The essential characteristic of genetic algorithms is the coding of the variables that describe the problem. The common coding method is to transform the variables to specific length binary strings [11]. For a problem depending on more than one variable the coding involves linking with each variable code. The following figures show the coding and decoding schemes in GA.



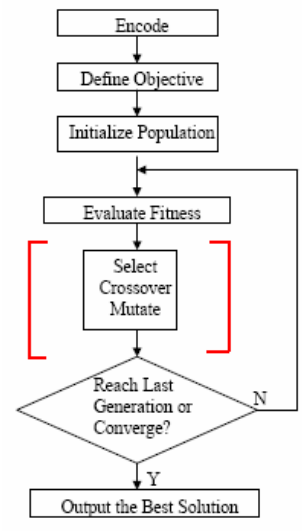


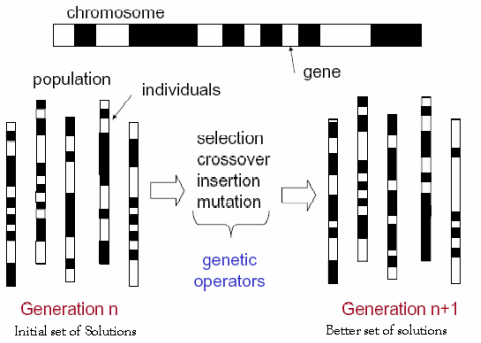
On [11], Goldberg presented two main principles for choosing GA encoding:

1) Principal of meaningful building blocks: "the user should select a coding so that short, low order schemata are relevant to the underlying problem and relatively unrelated to schemata over other fixed positions"
2) Principal of meaningful alphabets: "The user should select the smallest alphabet that permits a natural expression of the problem"

Generation of Population Using GA

According to Goldberg [12], when GA is allowed to generate several generations, there are certain steps or procedures that should be followed as shown in figure below. Three of the above procedures make actions on the existing chromosomes (individuals) to generate a new off springs (children) and these procedures are called GA operators.





Convergence Conditions in GA

In any GA the generation process must proceed until a certain termination condition is reached. Researchers have used various convergence conditions, some of these convergence conditions are:

1- Until no further improvement in the population occurs [2, 11]

2- Until all offspring in the population are replaced [11]

3- Until there is very little variation within the population itself. [11]

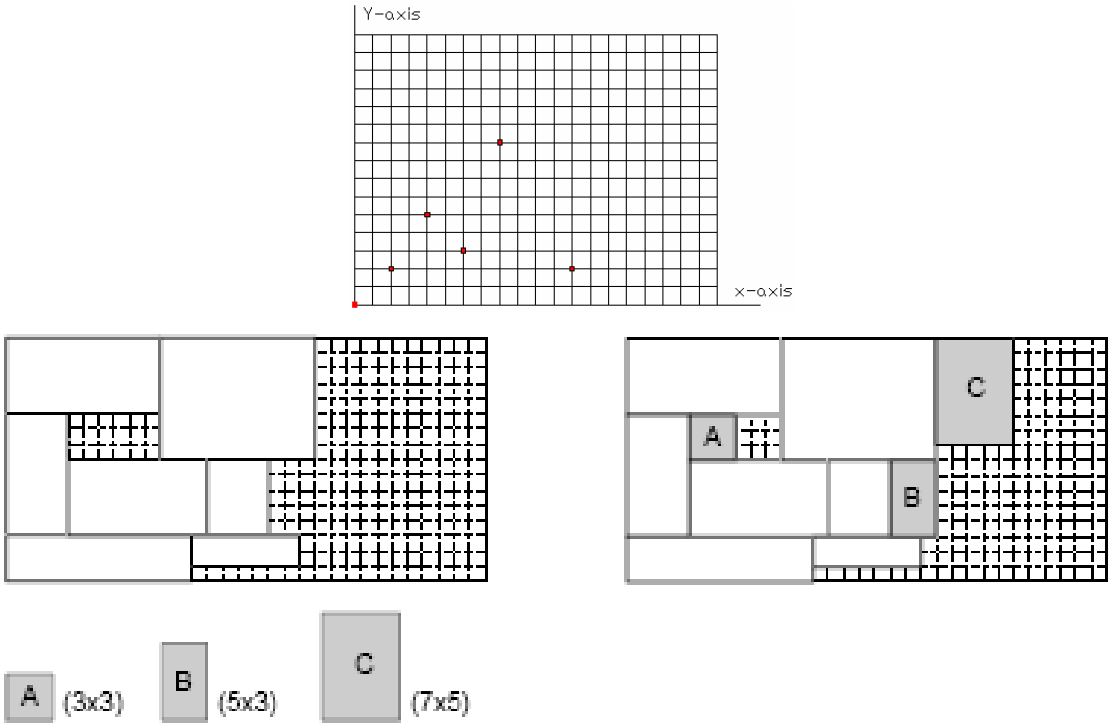
The layout reconfiguration problem is one of the domains of the broader plant layout problem and hence the systematic approach of any plant layout problem can hold for it. Normally when using the SLP approach, we normally come to a stage where a lot of alternative layout should be considered to select the optimum. I.e. the Search and Evaluate phases of SLP. So, we need to have a mechanism to generate much number of alternative solutions so that we will have higher probability of having an optimum solution.

Suitability of CAD and GA for Layout Re-configuration Problem

The layout re-configuration problem is evidently graphical in nature. Existing factory boundaries, aisles, and re-configurable facilities all occupy space and have distinct shape. Thus, the need to represent the relationship between all the aforementioned entities in some sort of graphical format is apparent**.**

Due to the powerful capabilities of generating different alternatives solutions and selecting the fittest out of them by using the natural selection mechanism, Genetic Algorithms are able to fill the gap in the search and evaluate stages of SLP.

The layout re-configuration problem can be seen as objects placement problem in which the algorithm generates sequences of objects and use some localization algorithm according to which the objects are properly placed on the grid.



The function of the algorithm can be explained as, the existing boundary of the existing factory first will be classified as a set of grid squares and the lower left corners of these squares will be coded by their location coordinate so that the reconfigurable facilities can be put on these coordinates.

In the lower picture depicted above, the first drawing depicts the initial situation on the grid. We take all the spaces occupied by facilities to be re-configured as a free space as shown above by grid areas, where the area shown as blank is an area for fixed facilities. The second picture shows simultaneously the final states after inserting objects in their legal positions. From the example we can see the situation when it is possible to place the object A at position which is surrounded by other objects. Object B did not fit into that place, so the algorithm rejects this placement coordinate for object B and randomizes the grid coordinates to find another placement point for B. This way all the reconfigurable facilities at hand shall be put and finally we will have one candidate layout.

In many optimization methods the GA move gingerly from a single point in the decision space to the next using some transition rule to determine the next point. This point to point method is dangerous because it is a perfect prescription for locating false peaks in multimodal search spaces. [1, 5] By contrast, when the GA is made to work from a rich database of points simultaneously, creates many peaks in parallel. GA's do not utilize gradient information. Thus, they are highly applicable to problems having non differentiable functions, as well as functions with multiple local optima. On the other hand, if there is a specialized optimization method for a specific problem, then genetic algorithm may not be the best optimization tool for that application [1, 10].

The Objective Function:

The objective function mostly taken in facility layout problems tends to minimize the total transportation cost i.e.

$$\textbf{Min:} \sum_{i=1}^{p-1} \sum_{j=j+1}^{p} W_{ij} * d_{ij} * C_{ij}$$

A little modification will be performed for the case of layout reconfiguration problem and it tends to minimize the transportation and relocation of facilities to be reconfigured.

$$\textbf{Min:} \sum_{i=1}^{p-1} \sum_{j=j+1}^{p} W_{ij} * d_{ij} * C_{ij} \text{ +Relocation cost}$$

This is because when facilities are dismantled and relocated to other positions there are costs related to these activities so the layout reconfiguration problem should consider these.

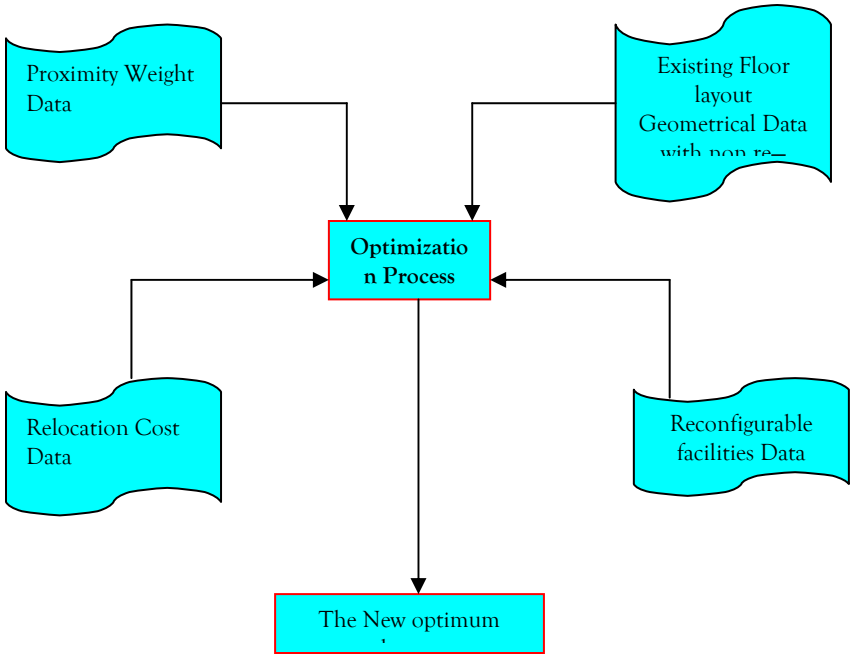
System Development

The reconfigurable layout planning system is comprised of three main components:

- An input facility that incorporates various types of data needed for the layout planning task.
- An optimization engine based on the concepts of genetic algorithms.
- An output facility that utilizes the interface features of the VB and AutoCAD application.

The automated system utilizes some amounts of data. Data used in the system can be grouped into four major categories (Table 4-1), namely temporary facility data, existing floor geometrical data and facility cost data.

Data	Description
Reconfigurable facility data	Reconfigurable facility requirements and the expected sizes of these facilities
Existing factory floor geometrical data	AutoCAD drawings representing the layout of fixed facilities.
Facility cost data	Inter-facility material handling costs along with the expected cost for relocating reconfigurable facilities.



Identification of Enclosed Factory Floor Space:

In order to perform this task the user must identify the factory floor boundaries (must be an AutoCAD lightweight poly-line) and any point lying inside the factory boundaries. Performing this task proceeds as follows:

- Using the rectilinear coordinates of the shop boundary's vertices, the VB module identifies the edges of the shop boundary as a set of equations:

  $Y = A_1$, $Y = A_2$, $X = B_1$, $X = B_2$
- Using a point inside the boundary, the VB module identifies the edges of the shop boundary as a set of inequalities:

  $Y >= A_1$, $Y <= A_2$, $X >= B_1$, $X <= B_2$
- By toggling through coordinates of all points inside the shop boundary's "Bounding Box" (Figure 4-3), the VB module chooses only those points that satisfy all linear inequalities simultaneously.

- Feasible grid squares are stored in 2 arrays: Available X ( ) and Available Y ( ) such that any point "j" has coordinates (Available X(j), Available Y(j))

Identification of Non-reconfigurable and Obstacles

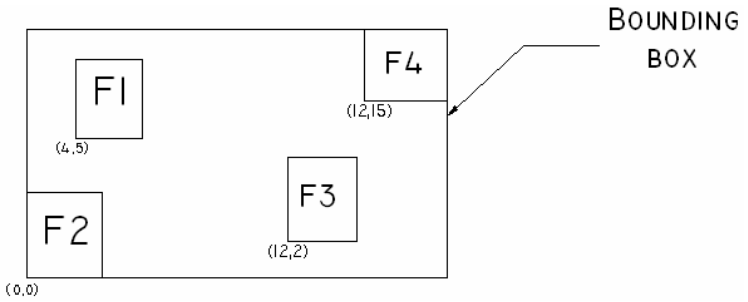
Till this stage no account has been made for obstacles or fixed facilities. Obstacles are defined as facilities with fixed positions having no closeness relationship with other facilities. Fixed facilities on the other hand have fixed positions but maintain closeness relationship with other facilities. In order to perform this task the user must identify all obstacles and fixed facilities present in the boundary. Performing this task proceeds as follows:

- Grid spaces occupied by the obstacles are removed from the arrays Available X and Available Y.
- Grid spaces occupied by the fixed facilities are removed from the arrays Available X and Available Y. The coordinates of their centroid are stored for future use in the optimization procedure.

After these two sub-tasks are accomplished, available space, obstacles and fixed facilities are translated into a set of X, Y coordinates. These coordinates will be utilized in the optimization procedure during the layout of the re-configurable facilities.

GA String Coding in the System Development

The coding of a certain solution representing the assignment of four re-configurable facilities is illustrated in Figure 4-4.



	F1	F2	F3	F4
X	4	0	12	15
Y	5	0	2	15

Constraint Satisfaction

Geometrical constraints are vital in the layout process. It is of utmost importance that re-configurable facilities be placed (1) Inside the boundaries and (2) In such a manner that no overlap occurs between any two re-configurable facilities or between re-configurable and fixed facilities.

To achieve the satisfaction of geometrical constraints, two main modules are utilized, namely "*Check-Space*" and "*Check-Overlap*". These modules are briefly explained by the figures below.

Check Space Module:

This module is a built-in function that makes sure that any re-configurable facility

(1) Lies inside the boundaries and

(2) Does not overlap with any fixed facility or space obstacles.

This function requires as input 4 variables; [Xmin, Ymin, FacilityX and FacilityY]. It provides a Boolean type true/false output.

In its operation it toggles through all grid coordinate occupied by the facility and compares them with the arrays AvailableX and AvailableY.

If any (X, Y) of facility ⊄ Available(X, Y) then Check-Space =
False
Else Check-Space = True

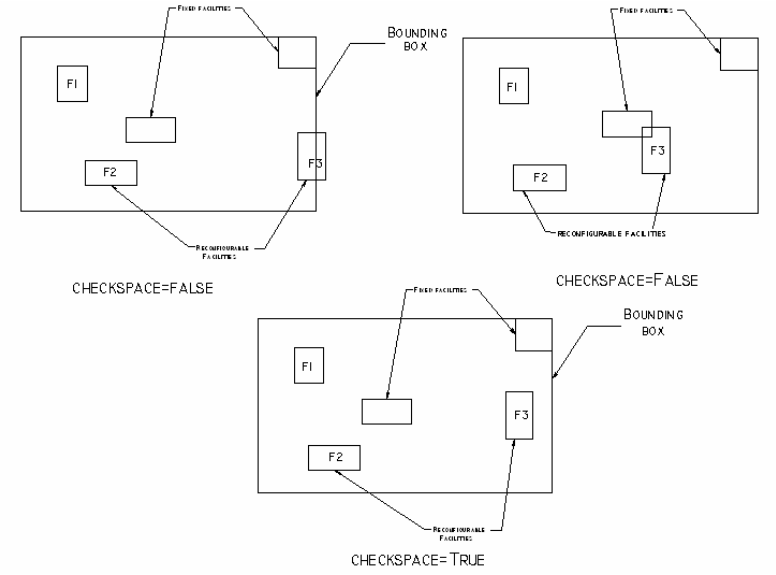


Figure 4-5: Functionality of the *Check space* module

Check Overlap Module

This module is a built-in function that performs two sequential tasks.

a) Makes sure that the facility being checked does not occupy space already being reserved for another facility that has been assigned on site.

   If any (X, Y) of facility C Available(X, Y) then Check-Overlap =
   False
   Else Check-over-lap = True

b) If no overlap occurs space is reserved for the facility

   For all (X, Y) of facility, Occupied(X, Y) = Facility(X, Y)

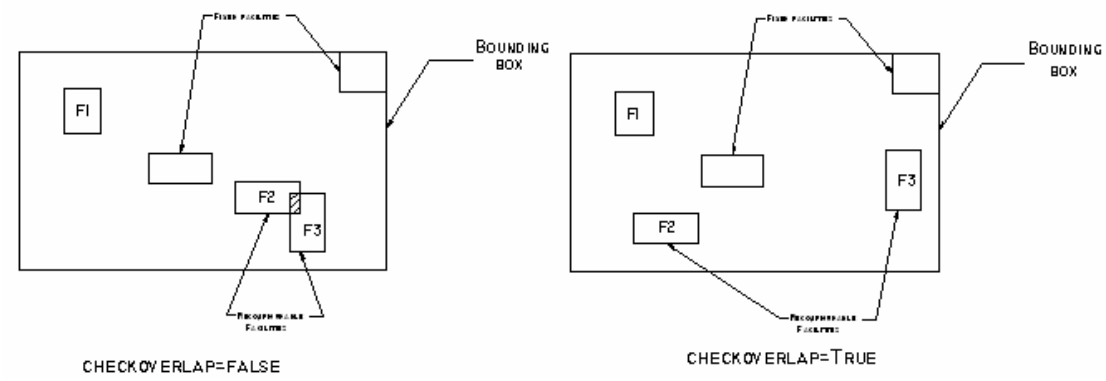
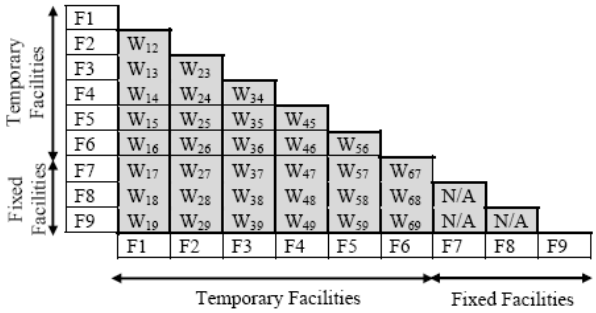


Figure 4.6: Functionality of the check overlap module

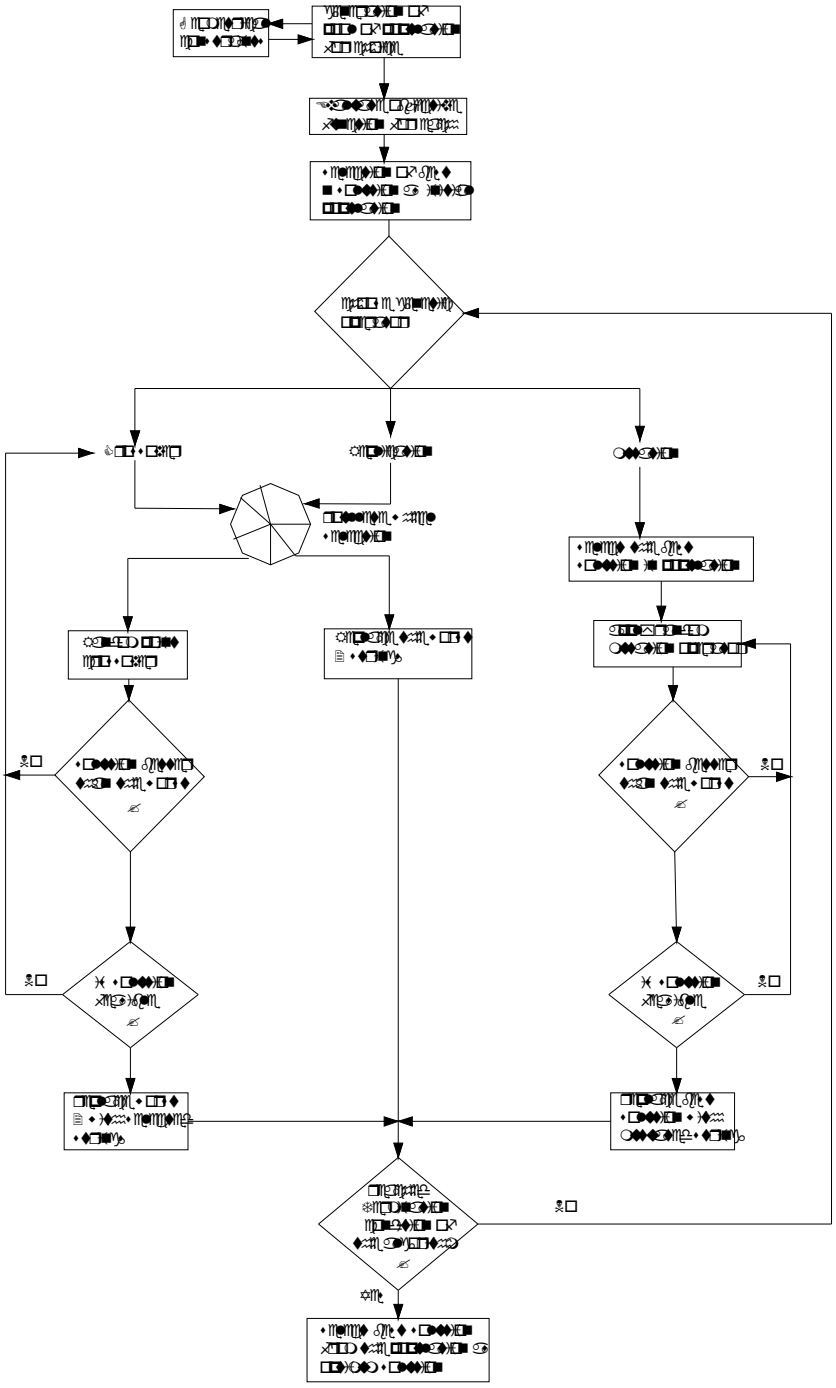
Inter-Facility Cost Matrix

The inter-facility material handling cost matrix contains the most important optimization relevant data. Using these numbers, the optimization procedure will assign facilities close to or far from one another. The proximity weight or cost data is represented in a lower triangular matrix as shown in Table 4-3 for six reconfigurable / temporary facilities and three fixed facilities.



Optimization Procedure

The details of the GA flowchart is explained hereafter



Optimum Solution Representation

The optimum solution must be represented in a user-comprehendible form. The final step in our system is the physical representation of the optimum solution.

The system, via the *draw facility* module automatically opens AutoCAD application and draws the reconfigurable facilities in their optimum positions as depicted by the genetic algorithm. Having the output in graphical form allows the designer to easily visualize the relationship between the fixed structures and reconfigurable facilities with in the factory floor space. Graphical solution representation proceeds as follows

Open CAD file representing the floor layout

For all reconfigurable facilities

Draw temporary facility in optimum position

Next temporary facility

Close CAD file

Case study

Akaki Spare Parts and hand tools Factory is one of the biggest spare parts and hand tools manufacturing industry in Ethiopia located some 25Km south of the capital Addis Ababa. The hand tools and cutlery division of this factory mostly suffer from re-configuration of its facilities as a new product is coming in to it. Maintenance crew and some industrial engineers working there try to rearrange the existing layout based on past experience.

So, this factory can be taken as a case where a variety of products are introduced to the manufacturing system from time to time. As an example the researcher took a product being manufactured for Ethiopian Electric Power Corporation which is called 33kV pin.

Case Data Inputs

During my visit to Akaki spare parts and hand tools factory, the following information's are collected to be inputs for the developed automated system.

1. The facilities required for the manufacturing of 33kV pin.
2. The actual layout of facilities in the hand tools and cutlery division including their dimensions.
3. The inter facility closeness relationships and
4. The facility relocation costs

Facilities Data

The facilities (places and machine sets) required to produce the 33kv pin are listed below

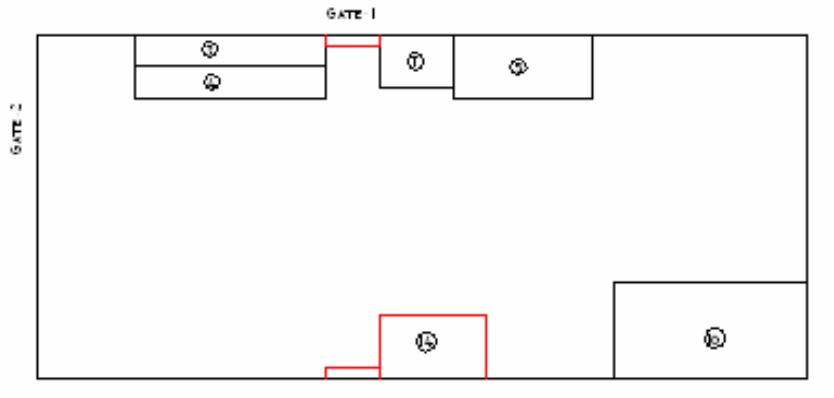
1. The bar yard including the gate from the yard
2. The shear cutting machine
3. Furnaces for preheating of the bars
4. Washing machines to unrest the bars
5. Press machines for upsetting the bars

6. Lathe machines for tuning
7. Thread rolling machines for thread making
8. Galvanization process for surface treatment
9. Assembly area for assembling nuts on the pin
10. The testing, inspection and packaging area

Among these facilities considred it is assumed that the cutting and shearing machine, the hydraulic press machines and the galvanization process areas as non reconfigurable facilities for they are somehow huge to relocate as we want and takes time as well as money compared to the others. The rest of the above facilities assumed to be reconfigured in the available area.

The Geometrical Data

The geometrical data of the hand tools and cutlery division is shown below. Spaces currently occupied by unnecessary facilities for the case product considered are taken to be obstacles which have no closeness relationship with other facilities.



The dimensions of the reconfigurable as well as the non reconfigurable facilities are taken directly measuring from the actual area and it is shown in table 6.3

- Reconfigurable facilities

No	Facility Name	Length	Width
F-1	Inspection and packaging area	16	4
F-2	Furnace	4	4
F-3	Washing Machines	12	6
F-4	Press Machines	6	6
F-5	Lath Machines	22	12
F-6	Thread Rolling	12	8
F-7	Storage and Assembly Area	22	10

- Fixed facilities

No	Facilities Name	Length	Width
F-8	Bar yard+ Gate-1	4	2
F-9	Shear cutting Machines	10	6
F-10	Surface Treatment	4	2

Facilities Closeness Relationships Data

The inter facility closeness relation ships of the above facilities are given a value based on the process flow shown above and it is taken as shown in table 6.4.

Table 6.4: Proximity weights of the facilities

	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10
F1										
F2	37									
F3	0	81								
F4	81	37	81							
F5	81	1	3	37						
F6	9	0	0	9	81					
F7	37	0	0	0	9	3				
F8	0	0	3	0	0	0	0			
F9	37	1	0	0	3	37	0	N/A		
F10	0	0	9	0	0	81	81	N/A	N/A	

Relocation Cost Data

The time required for relocating facilities depends on the type and method of installation. For e.g. most light weight machines can simply be put on the level area by just checking the water level and area requirement for maintainability and hence they require 3 to 8hrs based on their number. Those facilities which require anchorage to the ground may take up to two days. Based on the estimated time requirement by the facilities considered, the relocation costs of all the re-configurable facilities are shown in Table 6.5.

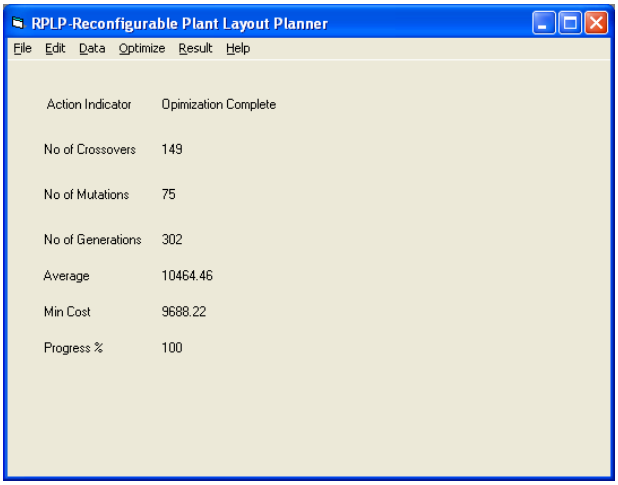
Table 6.5: Relocation cost (in Birr) data

No	Facilities	Relocation Cost
F-1	Inspection and packaging area	100
F-2	Furnace	208
F-3	Washing Machines	832
F-4	Press Machines	832
F-5	Lath Machines	1248
F-6	Thread Rolling	832
F-7	Storage and Assembly Area	100

The Automated System (ALREP) Output

The system performed the GA based optimization of the above data. A termination condition of Δ = 10% was used along with the following genetic parameters Pmutuation = 0.05, Pcrossover = 0.7, initial pool of populations = 50 and the population size = 25.

The final result of the optimum layout including its minimum material handling cost is shown below.



The final window after optimization process is completed

And the generated layout based on inter facility material handling cost is shown in fig 6.6.

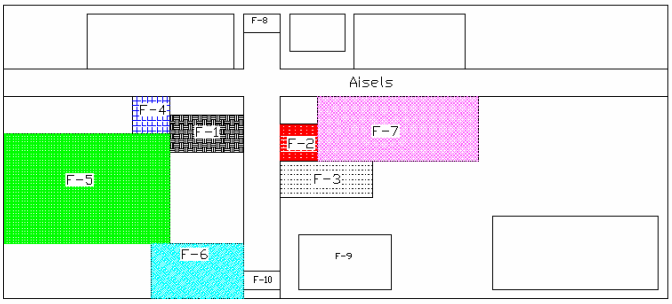
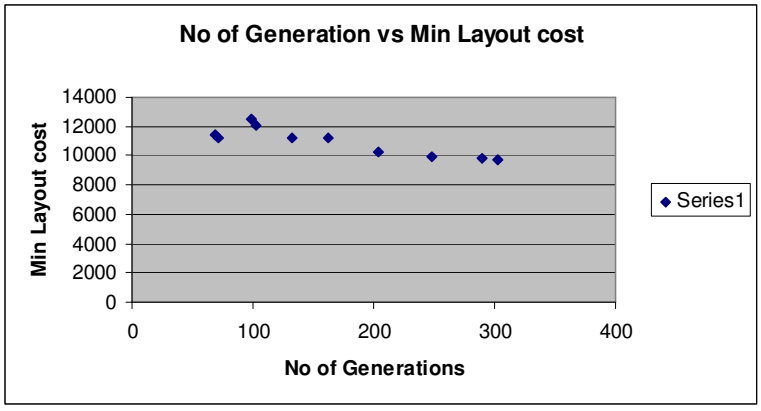


Figure 6.6: The automated system generated layout

We can see from the graph shown in the fig 6.7, the layout cost reduces as the number of generation of population increases. So it is better to have a more number of generations to get a better optimum layout.



No of generation vs material handling cost

Conclusions

- Genetic algorithms can satisfactorily be used to solve layout re-configuration problem: When tested with an actual case study, the Automated Layout Re-configuration Program yielded a highly optimum solutions with nearly 40% savings in production cycle material handling cost.
- The integration between CAD and genetic algorithms was successful in performing layout optimization process: the integrated system managed to benefit from the intricate search and optimization abilities of genetic algorithms and at the same time utilize the powerful graphical capabilities of CAD systems. Optimization results were very satisfactory as previously mentioned. Geometrical constraints dictated by the space geometry were strictly followed by the CAD platform as no overlap or out-of-boundary assignment of re-configurable facilities occurred.
- Assigning facilities with in the existing boundary of a factory should also consider the relocation costs of the facilities to be re-configured. Therefore, plant layout re-configuration problem should be based solely on material handling and relocation costs.
- It is very crucial for factories which frequently need to manipulate their facilities to use an automated tool like the one developed by this thesis. This will help them in reducing the non value adding cost in the manufacturing process which is the material transport cost.

Recommendations for Further Work

1. Provide the non rectangular bounding box identification module: the automated system is only able to identify rectangular bounding boxes of the existing factory geometry. Here I recommend a further work to be done on incorporating a module to identify factory bounding geometries with non rectangular shapes.
2. Enhance the system to be able to deal with irregular facility shapes instead of rectangular shapes: The present system is able to deal with rectangular facility shapes. Although most facilities tend to have rectangular shapes, fixed facilities may assume any irregular shape. Enhancing the system to be able to recognize irregular spaces occupied by facilities will further increase its practicality for use.
3. Enhance the system to be able to deal with actual travel distances between facilities instead of rectilinear distances: The present system utilizes rectilinear distances between facilities. These distances may not necessarily represent the realistic distances to travel inside the manufacturing area. Especially in congested environment, maneuvering around facilities may be required during travel. Using CAD features, actual travel distances around facilities can be estimated.
4. Enhance the computer automated system to take care of different errors made by the user and also to be more users friendly.

BIBLIOGRAPHY

1. A. Rangel-Merino, J. L. López-Bonilla, R. Linares y Miranda. **Optimization Method Based On Genetic Algorithms,** Col.Lindavista, C.P.07738, Mexico, D.F.Vol 12 No. 4, October 2004.
2. Christian Hicks, **A Genetic Algorithm Tool For Optimizing Cellular Or Functional Layouts in the Capital Goods Industry**,Newcastel University, Newcastel,2002.
3. Daniel Kitaw**, Industrial Management and Engineering Economy,** Addis Ababa, September 1994.
4. Dave Sly, **Object Oriented Factory Layout in AutoCAD**, Engineering Automation, Inc, Ames IA, U.S.A, 2002.
5. Jiri Kubalik, Jiri Lazanisky,Peter Ziki, **Layout Problem Optimization Using Genetic Algorithms,** Department of Cybernetics, CTU Prague, Czech Republic,2002.
6. Julia Case Bradley, Anita C. Millspaugh, **Programming in Visual Basic 5.0**, Irwin McGraw Hill, 1998.
7. Kazushi Ohashi and Kang G.Shin, **Model-Based Control for Reconfigurable Manufacturing Systems**, The University of Michigan, U.S.A and Mitsubishi Electric Corp, U.S.A, 2002.
8. Krishna K. Krishnan*, S. Hossein Cheraghi and Chandan , N. Nayak ,**Dynamic From-Between Chart: A New Tool For Solving Dynamic Facility Layout Problems,** InderScince Enterprise Ltd,2006.
9. Maher Lahmar, and Saifallah, Benjaafar, **Design of Dynamic Distributed Layouts,** University of Minnesota, December 12, 2002.
10. Norhashimah Morad**, Genetic Algorithms Optimization for the Machine Layout Problem,** University of Sains Malysia, Malysia, 2002.
11. Oliver De Weck, **Multidisciplinary System Design Optimization - Heuristic Techniques**, **A Basic Introduction To Genetic Algorithm,** Lecture 11, Massachusetts Institute of Technology, U.S.A, March 2004.
12. Oliver De Weck, **Multidisciplinary System Design Optimization Genetic Algorithms (III) Tabu Search**, Lecture 11, Massachusetts Institute of Technology, U.S.A, March 2004.
13. Parames Chutima, **Genetic Algorithm for Facility Layout Design with Unequal Departmental Areas and Different Geometric Shape Constraints**, hulalongkom University, Thailand, Thammasat Int.J.Sc.Tech.,Vo6,No2,May 2001.
14. Patrice Godefroid, Sarfraz Khurshid, **Exploring Very Large State Spaces Using Genetic Algorithms,** Massachusetts Institute of Technology, U.S.A, 2002.
15. Raymond R. Hill, **A Monte Carlo Study Of Genetic Algorithm Initial Population Generation Methods,** Department of Operational Sciences, Air Force Institute of Technology, Wright-Patterson AFB,1999.
16. Saifalah Benjafar,Sundersh S.Heragu,Sharukh A.Irani, **Next Generation Factory Layouts: Challenges And Recent Progress,** Troy NY, December 2000.

17. Sunderesh Heragu, Gang Meng1and Henk Zijm, Jan Kees vanmmeren**, Design and Analysis of Reconfigurable Layout Systems,** Renselear Polytechnic Institute, Troy, NY 2002.

18. Subhash Chandra[I] Abera Melesse[II], Vinay Pathak[III] Genetic Algorithm Demystifies Facility Layout Problems (FLP) Using Post Order Sequence Of Nodes In A Slicing Tree, Jimma University, 2003.

19. Tong Zhao and Chung-Li Tseng, **Flexible Facility Interior Layout: A real option Approach,** University of Missouri-Rolla, Rolla, U.S.A, 2002.

20. Tomahiro Fukaya, **HIPRAMS - Highly Productive and Re-Configurable Manufacturing Systems**, a Final Report, Intelligent Manufacturing Systems, June 2004.

21. P.P. Zouein and I.D. Tommelien, **Dynamic Layout Planning Using a Hybrid Incremental Solution Method,** Journal of construction management, November 1999.

22. Ronald W. Morrison, **Dispersion Based Population Initialization**, Mitreck Systems Inc, June 2002.

23. **Hasham Maged osman, CAD Based Dynamic Layout Planning, Cairo University, November 2002.**

APPENDICES

Appendix A

The Geometrical Data Detection Module

```
Sub MainProgram( )
BoundarySelection
GetEquations
GetPoints
GetFixedFac
GetObstacles
SendToFile
UserInput
End Sub
Private Sub FctoryBoundarySelection( )
Dim ShopPolyLine As AcadLWPolyline
Dim Pickedpt As Variant
Dim PolyLineVertices As Variant
Dim Vertix(10) As Double
Dim i As Byte 'Counter
With ActiveDocument.Utility
.GetEntity ShopPolyLine, Pickedpt, vbCr & "Select Factory boundaries (Must be polyline)"
End With
ShopPolyLine.GetBoundingBox Min, Max
Xmin = Min(0)
Ymin = Min(1)
Xmax = Max(0)
Ymax = Max(1)
PolyLineVertices = ShopPolyLine.Coordinates
For i = 0 To 2 * Max_Lines Step 2
On Error GoTo 10
  X(i / 2) = PolyLineVertices(i)
  Y(i / 2) = PolyLineVertices(i + 1)
Next i
10 Num_Eqn = i / 2
End Sub
Private Sub GetEquationoflines()
With ActiveDocument.Utility
InteriorPoint = .GetPoint(, vbCr & "select any point inside the boundary")
```

```
End With
For i = 1 To Num_Eqn
  If i = Num_Eqn Then
     B(i) = X(i - 1)
     If X(i - 1) < InteriorPoint(0) Then
        Flag(i) = 3
     Else: Flag(i) = 4
     End If
     GoTo 10
  End If
  If X(i) = X(i - 1) Then
     B(i) = X(i)
     If X(i) < InteriorPoint(0) Then
        Flag(i) = 3
     Else: Flag(i) = 4
     End If
     GoTo 10
  End If
  If Y(i) = Y(i - 1) Then
     B(i) = Y(i)
     If Y(i) < InteriorPoint(1) Then
        Flag(i) = 1
     Else: Flag(i) = 2
     End If
     GoTo 10
  End If
10 Next i
End Sub
Private Sub GetPointsinboundry()
Dim Counter As Byte
Dim j As Integer
Dim X As Integer, Y As Integer
For X = Xmin To Xmax - 1
  For Y = Ymin To Ymax - 1
    Counter = 0
     For i = 1 To Num_Eqn
       Select Case Flag(i)
```

```
        Case 3
        If X < B(i) Then GoTo 10
        Case 4
        If X > B(i) Then GoTo 10
        Case 1
        If Y < B(i) Then GoTo 10
        Case 2
        If Y > B(i) Then GoTo 10
        End Select
      Next i
      AvailableX(j) = X
      AvailableY(j) = Y
      j = j + 1
10 Next Y
Next X
NumofPoints = j + 1
End Sub
Private Sub GetFixedFacilities( )
Dim X1 As Integer, Y1 As Integer, X2 As Integer, Y2 As Integer
Dim FacilityBorder(10) As AcadLWPolyline
Dim PermenantFac(10) As AcadText
Dim i As Integer
'delete the selection set if it already exists
NumFixed = ActiveDocument.Utility.GetInteger("Enter the number of fixed facilities : ")
For i = 1 To NumFixed
With ActiveDocument.Utility
.GetEntity FacilityBorder(i), Pickedpt, vbCr & "Select Fixed Facilities (must be polyline)"
FacilityBorder(i).Color = acMagenta
FacilityBorder(i).Update
End With
Next i
For i = 1 To NumFixed
  FacilityBorder(i).Color = acWhite
  FacilityBorder(i).Update
Next i
For i = 1 To NumFixed
  FacilityBorder(i).GetBoundingBox Min, Max
```

```
  X1 = Min(0)
  Y1 = Min(1)
  X2 = Max(0)
  Y2 = Max(1)
  PCentroid(i, 1) = X1 + ((X2 - X1 + 1) * 0.5)
  PCentroid(i, 2) = Y1 + ((Y2 - Y1 + 1) * 0.5)
  Call RemoveOccupiedPlaces(X1, Y1, X2, Y2)
Next i
For i = 1 To NumFixed
ActiveDocument.Utility.GetEntity PermenantFac(i), Pickedpt, vbCr & "Select Fixed Facilities Name (Follow the same order of previous selection!)"
PFac_Name(i) = PermenantFac(i).TextString
PermenantFac(i).Color = acMagenta
Next i
For i = 1 To NumFixed
  PermenantFac(i).Color = acWhite
  PermenantFac(i).Update
Next i
End Sub
Private Sub GetObstaclesinboundry( )
Dim X1 As Integer, Y1 As Integer, X2 As Integer, Y2 As Integer
Dim FacilityBorder(10) As AcadLWPolyline
Dim PermenantFac As AcadText
Dim i As Integer
'delete the selection set if it already exists
NumObst = ActiveDocument.Utility.GetInteger("Enter the number Obstacles/Aisles in the shop : ")
For i = 1 To NumObst
With ActiveDocument.Utility
.GetEntity FacilityBorder(i), Pickedpt, vbCr & "Select the Obstacles in the shop (must be polyline)"
FacilityBorder(i).Color = acMagenta
FacilityBorder(i).Update
End With
Next i
For i = 1 To NumObst
  FacilityBorder(i).Color = acWhite
  FacilityBorder(i).Update
Next i
```

```
For i = 1 To NumObst
  FacilityBorder(i).GetBoundingBox Min, Max
  X1 = Min(0)
  Y1 = Min(1)
  X2 = Max(0)
  Y2 = Max(1)
  Call RemoveOccupiedPlaces(X1, Y1, X2, Y2)
Next i
End Sub
Private Sub RemoveOccupiedPlaces(X1 As Integer, Y1 As Integer, X2 As Integer, Y2 As Integer)
Dim Ywidth As Integer, Xwidth As Integer
Dim i As Integer
Dim j As Integer
Dim m As Integer
Ywidth = Y2 - Y1
Xwidth = X2 - X1
con = 1
For j = 1 To NumofPoints
 If AvailableX(j) = X1 And AvailableY(j) = Y1 Then
   For m = 0 To Ywidth
     AvailableX(j + m) = 0
     AvailableY(j + m) = 0
   Next m
   j = j + Ywidth
   X1 = X1 + con
 If X1 = X2 + 1 Then GoTo 10
 End If
Next j
10 End Sub
Private Sub Redimension( )
Dim i As Integer
Dim j As Integer
c = 0
For j = 1 To NumofPoints
 If AvailableX(j) = 0 Then
    For i = j To NumofPoints
      AvailableX(i) = AvailableX(i + 1)
```

```
      AvailableY(i) = AvailableY(i + 1)
    Next i
    j = j - 1
    c = c + 1
  End If
Next j
NumofPoints = NumofPoints - c
End Sub
Private Sub UserInput( )
ActiveDocument.Utility.Prompt "Access completed,Press any key to continue..."
Call CheckPoints
End Sub
Private Sub SaveToFile( )
Open "C:\cRPLP.LIN" For Output As #1
Write #1, NumofPoints - 1
For i = 1 To NumofPoints - 1
  Write #1, AvailableX(i - 1), AvailableY(i - 1)
Next i
Close #1
Open "C:\cRPLP.PER" For Output As #1
Write #1, NumFixed
For i = 1 To NumFixed
   Write #1, PFac_Name(i), PCentroid(i, 1), PCentroid(i, 2)
Next i
Close #1
End Sub
Private Sub CheckPoints()
Dim PPoints As AcadPoint
Dim P(2) As Double
P(2) = 0
Open "C:\cRPLP.LIN" For Input As #1
Input #1, NumofPoints
For i = 1 To NumofPoints
Input #1, P(0), P(1)
Set PPoints = ActiveDocument.ModelSpace.AddPoint(P)
Next i
End Sub
```

Appendix B

The Proximity Weights Input Module

```
Sub Proximityweightdata( )
  Dim i As Integer, j As Integer
  For j = 3 To 12
    For i = 3 To 12
      If ActiveSheet.Cells(i, j).Value < 0 Then
        Prox(j - 2, i - 2) = 0
      Else
        Prox(j - 2, i - 2) = ActiveSheet.Cells(i, j).Value
      End If
    Next i
  Next j
  Open "C:\cRPLP.xls" For Output As #1
  Write #1, Prox(j - 2, i - 2)
End Sub
```

Appendix C

The G.A based main optimization Module

```
Sub Optimization( )
 String
 ObjectiveFunc
 PopSort
 Initialize
 Convergence = 10
Do
Generate
 Statistics
 OutputData
Loop Until Delta < Convergence / 100
 Form1.lblProgress.Caption = 90
 SaveResults
 MinObjFunc = Minimum
 Form1.lblProgress.Caption = 100
 Form1.lblAction = "Opimization Complete"
 Form14.lblMincost.Caption = Minimum
 Form14.lblLayoutcost.Caption = Minimum + Relocationcost
```

```
End Sub
Private Sub String( )
 Dim random As Integer
 Dim j As Byte, N As Integer
 Form1.lblAction.Caption = "Initializing first population"
 ReservedPts = 1
 Row = 0
For N = 1 To InitialChoice
   For j = 1 To NumTemp
10 Randomize
   random = Int((Rnd * NumofPoints) + 1)
   PoolPop(N).TypePop.ChromosomeX(j) = AvailableX(random)
   PoolPop(N).TypePop.ChromosomeY(j) = AvailableY(random)
   'Check to make sure facility fits on space:
   If CheckSpace(PoolPop(N).TypePop.ChromosomeX(j), PoolPop(N).TypePop.ChromosomeY(j),
Fac_Length(j), Fac_Width(j)) = False Then GoTo 10
   'Check for no overlap between facilities:
   If CheckOverlap(PoolPop(N).TypePop.ChromosomeX(j), PoolPop(N).TypePop.ChromosomeY(j),
Fac_Length(j), Fac_Width(j)) = False Then GoTo 10
   Next j
   'Reserving places for placed facilities:
 EmptyOccupied
 Form1.lblProgress.Caption = (N / InitialChoice) * 100
 Form1.Refresh
Next N
End Sub
Public Function CheckSpace(Xmin As Byte, Ymin As Byte, X As Integer, Y As Integer) As Boolean
Dim X1 As Byte, Y1 As Byte, i As Integer
For X1 = Xmin To Xmin + X - 1
 For Y1 = Ymin To Ymin + Y - 1
  For i = 1 To NumofPoints
   If X1 = AvailableX(i) And Y1 = AvailableY(i) Then GoTo 5
  Next i
CheckSpace = False
GoTo 10
5 Next Y1
Next X1
```

```
 CheckSpace = True
10 End Function
Public Function CheckOverlap(Xmin As Byte, Ymin As Byte, X As Integer, Y As Integer) As Boolean
Dim X1 As Byte, Y1 As Byte, i As Integer
For X1 = Xmin To Xmin + X - 1
 For Y1 = Ymin To Ymin + Y - 1
  For i = 1 To ReservedPts
   If X1 = OccupiedX(i) And Y1 = OccupiedY(i) Then
    CheckOverlap = False
    GoTo 10
   End If
  Next i
 Next Y1
Next X1
CheckOverlap = True
'Reserve space for placed facility:
For X1 = Xmin To Xmin + X - 1
 For Y1 = Ymin To Ymin + Y - 1
  OccupiedX(ReservedPts) = X1
  OccupiedY(ReservedPts) = Y1
  ReservedPts = ReservedPts + 1
 Next Y1
Next X1
10 End Function
Private Sub EmptyOccupied( )
Dim i As Integer
For i = 1 To ReservedPts
 OccupiedX(i) = 0
 OccupiedY(i) = 0
Next i
ReservedPts = 1
End Sub
Private Sub PopSort( )
Dim Min As Single
Dim i As Integer
Dim j As Integer
Dim Flag As Integer
```

```
For i = 1 To PopSize
 Min = 100000
  For j = 1 To InitialChoice
    If PoolPop(j).TypePop.ObjectiveFunc < Min Then
       Flag = j
       Min = PoolPop(j).TypePop.ObjectiveFunc
    End If
  Next j
  For j = 1 To NumTemp
   CurrentPop(i).TypePop.ChromosomeX(j) = PoolPop(Flag).TypePop.ChromosomeX(j)
   CurrentPop(i).TypePop.ChromosomeY(j) = PoolPop(Flag).TypePop.ChromosomeY(j)
   CurrentPop(i).TypePop.ObjectiveFunc = PoolPop(Flag).TypePop.ObjectiveFunc
  Next j
  If PoolPop(Flag).TypePop.ObjectiveFunc = 0 Then
     CurrentPop(i).TypePop.Fitness = 0
  Else
     CurrentPop(i).TypePop.Fitness = 1 / PoolPop(Flag).TypePop.ObjectiveFunc
  End If
 PoolPop(Flag).TypePop.ObjectiveFunc = 100000
Next i
End Sub
Private Sub ObjectiveFunction( )
Dim i As Byte, j As Byte, k As Byte
Dim N As Integer
Dim ObjFunc As Single
Dim d(20, 20) As Single
TotalNumber = NumTemp + NumFixed
For N = 1 To InitialChoice
 'Centroid Position
  For i = 1 To NumTemp
    CG_X(i) = PoolPop(N).TypePop.ChromosomeX(i) + Fac_Length(i) / 2
    CG_Y(i) = PoolPop(N).TypePop.ChromosomeY(i) + Fac_Width(i) / 2
  Next i
  j = 1
  For i = 1 + NumTemp To TotalNumber
    CG_X(i) = PCentroid(j, 1)
    CG_Y(i) = PCentroid(j, 2)
```

```
    j = j + 1
  Next i
ObjFunc = 0
k = 2
For i = 1 To TotalNumber
  For j = k To TotalNumber
    d(i, j) = Distance(CG_X(i), CG_Y(i), CG_X(j), CG_Y(j))
    ObjFunc = ObjFunc + d(i, j) * Prox(i, j)
  Next j
  k = k + 1
Next i
PoolPop(N).TypePop.ObjectiveFunc = ObjFunc
Next N
End Sub
Public Function Distance(X1 As Single, Y1 As Single, X2 As Single, Y2 As Single) As Single
  Distance = ((X1 - X2) ^ 2 + (Y1 - Y2) ^ 2) ^ 0.5
End Function
Private Sub Sort( )
Dim i As Byte
Dim Max As Single, Min As Single
Max = CurrentPop(10).TypePop.ObjectiveFunc
Min = CurrentPop(1).TypePop.ObjectiveFunc
For i = 1 To PopSize
  If CurrentPop(i).TypePop.ObjectiveFunc < Min Then
    Min = CurrentPop(i).TypePop.ObjectiveFunc
    MinSolution = i
  End If
  If CurrentPop(i).TypePop.ObjectiveFunc > Max Then
    Maximum2 = Max
    Max = CurrentPop(i).TypePop.ObjectiveFunc
    MaxSolution2 = MaxSolution
    MaxSolution = i
  End If
Next i
Maximum = Max
Minimum = Min
End Sub
```

```
Private Sub GeneratePop( )
'This procedure generates a random offspring.
Dim i As Byte, j As Byte, k As Byte
Dim TempX As Integer, TempY As Integer
Dim jcross As Integer
Dim mate1, mate2 As Integer
Dim X As Single
CurrentSOF
Select Case Flip(Pcross, Pmutation)
Case 1
  Nmutation = Nmutation + 1
  Action = "Mutation"
  Mutation
Case 2
  Ncross = Ncross + 1
  Action = "Crossover"
  Crossover
Case 3
  Action = "Copy Parents"
  CopyParents
End Select
End Sub
Private Sub Mutation( )
Dim lop As Integer, h As Integer
Dim Best As Single
Dim m As Byte, T As Byte, d As Byte, j As Byte
m = MinSolution 'The best solution so far
Best = CurrentPop(m).TypePop.Fitness
For T = 1 To NumTemp
   For h = -1 To 1 Step 2
      For d = 0 To 1
         Select Case d
'Change in the X direction
         Case 0
On Error Resume Next
         CurrentPop(m).TypePop.ChromosomeX(T) = CurrentPop(m).TypePop.ChromosomeX(T) + h
         ObjectiveFunction (m)
```

```
          If CurrentPop(m).TypePop.Fitness < Best Then
'The new offspring are not better than the worst Population member
            CurrentPop(m).TypePop.ChromosomeX(T) = CurrentPop(m).TypePop.ChromosomeX(T) - h
            ObjectiveFunction (m)
            GoTo 5
          End If
EmptyOccupied
    For j = 1 To NumTemp
      If CheckSpace(CurrentPop(m).TypePop.ChromosomeX(j), CurrentPop(m).TypePop.ChromosomeY(j),
Fac_Length(j), Fac_Width(j)) = False Then
        CurrentPop(m).TypePop.ChromosomeX(T) = CurrentPop(m).TypePop.ChromosomeX(T) - h
        GoTo 5
      End If
      If CheckOverlap(CurrentPop(m).TypePop.ChromosomeX(j),
CurrentPop(m).TypePop.ChromosomeY(j), Fac_Length(j), Fac_Width(j)) = False Then
        CurrentPop(m).TypePop.ChromosomeX(T) = CurrentPop(m).TypePop.ChromosomeX(T) - h
        GoTo 5
      End If
    Next j
    GoTo 10
'Change in the Y direction
    Case 1
    On Error Resume Next
    CurrentPop(m).TypePop.ChromosomeY(T) = CurrentPop(m).TypePop.ChromosomeY(T) + h
    ObjectiveFunction (m)
    If CurrentPop(m).TypePop.Fitness < Best Then
      CurrentPop(m).TypePop.ChromosomeY(T) = CurrentPop(m).TypePop.ChromosomeY(T) - h
      ObjectiveFunction (m)
      GoTo 5
     End If
     EmptyOccupied
  For j = 1 To NumTemp
    If CheckSpace(CurrentPop(m).TypePop.ChromosomeX(j), CurrentPop(m).TypePop.ChromosomeY(j),
Fac_Length(j), Fac_Width(j)) = False Then
      CurrentPop(m).TypePop.ChromosomeY(T) = CurrentPop(m).TypePop.ChromosomeY(T) - h
      GoTo 5
    End If
```

```
    If CheckOverlap(CurrentPop(m).TypePop.ChromosomeX(j), CurrentPop(m).TypePop.ChromosomeY(j),
Fac_Length(j), Fac_Width(j)) = False Then
       CurrentPop(m).TypePop.ChromosomeY(T) = CurrentPop(m).TypePop.ChromosomeY(T) - h
       GoTo 5
    End If
  Next j
  GoTo 10
  End Select
5 Next d
 Next h
Next T
10 End Sub
Private Sub Crossover( )
Dim i As Byte, j As Byte, k As Byte
Dim temp1(Maxstring) As Byte, temp2(Maxstring) As Byte, temp3(Maxstring) As Byte, temp4(Maxstring) As
Byte
Dim TempX As Byte, TempY As Byte
Dim jcross As Integer
Dim mate1, mate2 As Integer
Dim Worst As Single 'Worst fitness so far
'i and k are the chromosome number to replace
i = MaxSolution
k = MaxSolution2
Worst = Maximum2
10 mate1 = SelectChrom(PopSize, Current_SOF)
mate2 = SelectChrom(PopSize, Current_SOF)
Randomize
jcross = Int(((NumTemp - 1) * Rnd) + 1)
'1st half of exchange
For j = 1 To jcross
   NewPop(1).TypePop.ChromosomeX(j) = CurrentPop(mate1).TypePop.ChromosomeX(j)
   NewPop(1).TypePop.ChromosomeY(j) = CurrentPop(mate1).TypePop.ChromosomeY(j)
   NewPop(2).TypePop.ChromosomeX(j) = CurrentPop(mate2).TypePop.ChromosomeX(j)
   NewPop(2).TypePop.ChromosomeY(j) = CurrentPop(mate2).TypePop.ChromosomeY(j)
Next j
'2nd half of exchange
For j = jcross + 1 To NumTemp
```

```
    TempX = CurrentPop(mate1).TypePop.ChromosomeX(j)
    TempY = CurrentPop(mate1).TypePop.ChromosomeY(j)
    NewPop(1).TypePop.ChromosomeX(j) = CurrentPop(mate2).TypePop.ChromosomeX(j)
    NewPop(1).TypePop.ChromosomeY(j) = CurrentPop(mate2).TypePop.ChromosomeY(j)
    NewPop(2).TypePop.ChromosomeX(j) = TempX
    NewPop(2).TypePop.ChromosomeY(j) = TempY
Next j
EmptyOccupied
For j = 1 To NumTemp
  If CheckSpace(NewPop(1).TypePop.ChromosomeX(j), NewPop(1).TypePop.ChromosomeY(j),
Fac_Length(j), Fac_Width(j)) = False Then GoTo 10
     If CheckOverlap(NewPop(1).TypePop.ChromosomeX(j), NewPop(1).TypePop.ChromosomeY(j),
Fac_Length(j), Fac_Width(j)) = False Then GoTo 10
Next j
EmptyOccupied
For j = 1 To NumTemp
  If CheckSpace(NewPop(2).TypePop.ChromosomeX(j), NewPop(2).TypePop.ChromosomeY(j),
Fac_Length(j), Fac_Width(j)) = False Then GoTo 10
     If CheckOverlap(NewPop(2).TypePop.ChromosomeX(j), NewPop(2).TypePop.ChromosomeY(j),
Fac_Length(j), Fac_Width(j)) = False Then GoTo 10
Next j
For j = 1 To NumTemp
    temp1(j) = CurrentPop(i).TypePop.ChromosomeX(j)
    temp2(j) = CurrentPop(i).TypePop.ChromosomeY(j)
    temp3(j) = CurrentPop(k).TypePop.ChromosomeX(j)
    temp4(j) = CurrentPop(k).TypePop.ChromosomeY(j)
    CurrentPop(i).TypePop.ChromosomeX(j) = NewPop(1).TypePop.ChromosomeX(j)
    CurrentPop(i).TypePop.ChromosomeY(j) = NewPop(1).TypePop.ChromosomeY(j)
    CurrentPop(k).TypePop.ChromosomeX(j) = NewPop(2).TypePop.ChromosomeX(j)
    CurrentPop(k).TypePop.ChromosomeY(j) = NewPop(2).TypePop.ChromosomeY(j)
Next j
ObjectiveFunction (i)
ObjectiveFunction (k)
If CurrentPop(i).TypePop.ObjectiveFunc > Worst Or CurrentPop(k).TypePop.ObjectiveFunc > Worst Then
'The new offspring are not better than the worst population member
For j = 1 To NumTemp
    CurrentPop(i).TypePop.ChromosomeX(j) = temp1(j)
```

```
  CurrentPop(i).TypePop.ChromosomeY(j) = temp2(j)
  CurrentPop(k).TypePop.ChromosomeX(j) = temp3(j)
  CurrentPop(k).TypePop.ChromosomeY(j) = temp4(j)
Next j
ObjectiveFunction (i)
ObjectiveFunction (k)
GoTo 10
End If
Parent1 = mate1
Parent2 = mate2
ObjectiveFunction (i)
ObjectiveFunction (k)
End Sub
Private Sub CopyParents( )
Dim i As Byte, j As Byte, k As Byte
Dim mate1, mate2 As Integer
'i and k are the chromosome number to replace
i = MaxSolution
k = MaxSolution2
mate1 = SelectChrom(PopSize, Current_SOF)
mate2 = SelectChrom(PopSize, Current_SOF)
For j = 1 To NumTemp
  CurrentPop(i).TypePop.ChromosomeX(j) = CurrentPop(mate1).TypePop.ChromosomeX(j)
  CurrentPop(i).TypePop.ChromosomeY(j) = CurrentPop(mate1).TypePop.ChromosomeY(j)
  CurrentPop(k).TypePop.ChromosomeX(j) = CurrentPop(mate2).TypePop.ChromosomeX(j)
  CurrentPop(k).TypePop.ChromosomeY(j) = CurrentPop(mate2).TypePop.ChromosomeY(j)
Next j
Parent1 = mate1
Parent2 = mate2
ObjectiveFunction (i)
ObjectiveFunction (k)
End Sub
Public Function SelectChrom(PopSize As Integer, Current_SOF As Single) As Byte
Dim rand, partsum As Single
Dim j As Byte
partsum = 0
j = 0
```

```
Randomize
rand = Rnd * Current_SOF
Do
  j = j + 1
  partsum = partsum + CurrentPop(j).TypePop.Fitness
Loop Until partsum >= rand Or j = PopSize
  SelectChrom = j
End Function
Private Sub Statistics( )
Dim i As Byte
Dim Max As Single, Min As Single, Total As Single
Max = 0
Min = 100000
Total = 0
For i = 1 To PopSize
  If CurrentPop(i).TypePop.ObjectiveFunc < Min Then
    Min = CurrentPop(i).TypePop.ObjectiveFunc
    MinSolution = i
  End If
  If CurrentPop(i).TypePop.ObjectiveFunc > Max Then
    Max = CurrentPop(i).TypePop.ObjectiveFunc
    MaxSolution = i
  End If
Total = Total + CurrentPop(i).TypePop.ObjectiveFunc
Next i
Average = Total / PopSize
Maximum = Max
Minimum = Min
Delta = (Maximum - Minimum) / Maximum
Ngener = Ngener + 1
End Sub
Private Sub OutputData( )
Dim i As Byte
Form1.lblMincost.Caption = Minimum 'Min
Form1.lblAverage.Caption = Average 'Avg
Form1.lblNcross.Caption = Ncross 'Number of crossovers
Form1.lblNmut.Caption = Nmutation 'number of mutations
```

```
Form1.lblNgen.Caption = Ngener 'number of generations
Form1.Refresh
End Sub
Public Function Flip(Pcross As Single, Pmutation As Single) As Byte
Randomize
If Rnd < Pmutation Then
  Flip = 1 'Mutation
ElseIf Rnd < Pcross Then
  Flip = 2 'Crossover
Else
  Flip = 3 'Neither (replication)
End If
End Function
Public Sub CurrentSOF( )
Dim i As Byte
Current_SOF = 0
For i = 1 To PopSize
  Current_SOF = Current_SOF + CurrentPop(i).TypePop.Fitness
Next i
End Sub
Private Sub Initialize( )
 Form1.lblAction.Caption = "Running GA"
 Nmutation = 0
 Ncross = 0
 Ngener = 0
End Sub
Private Sub SaveResults( )
Dim i As Byte
For i = 1 To NumTemp
  OptSol(i, 1) = CurrentPop(MinSolution).TypePop.ChromosomeX(i)
  OptSol(i, 2) = CurrentPop(MinSolution).TypePop.ChromosomeY(i)
Next i
End Sub
```

Appendix D

The Graphical Representation Module

```
Public Sub DispalyFacilities( )
Dim AutoCADapplication As AcadApplication
Dim PLinePoints(20) As Double
Dim TextPoint(2) As Double
Dim Point1(2) As Double, Point2(2) As Double, Point3(2) As Double, Point4(2) As Double
Dim Facility As AcadPolyline
Dim FacilityText As AcadText
Dim i As Byte, j As Byte
Set AutoCADapplication = CreateObject("AutoCAD.Application")
AutoCADapplication.Visible = True
AutoCADapplication.Documents.Open ("C:\cRPLP\Drawing macro.dwg")
Point1(2) = 0
Point2(2) = 0
Point3(2) = 0
Point4(2) = 0
For j = 1 To NumTemp
Point1(0) = OptSol(j, 1)
Point1(1) = OptSol(j, 2)
Point2(0) = Point1(0) + Fac_Length(j)
Point2(1) = Point1(1)
Point3(0) = Point2(0)
Point3(1) = Point2(1) + Fac_Width(j)
Point4(0) = Point1(0)
Point4(1) = Point3(1)
For i = 0 To 2
PLinePoints(i) = Point1(i)
PLinePoints(i + 3) = Point2(i)
PLinePoints(i + 6) = Point3(i)
PLinePoints(i + 9) = Point4(i)
PLinePoints(i + 12) = Point1(i)
Next i
TextPoint(0) = Point1(0) + 0.1
TextPoint(1) = Point1(1) + 0.1
TextPoint(2) = 0
```

```
Set Facility = AutoCADapplication.ActiveDocument.ModelSpace.AddPolyline(PLinePoints)
Set FacilityText = AutoCADapplication.ActiveDocument.ModelSpace.AddText(Fac_Name(j), TextPoint, 1)
Next j
End Sub
```

BIBLIOGRAPHY

1. A. Rangel-Merino, J. L. López-Bonilla, R. Linares y Miranda. **Optimization Method Based On Genetic Algorithms,** Col.Lindavista, C.P.07738, Mexico, D.F.Vol 12 No. 4, October 2004.

2. Christian Hicks, **A Genetic Algorithm Tool For Optimizing Cellular Or Functional Layouts in the Capital Goods Industry**,Newcastel University, Newcastel,2002.

3. Daniel Kitaw**, Industrial Management and Engineering Economy,** Addis Ababa, September 1994.

4. Dave Sly, **Object Oriented Factory Layout in AutoCAD**, Engineering Automation, Inc, Ames IA, U.S.A, 2002.

5. Jiri Kubalik, Jiri Lazanisky,Peter Ziki, **Layout Problem Optimization Using Genetic Algorithms,** Department of Cybernetics, CTU Prague, Czech Republic,2002.

6. Julia Case Bradley, Anita C. Millspaugh, **Programming in Visual Basic 5.0**, Irwin McGraw Hill, 1998.

7. Kazushi Ohashi and Kang G.Shin, **Model-Based Control for Reconfigurable Manufacturing Systems**, The University of Michigan, U.S.A and Mitsubishi Electric Corp, U.S.A, 2002.

8. Krishna K. Krishnan*, S. Hossein Cheraghi and Chandan , N. Nayak ,**Dynamic From-Between Chart: A New Tool For Solving Dynamic Facility Layout Problems,** InderScince Enterprise Ltd,2006.

9. Maher Lahmar, and Saifallah, Benjaafar, **Design of Dynamic Distributed Layouts,** University of Minnesota, December 12, 2002.

10. Norhashimah Morad**, Genetic Algorithms Optimization for the Machine Layout Problem,** University of Sains Malysia, Malysia, 2002.

11. Oliver De Weck, **Multidisciplinary System Design Optimization - Heuristic Techniques**, **A Basic Introduction To Genetic Algorithm,** Lecture 11, Massachusetts Institute of Technology, U.S.A, March 2004.

12. Oliver De Weck, **Multidisciplinary System Design Optimization Genetic Algorithms (III) Tabu Search**, Lecture 11, Massachusetts Institute of Technology, U.S.A, March 2004.

13. Parames Chutima, **Genetic Algorithm for Facility Layout Design with Unequal Departmental Areas and Different Geometric Shape Constraints**, hulalongkom University, Thailand, Thammasat Int.J.Sc.Tech.,Vo6,No2,May 2001.

14. Patrice Godefroid, Sarfraz Khurshid, **Exploring Very Large State Spaces Using Genetic Algorithms,** Massachusetts Institute of Technology, U.S.A, 2002.

15. Raymond R. Hill, **A Monte Carlo Study Of Genetic Algorithm Initial Population Generation Methods,** Department of Operational Sciences, Air Force Institute of Technology, Wright-Patterson AFB,1999.

16. Saifalah Benjafar,Sundersh S.Heragu,Sharukh A.Irani, **Next Generation Factory Layouts: Challenges And Recent Progress,** Troy NY, December 2000.

17. Sunderesh Heragu, Gang Meng1and Henk Zijm, Jan Kees vanmmeren**, Design and Analysis of Reconfigurable Layout Systems,** Renselear Polytechnic Institute, Troy, NY 2002.

18. Subhash Chandra[I] Abera Melesse[II], Vinay Pathak[III] **Genetic Algorithm Demystifies Facility Layout Problems (FLP) Using Post Order Sequence Of Nodes In A Slicing Tree,** Jimma University, 2003.

19. Tong Zhao and Chung-Li Tseng, **Flexible Facility Interior Layout: A real option Approach,** University of Missouri-Rolla, Rolla, U.S.A, 2002.

20. Tomahiro Fukaya, **HIPRAMS - Highly Productive and Re-Configurable Manufacturing Systems**, a Final Report, Intelligent Manufacturing Systems, June 2004.

21. P.P. Zouein and I.D. Tommelien, **Dynamic Layout Planning Using a Hybrid Incremental Solution Method,** Journal of construction management, November 1999.

22. Ronald W. Morrison, **Dispersion Based Population Initialization**, Mitreck Systems Inc, June 2002.