MODELING AND OPTIMIZATION OF FACILITY LAYOUT PROBLEM USING NATURE INSPIRED ALGORITHM

By

SYED HELAL UDDIN

MASTER OF SCIENCE IN INDUSTRIAL AND PRODUCTION ENGINEERING

BUET

DEPARTMENT OF INDUSTRIAL AND PRODUCTION ENGINEERING

BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY (BUET)

DHAKA, BANGLADESH

OCTOBER 2015

MODELING AND OPTIMIZATION OF FACILITY LAYOUT PROBLEM USING NATURE INSPIRED ALGORITHM

By

SYED HELAL UDDIN

A thesis submitted

to

The Department of Industrial and Production Engineering

in partial fulfillment for the degree of

Master of Science in Industrial and Production Engineering

MASTER OF SCIENCE IN INDUSTRIAL AND PRODUCTION ENGINEERING

BUET

DEPARTMENT OF INDUSTRIAL AND PRODUCTION ENGINEERING

BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY (BUET)

DHAKA, BANGLADESH

OCTOBER 2015

DECLARATION

It is hereby declared that this thesis or any part of it has not been submitted elsewhere for the award of any degree or diploma.

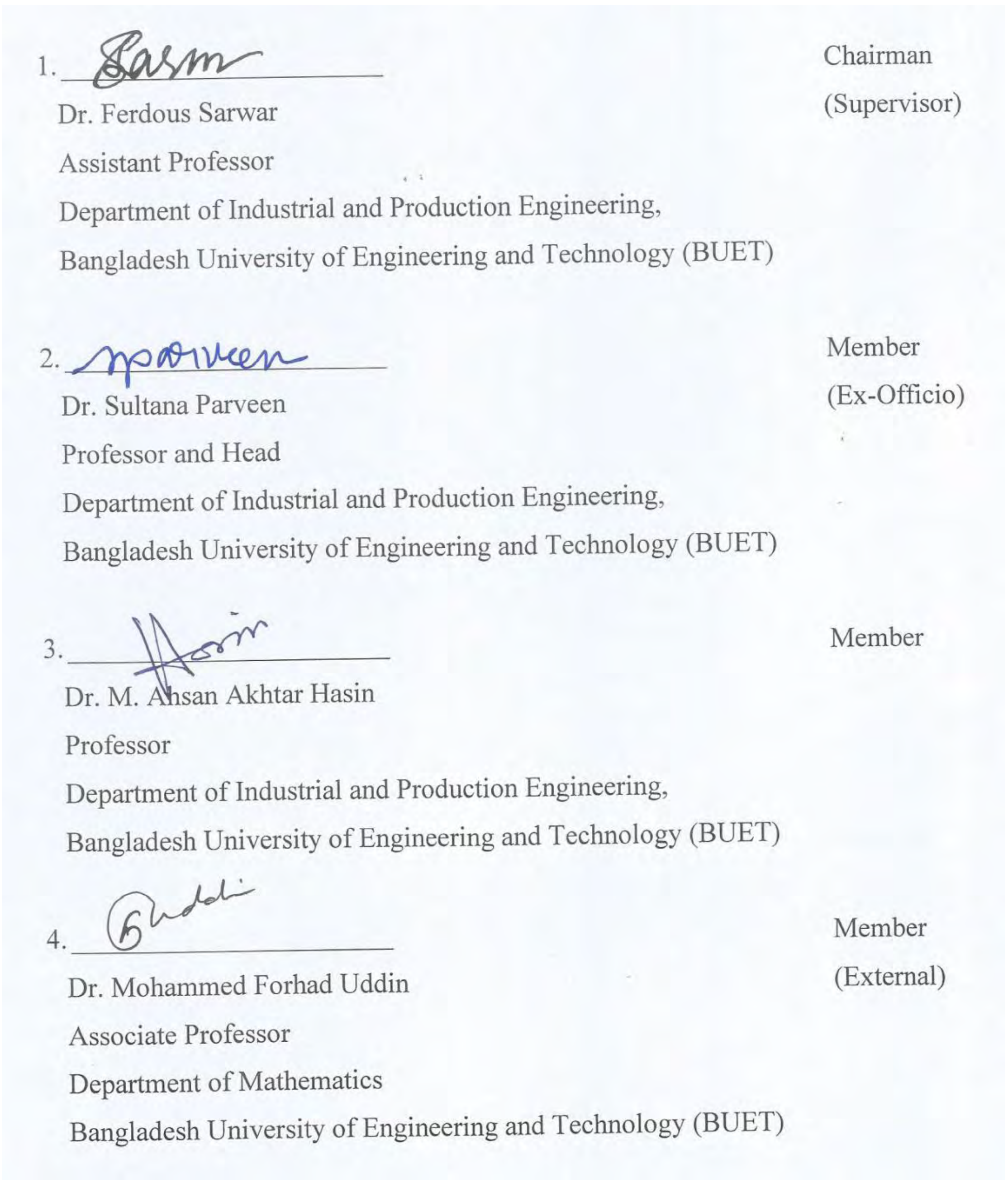
Signature of the Candidate

(Syed Helal Uddin)

CERTIFICATE OF APPROVAL

The Thesis titled " Modeling And Optimization Of Facility Layout Problem Using Nature Inspired Algorithm " submitted by Syed Helal Uddin, Student No: 0412082005; Session April 2012 has been accepted as satisfactory in partial fulfilment of the requirements for the degree of Master of Science in Industrial and Production Engineering on 21 October 2015.

BOARD OF EXAMINERS



1. [illegible] — Chairman (Supervisor)
Dr. Ferdous Sarwar
Assistant Professor
Department of Industrial and Production Engineering,
Bangladesh University of Engineering and Technology (BUET)

2. [illegible] — Member (Ex-Officio)
Dr. Sultana Parveen
Professor and Head
Department of Industrial and Production Engineering,
Bangladesh University of Engineering and Technology (BUET)

3. [illegible] — Member
Dr. M. Ahsan Akhtar Hasin
Professor
Department of Industrial and Production Engineering,
Bangladesh University of Engineering and Technology (BUET)

4. [illegible] — Member (External)
Dr. Mohammed Forhad Uddin
Associate Professor
Department of Mathematics
Bangladesh University of Engineering and Technology (BUET)

DEDICATION

To The Almighty

To my family

ACKNOWLEDGEMENT

All praise to Almighty Allah for giving me the ability to complete this thesis successfully.

I would like to express my sincere gratitude and deep appreciation to my supervisor, Dr. Ferdous Sarwar, Assistant Professor, Department of Industrial and Production Engineering, BUET. Under his continuous supervision, affectionate guidance, valuable suggestions, encouragement and inspiration throughout this work made this study possible.

I also express my profound thanks to the Head of the Department of Industrial and Production Engineering of BUET, Tanvir Hossain Bhuiyan, Imtiaz Ahmed, Assistant Professor of this department for rendering me assistance during my research work. I also like to thank all faculty and other members of the department for their kind assistance.

I am very much grateful to Noman H Chowdhury, Senior Lecturer, BRAC Business School, BRAC University, Dhaka, for his constant co-operation and assistance in developing the algorithm and computer coding.

I am thankful to my corps of Electrical and Mechanical Engineering of Bangladesh Army for selecting and allowing me to pursue in M.Sc. Engineering in BUET.

Finally, I like to thank my family whose continuous inspiration, sacrifice and support encouraged me to complete my study and research.

ABSTRACT

Productivity and efficiency of an organization greatly depends on how people plan, organize and utilize the facilities in that organization. From an upfront investment and recurring project expense, facilities planning are a critical issue in today's competitive manufacturing and service sectors. In addition to the upfront investment involved in facilities planning, there are operational issues that make facilities planning a critical issue. The most obvious impact is on material handling expenses. The impact of the facility layout goes beyond material handling costs. An effective facility layout implies that departments with high flow are close together. In facility layout problems, objective functions are modeled with different objectives in mind examples of which include minimization of cost or flow of materials, maximizations of closeness rewards etc. In this thesis, a mathematical model with a continuous representation of distance based adjacency matrix is developed. The resulting exact model will consider every all-rectangular-department solution. Solution from the new model is compared with solutions found from models based on binary based adjacency matrix. In this thesis, exact algorithm is used for finding feasible solution set from the total solution space. Further research can be done using other heuristic algorithms. In the function [$adjacency=1/k_1*e^{(k_2*x)}$] proposed in this thesis for generating continuous value adjacency matrix has two co-efficient, namely k_1(Denominator co-efficient) and k_2 (exponential co-efficient). Unit value for both of the co-efficient was assumed even of the fact that, there are strong rationales behind these two having industry specific values. There is huge scope of econometrical research to come up with series of values of k_1 and k_2 for different industries.

LIST OF ABBREVIATIONS

ERP	Enterprise Resource Planning
SCM	Supply Chain Management
JIT	Just-In-Time
FMS	Flexible Manufacturing Systems
WIP	Work-in-Process
FLP	Facility Layout Problem
QAP	Quadratic Assignment Problem
MIP	Mixed Integer Programming
DCTC	Distributed centroid-to-centroid
Bi-CLP	Bidirectional Circular Layout Problem
MWPG	Maximal Weighted Planar Graph
BATB	British American Tobacco Bangladesh

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS AND NOMENCLATURE

Weight of objective functions	α
Universal quantifier	$\forall$
Department indices ($i, j = 1, 2, \ldots, n$)	i, j
Dimension (axis) indices ($s = x, y$)	s
Total flow between departments i and j (multiplied with unit cost values if unit material handling cost differs among department pairs)	f_{ij}
Cost of flow between departments i and j	c_{ij}
Rectilinear distance between departments i and j	d_{ij}
Width ($s=x$) and length ($s=y$) of the facility in which the departments will be placed	L_s
s-axis coordinate of the center of department i	c_i
Binary variable showing whether department i is before department j in the sequence	z_{ij}

MODELING AND OPTIMIZATION OF FACILITY LAYOUT PROBLEM USING NATURE INSPIRED ALGORITHM

CHAPTER 1: INTRODUCTION

1.1 Introduction

The effective utilization of a company's facility is one of the key challenges facing plant managers. Facility utilization encompasses not only the utilization of facility space, but also the challenge of providing support for an efficient facility flow network. The cornerstone to both of these challenges is the facility layout. As such, solving the facility layout problem is a critical component to the competitiveness of a company.

The research outlined in this dissertation aims to provide an improvement in a methodology to help companies solve the critical problem of the facility layout problem.

1.2 Motivation

In the past 20 years, with rapidly increased global competition, elimination of waste and continuous productivity improvement have become more and more critical for manufacturing companies to run their business effectively and efficiently. Most of the business concepts and strategies arising recently, like Enterprise Resource Planning (ERP), Supply Chain Management (SCM), Just-In-Time (JIT) Manufacturing, Flexible Manufacturing Systems (FMS) and Lean Manufacturing, consider eliminating waste and continuous productivity improvement as their foundation.

The productivity and efficiency of an organization greatly depends on how people plan, organize and utilize the facilities in that organization. Facilities planning "determines how an activity's tangible fixed assets best support achieving the activity's objective" [1]. Thus, facilities planning have a great impact on the productivity and efficiency of running an organization.

"Since 1955, approximately 8% of the gross national product (GNP) has been spent annually on new facilities in the United States" [1]. Adding to this figure is the realization that many existing facilities are renovated each year, which yields an estimate of $250B spent each year on facilities planning and re-planning [1]. Thus, from an upfront investment and recurring project expense, facilities planning are a critical issue in today's competitive manufacturing and service sectors.

In addition to the upfront investment involved in facilities planning, there are operational issues that make facilities planning a critical issue. The most obvious impact is on material handling expenses. As suggested in [1], "effective facilities planning can reduce [material handling] costs by 10 to 30%."

However, the impact of the facility layout goes beyond material handling costs (which are likely to be a rather small cost in the facility). An effective facility layout implies that departments with high flow are close together. In addition to reducing material handling costs, this is also likely to reduce the material handling batch size. By reducing the material handling batch size, work-in-process inventory (WIP) will also decrease. Decreasing WIP has a direct cost implication (likely a large one) and is also likely to improve the lead time and quality of the product being moved (since feedback due to poor quality is shortened along with lead time). Finally, companies that are able to simultaneously shorten lead time, improve quality, and reduce their costs are much more likely to have increased opportunities for their product. Thus, the impact of facilities planning goes significantly beyond the impact on material handling expenses (e.g., productivity ratios concerning manufacturing cycle, aisle space, and energy [2]. In summary, facilities planning have an impact on many aspects of the company, either directly or indirectly.

1.3 Components and layout

The main components of facilities planning include facility location, facility system design, facility layout design, and material handling system design. As one of the critical steps in facilities planning, the facility layout design is "concerned with determining the 'most efficient' arrangement of interacting departments within a designated section of a building subject to constraints imposed by the site plan, the building, the departmental area, service requirements, and the decision-maker" [3].

The facility layout problem (FLP) has broad applications, from a new hospital to an assembly line, from an existing warehouse to the baggage department in an airport, from an office to a retail store. In manufacturing, the facility layout design involves the determination of how to design the physical layout of manufacturing facility systems to provide the best support for production.

More specifically, the facility layout procedure traditionally includes two phases: the block layout phase and the detailed layout phase. The block layout phase specifies the relative location and size of each department (see Figure 1.1(a)). Based on the block layout output, the detailed layout phase determines exact department locations, aisle structures, input/output (I/O) point locations, and the layout within each department (see Figure 1.1(b)).

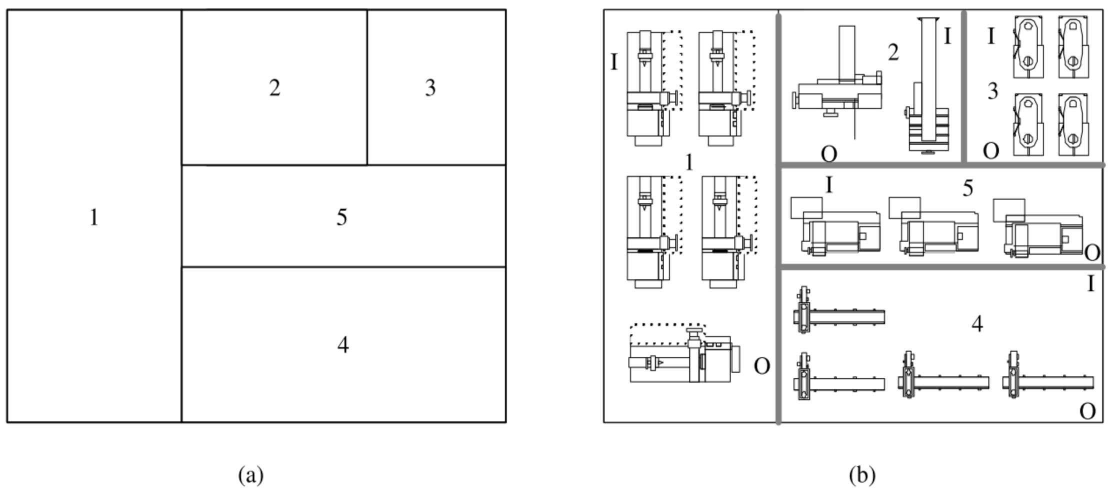


Figure 1: Facility Layout Solutions in (a) Block Layout and (b) Detailed Layout

1.4 Research Objectives

We state the objectives of our research in this section.

1. To develop a mathematical model with a continuous representation of distance based adjacency matrix. The resulting exact model will consider every all-rectangular-department solution.

2. To compare the performance of the proposed model with previous model based on reward functions, where adjacency matrix is constructed with binary values.
3. To develop a MATLAB-based program for implementing and testing the model in a real world scenario

1.5 Outline of Methodology/Experimental Design

The proposed research methodology is outlined below:

- At first traditional layout facility problems will be investigated in general to understand the types, severity and frequency of the changes in the environment of a layout.
- Based on the methodologies for layout design used in industrial facilities, decision variables and constraints will be identified.
- A Mathematical model for multi objective mixed integer facility layout problem will be proposed.
- MATLAB programming will be used to solve the problem based on a nature inspired algorithm.
- The proposed heuristic will be compared to the results obtained using traditional method.
- The performance of the proposed model will be determined by improvements in traditional methods demonstrating the solution of one specially built problem.
- The improved version of the model will be used to solve a real life facility layout problem.

CHAPTER 2: LITERATURE REVIEW AND THEORETICAL BACKGROUND

Since an efficient facility layout is critical for high productivity and quality manufacturing, a lot of research has been performed — and is still being performed — in this area. However, the extremely complicated nature underlying the FLP, various application and implementation issues, as well as the continuously increasing requirements from industry, lead us to the conclusion that the research in the FLP is still far from being “well done.” As a result, research related to the FLP continues to be one of the academic focus areas in industrial engineering and operations research. Developing some cutting-edge algorithms for the FLP is not only important to academia, but also to industry.

In the FLP research literature, a variety of approaches are proposed to solve this combinatorial optimization problem. These approaches are different in terms of layout representation, objective functions, constraints, algorithm search strategies, etc. One of the most widely used classification methods for these approaches is to divide them into two categories: exact algorithms and heuristics. Another important classification is based on layout representation: discrete or continuous. In this chapter we give a detailed literature review of FLP research based on this two-level classification. First, we classify the literature into exact algorithms and heuristics. Second, in each of these two categories, the literature is further classified and reviewed with respect to their layout representation.

2.1 Theoretical Background

2.1.1 Genetic Algorithms for the Facility Layout Problem

The encoding is done through the structure named chromosomes, where each chromosome is made up of units called genes.

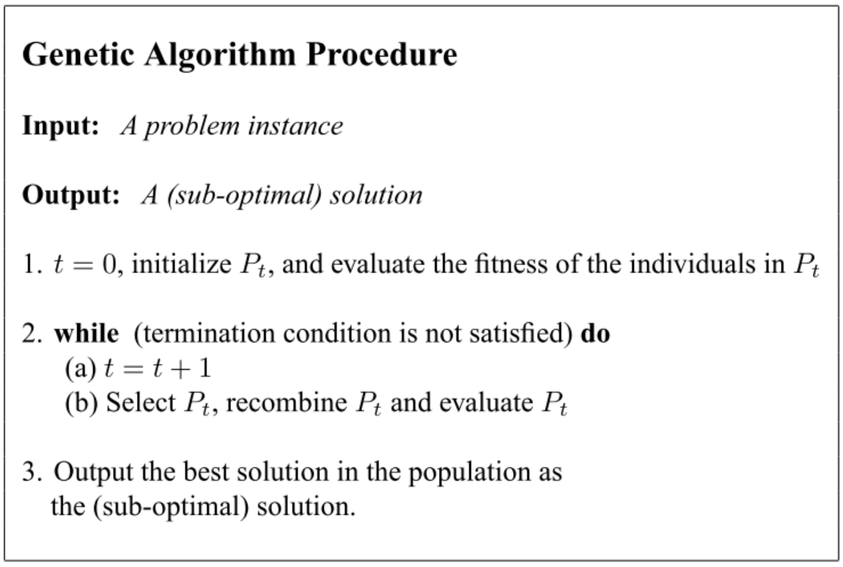
There are some determining factors that strongly affect the efficiency of genetic algorithms:

1. The representation of the solutions by strings.
2. The generation of the initial population.
3. The selection of individuals in an old population (parents) that will be allowed to affect the individuals of a new population.
4. The genetic operators that are used to recombine the genetic heritage from the parents to produce children. The most often-used operators are the crossover and the mutation.

The selection of individuals that will be allowed to affect the following generation is based on the fitness of the individuals. This is done in such a way that individuals with better fitness are more likely to be chosen to become parents. The recombination of the population consists of the following four operations:

1. Crossover. By combining the coded solution strings of two parents two children are created. If one considers the biological origin of the genetic algorithms it makes sense to denote the coded solution string "genome" and look at this procedure as a result of mating. To avoid chaotic behavior, not all individuals in the new population are generated by this operator. The probability of applying this operator (crossover rate) is denoted by pc.
2. Mutation. In order to give the populations new impulses some random changes in the genomes are allowed to occur. The mutation operator changes a "gene" in a solution with a probability (mutation rate) pm.
3. Local search. It has proven very efficient to search for locally optimal solutions in the neighborhood of the children. If one is able to find a better solution then it will replace the original child as a member of the new population.
4. Control of new individuals. It is not unlikely that a child will have worse fitness than its parents. In that case the child might not be accepted in the new generation.

Let us note also that a GA implementation requires the specification of certain parameters such as population size, and number of generations. Let Pt denote the population at time t. Then the genetic algorithm procedure can be described as in Figure below.



Genetic Algorithm Procedure

Input: *A problem instance*

Output: *A (sub-optimal) solution*

```
1. $t = 0$, initialize $P_t$, and evaluate the fitness of the individuals in $P_t$

2. while (termination condition is not satisfied) do
     (a) $t = t + 1$
     (b) Select $P_t$, recombine $P_t$ and evaluate $P_t$

3. Output the best solution in the population as
   the (sub-optimal) solution.
```

Figure 2: Genetic Algorithm Procedure

We continue with the description of various implementations of the genetic algorithm for the facility layout problem.

As we have seen in the section of SA for the facility layout problem, Tam [64] uses a simulated annealing approach to solve the inter-cell problem. The same author using the same problem formulation and representation of the floor plan layout as a slicing tree, attempts a solution approach to the problem using Genetic Algorithms. In applying a GA an important part of the implementation is the coding of solutions as strings of finite length. For the problem formulation under consideration, a slicing tree can be generated by a string using as its elements the nodes of the tree in a sequence which starts from the bottom level nodes and ends at the root of the tree. The nodes of the tree represent either facility identifications (operands) or "cut" symbols (operators). The proposed GA uses for the recombination of the population the crossover and mutation operators, as described for the general genetic algorithm. For the selection of the new population the reproduction operator is used. Under this operator the chance of being selected to remain in the new population P_{t+1} is proportional to the fitness value of the individual.

GA was run for 150 generations with 10 different sets of initial solutions. The best and average solution in each generation was gathered. The performance of GA was compared with that of a hill climbing method (HC), which searches through a neighborhood N, where N is the set of operator sequences generated from changing one operator. GA outperformed HC both in terms of minimum and average costs. For the 30-facility layout GA improved the minimum cost by 10:5% and the average cost by 13%.

Koakutsu and Hirata [61] propose an interesting combined approach called genetic simulated annealing (GSA) for the solution of the floor plan design of VLSI (Very Large Scale Integrated) circuits. The problem involves the arrangement of a given set of rectangular modules (with no fixed shapes or dimensions) in the plane, with the objective to minimize:

(1) The area of the enclosing rectangle which should contain all the modules, and (2) the total wire length between modules that should be connected in the circuit. The main features of the algorithm are the following:

- Stochastic Optimization: GSA uses the stochastic optimization used in simulated annealing so that a neighbor state for which there is an increase of the cost function is accepted with a certain probability.
- Multiple Search Paths: A population of solutions corresponding to the population of GAs is used to initialize the search in multiple directions. The stochastic optimization is applied to each solution of the population.
- Selection of search paths: The selection operator replaces solutions which have value higher than the average value of the population, with solutions that have lower cost value than the average value of the population. This way, paths which are expected to reach good solutions are selected.
- Genetic Operators: A genetic crossover operator is used to generate new solutions.

The formulation of the problem represents the floor plan layout as a slicing tree. The representation of a solution as a string is similar to the one described previously, using in this case, vertical and horizontal cuts with corresponding branching operators.

GSA is tested on three floor plan problem instances. The first has 16 modules, each one a fixed square of unit area, having wires connecting to its horizontal and vertical neighbors.

The second problem has 16 modules and 25 wires, and the third one has 20 modules and 31 wires. For the last two problems the total module area is 100. The proposed algorithm was compared to a regular SA algorithm. Both algorithms run 100 times with different initial solutions for each of the above problem instances. The average costs are used for the comparison. The results show that GA improves the average cost by 1.7% - 9.8% compared to the SA within the same computational time.

More recently Banerjee and Zhou [62] developed a genetic algorithm to solve a variation of Montereuil's mixed integer programming formulation for the FLP [46], and in particular for the special case of single loop material flow path configuration. They introduce a "knowledge-augmented mutation operator" to determine the flow path direction, which appears to perform well for the cases where the layout has very low flow path dominance.

Previous applications of GA for facilities layout design can be found in [63] from the same authors and Montreuil.

Tate and Smith [64] applied GA using an adaptive penalty function to the unequal-area facility layout problem with shape constraints. The rectangular area in which the facilities are to be located is divided into vertical bays of different width and each bay is divided into rectangular departments of different length. The encoding of the solutions to strings is done with two distinct chromosomes. The first one is the sequential chromosome which is represented by a permutation of the set N = {1, 2, ….., n), where n is the number of departments. The sequence of the permutation starts by reading departments bay to bay, from top to bottom and from left to right at the rectangular area. The second chromosome

is the bay chromosome where each gene shows for each bay the number of departments contained in the previous bays including the involved one, showing this way the breaks that occur in the sequence between bays. For example, consider 4 bays having 3, 4, 6 and 2 departments respectively starting from the left bay. Then using the bay chromosome the solution encoding is (3; 7; 13). Note that the last breakpoint at 15 is obvious. The proposed GA uses variants of crossover and mutation operators.

- The variant of the crossover operator works as follows: using two individuals to be the parents, one offspring (child) is generated by the following rules. For the case of GA encoding using the sequential chromosome, each location in the child' s sequence is the department number in the corresponding location from one of the parents, both having the same probability to be selected. This will force the common locations in the sequences of the parents to be carried over to the child. Also each department must occur only once in the child. For the bay chromosomes, the location and number of bay breaks in the child's sequence is taken from one of the parents, both having equal probabilities to be selected.
- The mutation uses three different operators. Two of the operators alter the number of bays affecting only the bay chromosome and one operator reverses a subsequence of the departments affecting the sequence chromosome.

The evolution parameters, i.e. the population size, and the crossover and mutation rates are determined after several trial runs. An adaptive penalty function is used to find good feasible solutions. The penalty function is adaptive because during the course of the algorithm it uses observed population data to adjust the level of the penalty that is applied to the infeasible solutions. Test problems with size ranges from 10 to 20 departments were used to evaluate the efficiency of the proposed genetic algorithm. The proposed approach proved to be the best in terms of quality solution when compared with previous published results for the problems under consideration.
Genetic algorithms are inherently parallel in nature. Several implementations of GA in parallel environments have recently appeared, introducing in this way a new group of GA, the Parallel Genetic Algorithms (PGA). The population of a parallel genetic algorithm is divided into subpopulations. Then an independent GA is locally performed on each of these subpopulations, and the best solutions in each case are transferred to all the other subpopulations. Two types of communication are established among the subpopulations. Either among all nodes where the best solution

of each subpopulation is broadcasted to all the other subpopulations, or among the neighboring nodes, where only the neighboring subpopulations receive the best solutions.

The most important features of PGA, which result in a considerable speedup relative to sequential GAs, are the following:

- Local selection: In sequential GAs the selection operation takes place by considering the whole population. In a PGA this operation is performed locally by the selection of an individual in a neighborhood.
- Asynchronous behavior: It allows the evolution of different population structures at different speeds, resulting in an overall improvement of the algorithm in terms of computational time.
- Reliability in computation performance: The computation performance of one processor does not affect the performance of the other processors.

Several implementations of PGA have been proposed for the solution of the quadratic assignment problem. An application of an asynchronous parallel GA called ASPARA-GOS has been presented by Muhlenbein [65] for the QAP, introducing a poly-sexual voting recombination operator. The PGA was tested on QAPs of size 30 and 36 with known solutions. The algorithm found a new optimum for the Steinberg's problem (QAP of size 36). The numbers of processors that were used to run this problem were 16, 32 and 64. The 64 processors implementation (on a system with distributed memory) gave by far the best results in terms of computational time. Furthermore, Huntley and Brown [66] developed a parallel hybrid of SA and GA to solve the QAP approximately. A parallel genetic algorithm is used to produce a good initial solution for each population and the SA algorithm is used for improving these solutions. More recently, Battiti and Tecchiolli in [67] developed parallelization schemes of genetic algorithms for quadratic assignment problems presenting indicative experimental results.

2.1.2 Simulated Annealing for the Facility Layout Problem

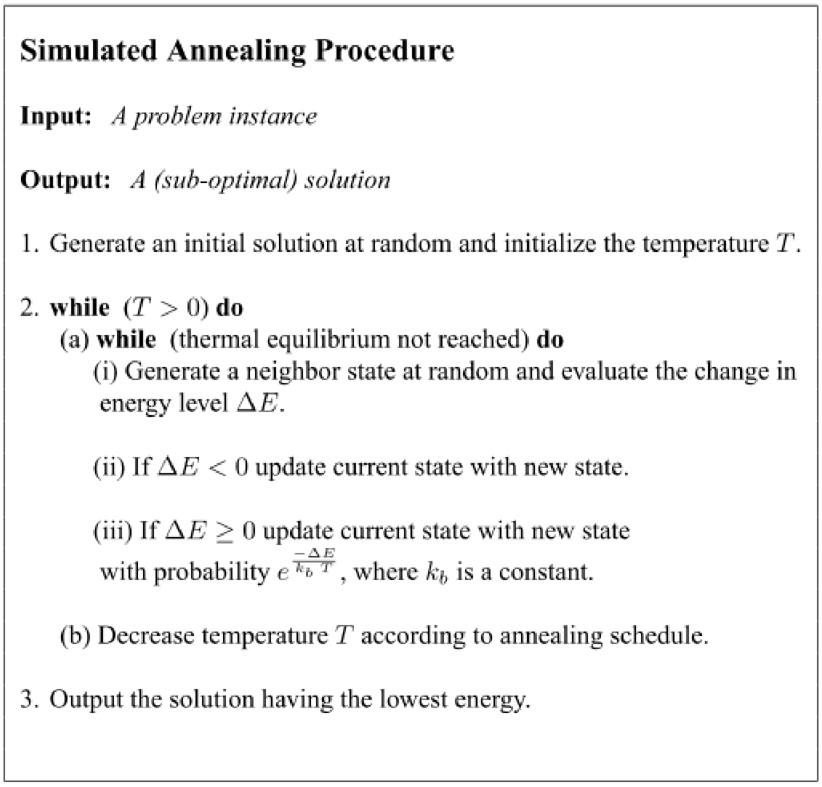
Simulated annealing was first proposed by Kirkpatrick et al. [68] as a method for solving combinatorial optimization problems. The name of the algorithm derives from an analogy between the simulation of the annealing of solids first proposed by Metropolis et al. [69], and the strategy of solving combinatorial optimization problems. Annealing refers to a process of cooling material slowly until it reaches a stable state. Starting from an initial state, the system is perturbed at random to a new state in the neighborhood of the original one, for which a change of ¢E in the objective function value (OFV) takes place. In a minimization process if the change ¢E is negative then the transformation to the new state is accepted.

If $\Delta E \geq 0$ the transformation is accepted with a certain probability of $p(\Delta E) = e^{\frac{-\Delta E}{k_b T}}$, where T is a control parameter corresponding to the temperature in the analogy and k_b is Boltzmann' s constant. The change ¢E in the OFV corresponds to the change in the energy level (in the analogy) that occurs as the temperature T decreases. SA gives us a mechanism for accepting small increases in the objective function value, controlling though the probability of acceptance $p(\Delta E)$ through the temperatures. Kirkpatrick et al. [68] argue that allowing "hill climbing" moves, one can avoid configurations that lead to locally optimal solutions and eventually higher quality solutions can be obtained. So the main advantage of the simulated annealing method is its ability to escape from local optima.

The main features of the SA method are:

- The temperature T, which is the parameter that controls the probability $p(\Delta E)$ of accepting a cost-increasing interchange. During the course of the algorithm T is decreased in order to steadily reduce the probability of acceptance of interchanges that increase the value of the objective function,
- The equilibrium, i.e. the condition in which a further improvement in the solution using additional interchanges is highly unlikely to occur,
- The annealing schedule that determines when and by how much the temperature is to be reduced.

A pseudo-code of the simulated annealing procedure is given in Figure 1 [54].



```
Simulated Annealing Procedure

Input:  A problem instance

Output:  A (sub-optimal) solution

1. Generate an initial solution at random and initialize the temperature $T$.

2. while ($T > 0$) do
   (a) while (thermal equilibrium not reached) do
       (i) Generate a neighbor state at random and evaluate the change in
        energy level $\Delta E$.

       (ii) If $\Delta E < 0$ update current state with new state.

       (iii) If $\Delta E \geq 0$ update current state with new state
        with probability $e^{\frac{-\Delta E}{k_b T}}$, where $k_b$ is a constant.

   (b) Decrease temperature $T$ according to annealing schedule.

3. Output the solution having the lowest energy.
```

Figure 3: Simulated Annealing Procedure

Several implementations of the simulated annealing algorithm have been proposed for the facility layout problem. We will present the main concepts of the most recent approaches and comment on the computational results.

Heragu and Alfa in [70], present an extensive experimental analysis of two simulated Annealing based algorithms, implementing them on two patterns of layout, the single-row and multi-row facility layouts. The first algorithm uses the standard techniques of the SA heuristic. In the main step the algorithm examines the random exchange of the positions of two facilities. The new solution is accepted if the exchange results in a lower OFV.

Otherwise, the difference ΔE between the OFV of the best solution obtained so far and the current solution is computed. This solution is accepted with probability $e^{\frac{-\Delta E}{T}}$. This step is repeated 100n times or until the number of new solutions accepted is equal to 10n, where n is the number of facilities in the layout problem. Next, the algorithm decreases the value of temperature T by multiplying it by the cooling ratio r and repeats the main step.

The stopping criterion is a fixed maximum number of temperature change steps. The initial temperature T is set as a number sufficiently larger than the largest ΔE encountered for problems tested with other heuristics.

The second algorithm presented in the same thesis is a hybrid SA algorithm (HSA), which uses a "core" algorithm to generate a "good" initial solution, and then improves it using the SA algorithm described before. The core algorithm is a modified penalty algorithm (MP) presented in [71]. Eight test problems of size up to 30 (available in the literature) are used for the single-row case. Each test problem is solved 10 times using the same initial solution.

For six of the problems the HSA algorithm produces optimal or best-known solutions. For the remaining two problems, the solutions are better than those previously reported in the literature. A comparison between the HSA and the SA algorithms is presented, as well as with three other heuristic algorithms (a 2-way exchange, a 3-way exchange and a Wilhem-Ward version of simulated annealing [72]) using 15 equal-area multi-row FLPs. The HAS in terms of solution quality, performed better than all the other algorithms though requiring more computational time than the SA algorithm. Also as the number of annealing runs increases, SA seems to produce similar quality solutions with HSA with less computational effort.

Another implementation of the SA algorithm applied to the cellular layout problem can be found in [73]. This problem involves the determination of the relative positions of n equi-dimensional manufacturing entities which may represent either the set of machines belonging to a cell (intra-cell problem) or the manufacturing cells within a shop (inter-cell problem). The objective of both layout problems is to minimize the total material flow (cost) between the manufacturing entities. The method presented in the paper is called CLASS, which stands for Computerized Layout Solutions using simulated annealing. The proposed algorithm is a regular simulated annealing algorithm with the following most important elements:

Solution space: The solution space consists of an n X n grid, i.e. n^2 positions are available to be occupied by the n entities. The distance between all pairs of positions is determined using geometric or Manhattan distances.

Interchanges: The interchange given a solution can be either a move of an entity from its current position to an unoccupied position or an "exchange" of the positions of two entities. The two positions from the solution space that are exchanged are selected randomly.

Annealing schedule: The annealing schedule considers the initial temperature to be sufficiently large so that all interchanges are eventually accepted. The temperature is reduced by multiplying it with a constant that takes values between 0 and 1.

Parameters: The number of interchanges to be attempted at each temperature, the number of accepted interchanges at each step and the total number of temperature change steps are 100n, 10n and 100 respectively.

Interchange Acceptance Criterion: The interchange is accepted if a randomly generated number between 0 and 1 is less than the value of $e^{\frac{-\Delta E}{T}}$, where ΔE and T are respectively the difference in the OFV and the temperature at the current step.

CLASS was compared to twelve other layout methods in terms of both the quality of the solution and the speed of convergence. Eight problems available in the literature were used for the comparison of the algorithms, with sizes between n = 5 and n = 30. In each case CLASS either equals the performance of, or outperforms each of the other methods. The sensitivity of CLASS to the initial conditions was tested by running each of the test problems of sizes 5, 6, 7, and 8, five times, each time with a different initial solution. The optimal solution was obtained in each case, indicating the insensitivity of the solution quality to the initial conditions.

For the inter-cell problem Tam describes a SA solution approach which takes into consideration the traffic between cells, the geometric constraints of the individual cells and any occupied regions on the floor plan. The objective is to find a layout that minimizes the weighted flow of parts between the manufacturing cells while satisfying the area and shape constraints of the individual cells. There are several critical points concerning the problem formulation:

- Layout representation: The layout takes the form of a slicing structure, which is represented by a slicing tree. This is a binary tree representing the recursive partitioning process of a rectangular area, through cuts. A cut specifies the relative position of the departments (left, right, below or above each other) through four distinguished branching operators.
- Solution space: The solution space is defined as the set S which consists of all slicing trees that can be generated by rearranging cuts of a given structure. It is shown that$|S| = 4^{n-1}$, where n is the number of cells and the size of the neighborhood N is $|N| = 4n - 5$.
- Area constraints: The location where a rectangular partition is cut, i.e. the cut point, must be chosen so that the split partitions receive their required areas. The cut point is determined in a top-down fashion starting from the "root" of the tree.
- Shape constraints: The cell's shape is described using the aspect ratio and the dead space ratio. The first ratio is the height over the width of the partition allocated to a cell. The second ratio is used to measure the amount of unusable space within the partition allocated to a manufacturing cell. Both ratios have lower and upper bounds.
- Slicing tree construction: Using numerical clustering techniques a slicing tree is constructed in such a way that cells with large inter-cell traffic volume are placed in close proximity with each other.

The attractive element of the algorithm is that it exploits the hierarchical representation of the layout, so that the probability of selecting a neighborhood state is not uniformly distributed (as in a regular SA algorithm), but is dependent on T. More particularly, when T is high at the first steps of the annealing procedure, a cut near the root of the slicing tree will be selected, causing large swings in the cost function value since a large number of cells will have to be relocated. As T decreases during the course of the algorithm, cuts that are located at a lower level in the tree are selected, to generate a neighborhood state. So a guided search in the set of neighboring solutions is adopted. The algorithm was compared to two other local search methods, denoted as HC (a straightforward hill climbing method) and BC (a modified version of HC). Two test problems of size n = 20 and n = 30 were constructed for the comparison. Each method was run 10 times with different initial solutions. The computation time was kept the same among the three methods. In terms of solution quality the proposed SA algorithm outperformed the other two methods, both in average and minimum cost.

Kouvelis and Chiang address the single row layout problem (SRLP) in flexible manufacturing systems (FMSs). The problem deals with the optimal arrangement of n machines along a straight track with a material handling device moving jobs from one machine to another. The difficulty of the problem is due to the variety of parts to be processed in different ranges of operation sequences. When the sequence of operations of a job is not the same as the sequence of the locations of the machines, the job sometimes has to travel in reverse (backtrack) in order to receive the required operations. The objective of the SRLP is to find the ordering of the machines that minimizes the total backtracking distance of the material handling device. If we consider n machines and n candidate locations for the machines to be placed, the solution to the SRLP is one of the possible permutations of the set S = {1,2,….n}defined as the set of the workstation assignment vectors, each one representing a configuration of the machines in a single row. The neighborhood of a configuration is the set N of configurations resulting by the interchange of the locations of two machines. The initial configuration is obtained by randomly assigning machines to locations. For the setting of the parameters of the SA algorithm, i.e. the initial acceptance probability (through which the initial temperature will be calculated), the number of interchanges attempted before the reduction of the temperature, the value of the cooling ratio, and the number of steps to reach the equilibrium, a sensitivity analysis was performed with respect to each individual parameter. For each parameter a range of values is tested while all other parameters are held fixed. The best values of the parameters are kept as the final ones to be used in the algorithm. The experimental analysis showed that fine-tuning of the SA parameters with respect to each specific application and the selection of the initial

solution is very important for the performance of the algorithm in terms of quality solution.
The same authors and J. Fitzsimmons describe two distinct implementations of the simulated annealing algorithm for machine layout problems in the presence of zoning constraints. These constraints are restrictions on the arrangement of machines. Positive zoning constraints require that certain machines have to be placed near each other, while negative zoning constraints do not allow certain machines to be in close proximity. The problem is formulated as a restricted quadratic assignment problem. Assuming that the number of candidate locations is equal to the number of machines, the objective is to assign the machines to the locations in a way that the cost function is minimized with respect to the zoning constraints. The first of the SA algorithms called the Compulsion Method takes into consideration the zoning constraints mostly during the search for a new layout in the neighborhood of the original one. The second algorithm, the Penalty Method, takes into account the presence of the zoning constraints in the objective function through the use of appropriate penalty terms. For each layout that violates any of the zoning constraints, corresponding penalty terms are charged in the OFV. The two versions are compared on an extensive set of computational experiments using test problems of size ranges from 5 to 30 machines. The results showed that the Compulsion Method outperforms the Penalty Method in terms of CPU time and solution quality. The basic advantage of the Penalty Method is that it can be easily changed to handle the addition of extra zoning constraints.
Meller and Bozer describe a Simulated Annealing Based Layout Evaluation algorithm (SABLE), which introduces a new generator routine for candidate layout solutions, combined with the use of space-filling curves. The algorithm is implemented on a set of single and multiple floor facility layout problems. For the single-floor case test problems of sizes 11 to 25 are used. An average and a worst-case analysis shows that the proposed algorithm performs the best in terms of solution quality. Additionally, SABLE performed better than Tam's SA algorithm on a data set of 20 and 30-size department single-floor FLPs. Let us note that regarding the department shapes, Tam's algorithm generally assumes rectangular shapes, while the proposed algorithm

tends to generate departments with non-rectangular shapes. For the multi-floor case, test problems with up to 4 floors and 40 departments were used to evaluate the performance of SABLE.

The results indicate the robustness of the algorithm to changes in the vertical to horizontal ratio.

For the special case of QAP several SA approaches have been proposed. Burkard and Rend were the first to apply simulated annealing for solving the QAP. They reported on rather favorable computational results indicating that the obtained solutions deviate only 1-2% from the best known solutions. Wilhelm and Ward also applied the SA algorithm to quadratic assignment problems, by further experimenting on the procedure.

They report on the sensitivity of SA to the control parameters, and evaluate the algorithm using problems ranging in size from n = 5 to n = 100. In particular computational results were provided for the test problems in Nugent et al. and for two test problems they introduced in the paper. Connolly discusses the implementation of SA on 7 problems. The computational results indicate that examining sequentially generated neighboring solutions, rather than randomly generated ones, makes the SA algorithm more efficient. More recently Laursen investigated the performance of the SA algorithm by varying two parameters: (1) the number of simulations, and (2) the simulation length, while in both cases the algorithm uses the same computational time for a specific instance problem. Laursen concluded that the length of each simulation is optimizable and that a large range of its values generate a near-optimal solution quality.

2.2 Recent Nature Inspired Algorithms

2.2.1 Particle swarm optimization (PSO)

Particle swarm optimization (PSO) is a population based stochastic optimization technique developed by Dr. Eberhart and Dr. Kennedy in 1995, inspired by social behavior of bird flocking or fish schooling.

PSO shares many similarities with evolutionary computation techniques such as Genetic Algorithms (GA). The system is initialized with a population of random solutions and searches for optima by updating generations. However, unlike GA, PSO has no evolution operators such as crossover and mutation. In PSO, the potential solutions, called particles, fly through the problem space by following the current optimum particles.

Each particle keeps track of its coordinates in the problem space which are associated with the best solution (fitness) it has achieved so far. (The fitness value is also stored.) This value is called *pbest*. Another "best" value that is tracked by the particle swarm optimizer is the best value, obtained so far by any particle in the neighbors of the particle. This location is called *lbest*. When a particle takes all the population as its topological neighbors, the best value is a global best and is called *gbest*.

The particle swarm optimization concept consists of, at each time step, changing the velocity of (accelerating) each particle toward its p*best* and *lbest* locations (local version of PSO). Acceleration is weighted by a random term, with separate random numbers being generated for acceleration toward *pbest* and *lbest* locations.

In past several years, PSO has been successfully applied in many research and application areas. It is demonstrated that PSO gets better results in a faster, cheaper way compared with other methods.

Another reason that PSO is attractive is that there are few parameters to adjust. One version, with slight variations, works well in a wide variety of applications. Particle swarm optimization has been used for approaches that can be used across a wide range of applications, as well as for specific applications focused on a specific requirement.

[http://www.swarmintelligence.org/]

2.2.2 Firefly Algorithm

Firefly Algorithm (FA) was first developed by Xin-She Yang in late 2007 and 2008 at Cambridge University, which was based on the flashing patterns and behavior of fireflies. The theoretical side of this algorithm is explained in Chapter 6.

2.2.3 Cuckoo search

Cuckoo search (CS) is an optimization algorithm developed by Xin-she Yang and Suash Deb in 2009. It was inspired by the obligate brood parasitism of some cuckoo species by laying their eggs in the nests of other host birds (of other species). Some host birds can engage direct conflict with the intruding cuckoos. For example, if a host bird discovers the eggs are not their own, it will either throw these alien eggs away or simply abandon its nest and build a new nest elsewhere. Some cuckoo species such as the New World brood-parasitic Tapera have evolved in such a way that female parasitic cuckoos are often very specialized in the mimicry in colors and pattern of the eggs of a few chosen host species.
Cuckoo search idealized such breeding behavior, and thus can be applied for various optimization problems. It seems that it can outperform other meta-heuristic algorithms in applications.
2.2.4 Cuckoo search (CS) uses the following representations
Each egg in a nest represents a solution, and a cuckoo egg represents a new solution. The aim is to use the new and potentially better solutions (cuckoos) to replace a not-so-good solution in the nests. In the simplest form, each nest has one egg. The

algorithm can be extended to more complicated cases in which each nest has multiple eggs representing a set of solutions.

CS is based on three idealized rules:

- Each cuckoo lays one egg at a time, and dumps its egg in a randomly chosen nest;
- The best nests with high quality of eggs will carry over to the next generation;
- The number of available hosts nests is fixed, and the egg laid by a cuckoo is discovered by the host bird with a probability p_a \in (0, 1). Discovering operate on some set of worst nests, and discovered solutions dumped from farther calculations.

In addition, Yang and Deb discovered that the random-walk style search is better performed by Lévy flights rather than simple random walk.

[Wikipedia]

2.2.4 BAT algorithm

Bats are fascinating animals. They are the only mammals with wings and they also have advanced capability of echolocation. It is estimated that there are about 996 different species which account for up to 20% of all mammal species. Their size ranges from the tiny bumblebee bat (of about 1.5 to 2g) to the giant bats with wingspan of about 2 m and weight up to about 1 kg. Microbats typically have forearm length of about 2.2 to 11cm. Most bats uses echolocation to a certain degree; among all the species, microbats are a famous example as microbats use echolocation extensively while megabats do not.

Microbats use a type of sonar, called, echolocation, to detect prey, avoid obstacles, and locate their roosting crevices in the dark. These bats emit a very loud sound pulse and listen for the echo that bounces back from the surrounding objects. Their pulses vary in properties and can be correlated with their hunting strategies, depending on the species. Most bats use short, frequency-modulated signals to sweep through about an octave, while others more often use constant-frequency signals for echolocation. Their signal bandwidth varies depends on the species, and often increased by using more harmonics.

Though each pulse only lasts a few thousandths of a second (up to about 8 to 10 ms), however, it has a constant frequency which is usually in the region of 25 kHz to 150 kHz. The typical range of frequencies for most bat species are in the region between 25kHz and 100kHz, though some species can emit higher frequencies up to 150 kHz. Each ultrasonic burst may last typically 5 to 20 ms, and microbats emit about 10 to 20 such sound bursts every second. When hunting for prey, the rate of pulse emission can be sped up to about 200 pulses per second when they fly near their prey. Such short sound bursts imply the fantastic ability of the signal processing power of bats. In fact, studies shows the integration time of the bat ear is typically about 300 to 400 μ_s. As the speed of sound

in air is typically v = 340 m/s, the wavelength λ of the ultrasonic sound bursts with a constant frequency f is given by $\lambda = v/f$, which is in the range of 2mm to 14mm for the typical frequency range from 25kHz to 150 kHz. Such wavelengths are in the same order of their prey sizes.

Studies show that microbats use the time delay from the emission and detection of the echo, the time difference between their two ears, and the loudness variations of the echoes to build up three dimensional scenario of the surrounding. They can detect the distance and orientation of the target, the type of prey, and even the moving speed of the prey such as small insects. Obviously, some bats have good eyesight, and most bats also have very sensitive smell sense.

In reality, they will use all the senses as a combination to maximize the efficient detection of prey and smooth navigation. However, here we are only interested in the echolocation and the associated behavior. Such echolocation behavior of microbats can be formulated in such a way that it can be associated with the objective function to be optimized, and this makes it possible to formulate new optimization algorithms.

If we idealize some of the echolocation characteristics of microbats, we can develop various bat-inspired algorithms or bat algorithms. In the basic bat algorithm, the following approximate or idealized rules were used.

1. All bats use echolocation to sense distance, and they also 'know' the difference between food/prey and background barriers in some magical way;

2. Bats fly randomly with velocity v_i at position x_i with a frequency f_{min}, varying wavelength λ and loudness A_0 to search for prey. They can automatically adjust the wavelength (or frequency) of their emitted pulses and adjust the rate of pulse emission r ∈ [0, 1], depending on the proximity of their target;

3. Although the loudness can vary in many ways, we assume that the loudness varies from a large (positive) A_0 to a minimum constant value A_{min}.

Another obvious simplification is that no ray tracing is used in estimating the time delay and three dimensional topography. Though this might be a good feature for the application in computational geometry, however, we will not use this feature, as it is more computationally extensive in multidimensional cases.

In addition to these simplified assumptions, we also use the following approximations, for simplicity. In general the frequency f in a range [f_{min}, f_{max}] corresponds to a range of wavelengths [λ_{min}, λ_{max}]. For example a frequency range of [20kHz, 500kHz] corresponds to a range of wave-lengths from 0.7mm to 17mm in reality. Obviously, we can choose the ranges freely to suit different applications.

2.2.4.1 Bat Motion

For the bats in simulations, we have to define the rules how their positions x_i and velocities v_i in a d-dimensional search space are updated. The new solutions $\boldsymbol{x}_i^t$ and velocities $\boldsymbol{v}_i^t$ at time step t are given by

$$f_i = f_{\min} + (f_{\max} - f_{\min})\beta, \qquad (1)$$

$$\boldsymbol{v}_i^{t+1} = \boldsymbol{v}_i^t + (\boldsymbol{x}_i^t - \boldsymbol{x}_*)f_i, \qquad (2)$$

$$\boldsymbol{x}_i^{t+1} = \boldsymbol{x}_i^t + \boldsymbol{v}_i^t, \qquad (3)$$

where β ∈ [0, 1] is a random vector drawn from a uniform distribution. Here x_* is the current global best location (solution) which is located after comparing all the solutions among all the n bats at each iteration t. As the product $\lambda_i f_i$ is the velocity increment, we can use f_i (or λ_i) to adjust the velocity change while fixing the other factor λi (or fi), depending on the type of the problem of interest. In our implementation, we will use $f_{min} = 0$ and $f_{max} = O(1)$, depending on the domain size of the problem of interest. Initially, each bat is randomly assigned a frequency which is drawn uniformly from $[f_{min}, f_{max}]$.

For the local search part, once a solution is selected among the current best solutions, a new solution for each bat is generated locally using random walk

$$\boldsymbol{x}_{\text{new}} = \boldsymbol{x}_{\text{old}} + \boldsymbol{\epsilon}\, A^t,$$

where ϵ is a random number vector drawn from [−1, 1], while $A^t = <A_i^t>$ is the average loudness of all the bats at this time step.

The update of the velocities and positions of bats have some similarity to the procedure in the standard particle swarm optimization, as f_i essentially controls the pace and range of the movement of the swarming particles. To a degree, BA can be considered as a balanced combination of the standard particle swarm optimization and the intensive local search controlled by the loudness and pulse rate.

Loudness and Pulse Emission

Furthermore, the loudness A_i and the rate r_i of pulse emission have to be updated accordingly as the iterations proceed. As the loudness usually decreases once a bat has found its prey, while the rate of pulse emission increases, the loudness can be chosen as any value of convenience. For example, we can use $A_0 = 100$ and $A_{min} = 1$. For simplicity, we can also use $A_0 = 1$ and $A_{min} = 0$, assuming $A_{min} = 0$ means that a bat has just found the prey and temporarily stop emitting any sound. Now we have,

$$A_i^{t+1} = \alpha A_i^t, \quad r_i^t = r_i^0[1 - \exp(-\gamma t)],$$

where α and γ are constants. In fact, α is similar to the cooling factor of a cooling schedule in the simulated annealing [107]. For any $0 < \alpha < 1$ and $\gamma > 0$, we have

$$A_i^t \to 0, \quad r_i^t \to r_i^0, \text{ as } t \to \infty$$

In the simplest case, we can use $\alpha = \gamma$, and we have used $\alpha = \gamma = 0.9$ in our simulations.

Preliminary studies by [58] suggested that bat algorithm is very promising for solving nonlinear global optimization problems. Now we extend it to solve multi-objective optimization problems.

Multi-objective Bat Algorithm

Multi-objective optimization problems are more complicated than single objective optimization, and we have to find and/or approximate the optimality fronts. In addition, algorithms have to be modified to accommodate multi-objectives properly.

2.3 Literature Review

2.3.1 Exact Algorithms

Exact algorithms for the FLP represent those algorithms developed to obtain, in theory, an optimal solution to the facility layout problem. The major advantage of an exact algorithm is that it considers the whole solution space and the optimality of the final layout solution can be guaranteed. Unfortunately, these models are not necessarily of practical value. This is because they can only consider very small sized problems (less than 10 unequal sized departments), which are far from the size of common industry-practical problems (30–40 departments). When the size of the problem increases, the algorithms become impossible to solve in a practical sense because of the computational complexity of the FLP.

The well-known exact algorithms for the FLP include the quadratic assignment problem (QAP) model and the mixed integer programming (MIP) model. The QAP [4], as a special case of distance-based FLP with discrete representation, assumes that every department has equal area and that all locations (grids) are fixed and known a priori. The QAP formulation assigns every department to one location and at most one department to each location, which means a one-to-one matching between departments and locations. The cost of placing a department at a particular location is dependent on the location of the interacting departments. Although the QAP formulation greatly simplifies the FLP and cannot describe the reality of the industrial applications, the QAP is still one of the most challenging optimization problems—recently, a 30-facility QAP required 1000 computers in a massive parallelization effort over a seven-day period that lead to an equivalent 6.9 years of computational effort. The size of the QAP that can be solved in a reasonable computational effort is around 20 departments [5].

A MIP-based formulation for the facility layout problem (MIP-FLP) was presented by Montreuil [6]. He formulates the FLP as a 0-1 mixed integer programming model with a distance-based objective function. Because the MIP-FLP model utilizes a continuous representation, it is more accurate and realistic than the traditional QAP model [6,7].

The MIP-FLP has become one of the main focus areas in FLP research in recent years. However, since the MIP-FLP is very difficult to solve to optimality (less than 10 departments), an efficient heuristic that is based on the MIP-FLP needs to be developed.

The QAP was the first exact approach in FLP research. The QAP was first proposed by Koopmans and Beckman in 1957 [4], which was introduced to model interacting plants of equal areas. A typical QAP model is given as follows:

$$\min \quad \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} c_{ijkl}x_{ik}x_{jl}$$

$$\begin{aligned} \text{s.t.} \quad \sum_{i=1}^{n} x_{ik} &= 1, \quad k = 1,\ldots,n \\ \sum_{j=1}^{n} x_{jl} &= 1, \quad l = 1,\ldots,n \end{aligned}$$

$$x_{ik} = \begin{cases} 1 & \text{if department } x \text{ is assigned to location } k, \\ 0 & \text{otherwise} \end{cases} \qquad i,k = 1,\ldots,n$$

,

Where c_{ijkl} is the cost incurred by assigning department i in location k and department j in location l. The binary decision variable, x_{ik}, is equal to 1 if department i is assigned to location k and 0 otherwise.

As we discussed in Chapter 1, the QAP assumptions include equal-area departments and fixed and known locations to place the departments. Utilizing a discrete representation, the QAP formulation takes fixed locations as “giant grids” and assigns every department to exactly one grid (see (2.2) and (2.3)). The cost of such a one-to-one assignment between departments and grid locations depends on the location of the interacting departments (see (2.1)).

An alternative formulation of the QAP considers assigning interdepartmental distances to department pairs [14]. The QAP has been proven to be NP-complete [15]. Optimal solutions for the QAP model in general cases can only be found for problems with less than 18 departments [16].

Some modified QAP models [17] were presented to solve the unequal-area FLP by breaking departments into small grids with equal area, assigning large artificial flows between those grids of the same department to ensure that the departments are not split, and solving the resulting QAP. Such approaches actually increase the discrete representation resolution (i.e., smaller grids underlying the facility), allowing each department to be assigned to more than one grid. However, due to the increase in the number of "departments," it is not possible to solve even small-sized problems with a few unequal-area departments. Moreover, it is shown in [9] that such an approach is not effective because it implicitly adds a department shape constraint. Such a constraint limits the solution space in a manner that cannot be known beforehand.

Some researchers have used QAP in a modified form to solve a specific facility layout problem. For example, [18] developed a QAP model to address the bidirectional circular layout problem (Bi-CLP), where the departments are arranged along a simple closed-loop aisle and the flow between departments can occur either in the clockwise or counterclockwise direction based on whichever is shorter along the aisle.

Another discrete representation based exact algorithm is the MIP-FLP model represented by [19] to solve the process plant layout problem in the chemical industry. This model is different from the QAP in the following ways: (1) considering the equipment size and orientation; (2) considering both the two-dimension and the three-dimension layout solutions; and (3) developing a multi-objective function that includes not only material handling cost, but also land, piping and floor construction costs. However, this model still suffers from the discrete representation weaknesses discussed in Section 1.3.2.

2.3.2 Graph Theoretic Approaches

Graph theoretic approaches assume that the closeness ratings between departments are known a priori. Each department is represented as a node and adjacency relationships between departments are represented by an arc connecting the two adjacent nodes (departments) in the adjacency graph [8]. There are no underlying grids in such a representation, so usually graph theoretic approaches are considered as continuous representation based approaches.

The optimization objective used in graph theoretic approaches is the closeness rating function in (1.1). This objective function is first translated to obtain a maximal weighted planar graph (MWPG). Secondly, the MWPG is transformed into a dual graph. Finally, a block layout is generated through the dual graph. Giffin [20] showed that MWPG is a NP-complete problem. Like the QAP approach, even small-sized problems cannot be solved to optimality. As a result, many construction heuristics based on graph theoretic models are developed. Some of them are reviewed in the following section. A thorough review of such heuristics can be found in [21].

2.3.3 Mixed Integer Programming Approaches

A MIP formulation for the FLP was originally presented in 1990 by Montreuil [6]. This model uses a distance-based objective, but is not based on the traditional QAP framework.

Instead, it utilizes a continuous representation of a layout and considers departments with unequal areas. In this model, the locations of, and dimensions of, departments are decision variables. A number of binary integer variables are used to avoid overlapping departments.

1A graph is planar if it can be drawn in the plane and each arc intersects no other arcs and passes through no other nodes. A planar sub-graph of an adjacency graph is called a maximal planar graph if no arcs can be added without destroying the planarity of the graph.

This model is commonly referred to as FLP0. One of the problems in FLP0 is that in lieu of the exact nonlinear (specifically non-convex and hyperbolic) area constraint, a bounded perimeter constraint is used to linearize the model. However, using a bounded perimeter constraint instead of an exact area constraint can lead to errors in the final area of each department. For the maximum aspect ratio of departments equal to 2, 3, 4, and 5, the boundary perimeter constraint used in FLP0 is satisfied even if the final area of a department is less than its actual required department by 11%, 25%, 36% and 44%, respectively.

A modified MIP-FLP model based on FLP0 was presented in 1999 by Meller, Narayanan and Vance [7] to improve the model accuracy and approach efficiency. This model is commonly referred to as FLP1. The bounded perimeter constraint in FLP1 is modified, which results in final department areas that are no less than their actual area requirements by 2.5%, 2.5%, 6.3% and 14.3% for an aspect ratio equal to 2, 3, 4, and 5, respectively. More importantly, this modified MIP-FLP model also adds some valid inequalities in order to eliminate some infeasible solutions from the solution space and to improve the algorithm's efficiency.

Numerical results from that literature show that FLP1 is more accurate and effective than FLP0 in terms of solution quality and computational efforts.

This MIP-based model has advantages over the QAP and graphic theoretic approaches, especially in terms of department shapes and problem representation. However, because of the added complications of unequal areas, varying department horizontal and vertical dimensions, and overlapping prohibition constraints, it is extremely difficult to solve such MIP-based models to optimality. The literature shows that for FLP0 it can only solve very small sized problems ($n \approx 5$). For FLP1, even though the authors introduced a number of valid inequalities to the model, the increased problem size that can be solved ($n \approx 7$) is still far from the size of common industry practical problems (30–40 departments).

The aspect ratio of a department is the ratio of its longest side length over its shortest side length.

In order to further improve the performance of the MIP-FLP model and algorithm, a series of enhancements were presented by [22]. Those new enhancements are based on FLP1, including a novel polyhedral outer approximation scheme for the nonlinear area constraints, symmetry-avoiding valid inequalities, several surrogate constraints and inequalities to prevent the department overlapping, and a well-designed branching variable selection priority scheme. The computation results from [22] show that he efforts and accuracy of final solutions are increased ($n \approx 9$) and some difficult test cases are solved for the first time in the literature. However, the problem size is still limited and not applicable for most industrial applications.

One of the major difficulties that arises in solving the MIP-FLP is from the disjunctive constraints and the large quantity of binary integer variables that prevent departmental overlap. Hence, many researchers [23] & [6] have attempted to solve such MIP-FLP models by heuristically fixing a subset of those binary integer variables and then solving the resulting simplified model. Some of the literature in studying the heuristics for the MIP-FLP model is reviewed in the Section 2.2.2.

2.3.4 Heuristic Approaches

Because of the computational difficulty in solving the QAP, graph theory models, or the MIP-FLP to optimality, a great deal of research has centered on finding “good” solutions by implementing heuristic approaches. There are two types of heuristics: construction heuristics and improvement heuristics. Construction-type heuristics build a single solution from scratch (typically in an open space) by successively selecting and locating a new department until the layout is completed. Alternatively, improvement-type heuristics require an initial layout as input, and the algorithm improves the initial layout by making use of some improvement mechanism, such as pair-wise or multi-pair-wise exchanges, until no further improvements can be found. Many improvement routines have been applied (e.g., steepest descent, simulated annealing, genetic algorithms, etc.) to improvement-type heuristics.

In addition to the above classification on the basis of search mechanism, we also classify and review the heuristic literature according to the layout solution representation (discrete or continuous). We do so because the focus of this chapter is the layout representation used in previous research instead of search strategy.

2.3.4.1 Heuristics with Discrete Representation

Montreuil, Ratliff, and Goetschalckx [24] presented an interactive construction-type heuristic, MATCH, which utilizes a discrete representation and integer programming to solve a b-matching model. A b-matching problem is to find a maximum weighted matching in an edge-weighted graph that each edge has its lower and upper bounds to restrict the number of times the edge can be used and each vertex has a integer parameter to specify the number of the vertex must be matched with all other vertexes. Their approach tries to find a matching that maximizes the adjacency score while satisfying the constraints for number of matches in the adjacency graph.

SHAPE [25] is a construction-type heuristic based on a discrete representation and distance-based objective. The department entry sequence is determined by each department's flows and a user-defined critical flow value. The first department is placed at the center of the layout. Subsequent departments are placed based on the objective function value increase if placed on each of the four sides of current layout.

CRAFT [26] is one of the first improvement-type heuristics. CRAFT searches for the improvements by implementing two-way or three-way exchanges of the centroids of non-fixed departments. Due to the primitive exchange routine, only departments that are either of the same size or adjacent in the current layout may be exchanged.

The space-filling curve representation is another example of a discrete representation. A space-filling curve is a curve visiting the underlying grids contiguously to avoid the presence of split departments. MULTIPLE [3] and SABLE [9] are two algorithms based on the space-filling curve representation. MULTIPLE utilizes a two-way exchange to improve the initial layout. SABLE applies a simulated annealing (SA) algorithm to search for "good" layout solutions. Both algorithms are capable of solving single-floor and multiple-floor layout problems.

Some researchers combine meta-heuristics with CRAFT to provide randomness mechanism to allow CRAFT to explore additional two-way local optimal solutions.

One of the newest research results is presented by [27], named Meta-RaPS CRAFT, which is based on a discrete representation and a distance-based objective. Meta-RaPS is a strategy used to change the priority rules based on the insertion of a random element. In Meta-RaPS CRAFT, the decision of department exchange is based on the priority rule, which is determined by Meta-RaPS under a random mechanism.

2.3.4.2 Heuristics with Continuous Representation

The delta-hadron approach (DA) [28] is one of the most widely cited construction-type heuristics. As a graph-based approach, DA uses the adjacency-based objective and generates a layout by determining the entry sequence of nodes (departments) into the graph. At each stage, a node (department) enters the graph to maximize the adjacency benefits with the other nodes (departments) in the graph. A great deal of research has been conducted to improve DA's performance [29,30,31]. Another construction-type heuristic based on graph theoretic approaches and the adjacency-based objective is SPIRAL [32], which utilizes the concept of "relationship tuples" to construct an adjacency graph.

LOGIC [33] is an improvement-type heuristic based on a collection of rectangular partitions called a slicing tree. Based on the slicing tree structure, the given facility is recursively partitioned. LOGIC can consider fixed and non-fixed departments.

NLT [34] is an approach based on nonlinear programming and the distance-based objective. NLT utilizes a continuous representation and solves the constrained nonlinear programming model by transforming the model into an unconstrained form by an exterior point quadratic penalty function method. The resulting department shapes are all rectangular.

Some heuristics have been developed to improve the performance of Montreuil's MIP-FLP model. [23] and [35] applied qualitative layout anomalies (QLAs) and design skeletons to Montreuil's MIP-FLP model. The heuristics utilize context-based information to reduce the solution tree. Lacksonen [36] proposed an approach that combines the QAP model with Montreuil's MIP-FLP model. First, a QAP model is solved by applying a cutting plane heuristic. The result of the QAP is used as an input of approximate location information of departments, which is used to reduce the number of binary variables in the MIP model.

Langevin et el [37] proposed a heuristic approach based on Montreuil’s MIP-FLP model to solve the spine layout problem, where a main aisle is used for material handling and all departments are located along the both sides of the aisle. This approach first generates an ordered list of departments based on a heuristic proposed by Heragu and Kusiak [38] to solve a single row layout problem. Then, it applies the ordered list to the Montreuil’s MIP-FLP model to fix the binary variables and transforms Montreuil’s model from an MIP model to a linear programming model. The maximum size of test problems presented in [37] is 22 departments. This approach uses a heuristically-fixed ordered list as initial input and cannot consider all the possible solutions. It is also specifically designed for the spine layout problem. As such, it is not suitable for the general FLP.

Lacksonen [36] proposed a pre-processing heuristic to fix a subset of the total binary variables according to a regression formula based on the area of each department and material flows associated with each department. The maximum size of test problems is 12 departments.

Montreuil et al. [39] presented an Ant Zone meta-heuristic based on a continuous representation, where an ant colony approach is used to generate the layout code, and given a layout code, a zone-based linear programming model is solved to optimize the zone-based layout solution.

Another type of continuous-representation-based heuristic design is focused on studying the FLP with fixed-shaped departments and fixed input/output locations, where the locations of the departments are represented continuously. One of the most recent research is presented by Kim and Kim [40], where an MIP model is formulated and a construction-improvement heuristic is presented based on the MIP model to minimize the distance-based objective function for the FLP with pre-specified-shaped departments and fixed input/output locations. However, the department shapes are restricted to rectangular-shaped only.

Irohara and Yamada [41] present a location matrix based heuristic to solve the FLP with aisle structure where there are three alternatives for the input/output locations. One main limitation for this research is that the approach assigns departments within a zone in a sequential-order along either the horizontal direction or vertical direction. Therefore, it cannot consider all-possible layout solutions.

2.3.5 Simulated annealing-based method

SA is a stochastic search process based on the concept of 'annealing' [42]. The annealing of a solid material is two-phase method. In the first phase, solid material is heated up to a certain temperature where its atoms can move freely or randomly in material. In second phase, this hot material is allowed to cool slowly so that the atoms can rearrange themselves into a lower energy state to form crystal. This second phase is also known as crystallization process. Since the crystalline state is the minimum energy state of the system, this process can be thought of minimization of free energy of the system or solid. It is found that improper heating and/or fast cooling can lead solid to an amorphous state with higher energy level. This situation is an analogy of reaching a local minimum instead of global minimum of the system energy, and hence proper schedule for annealing is vital.

There are several important analogies found in literature between annealing and combinatorial optimization. The system analogies are:

- The system state at any point of time, that is, at a system energy level is analogous to particular solution of the optimization problem the free energy of the system is analogous to the decision variables of the objective function the slight perturbation imposed on the system to change state analogous to a movement into a neighborhood solution with respect to the local search the cooling schedule corresponds to the control or iteration mechanism for the search algorithm
- The crystalline state of the system analogous to the final solution generated by the algorithm (single solution).

The first two analogies stated here-in-before are subjected to minimization.

In case of single objective optimization problem SA is guaranteed to converge in asymptotic time, though the computational time grows exponentially with respect to the size of the problem [42]. The disadvantage of SA is that it requires multiple runs to defining proper cooling schedule in order to get optimum solution.

It seems that SA as a method is being preferred by the researchers in the field, staring form manufacturing cell design to multi-objective optimization of dynamic and static behavior of FLP irrespective of whether it is an equal or an unequal sized facilities [43,44,45,46,47,48]. Concept of Pareto front generation, application of non-dominated solution techniques are getting powered by SA-based algorithms [49]. Hybrid method of SA with TS [50], SA and GA [51] helps to avoid high computational cost [52] and improves the solution, both qualitatively as well as quantitatively.

Table 1: Survey of SA-based FLP literature

			Objectives							
Sl. No.	*References*	*Modelling technique*	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*
1	Castillo and Peters (2003)	NLMIP						√		
2	Dong et al. (2009)	MIP		√						
3	Ioannou (2007)	IP				√	√			
4	Matsuzaki et al. (1999)	QAP and slicing tree						√	√	
5	McKendall et al. (2006)	Modified QAP		√				√		
6	Mir and Imam (2001)	QAP						√		
7	Şahin and Türkbey (2009)	MIP	√							
8	Singh and Sharma (2007)	QAP						√		
9	Sugiyono (2006)	Bond energy algorithm			√					
10	Şahin et al. (2010)	QAP		√				√		
11	Yang et al. (2005)	MIP								√

A survey of SA-based facility layout literature, done for this research, is tabulated in Table 1, wherein eight major objectives selected for SA-based solution are tabulated in columns A to G and detailed out as follows:

A minimization of Material handling and total closeness rating score

B minimizing re-layout cost

C total area optimization and stochastic level of production

D vehicle routing

E optimization of fixed and variable cost of production

F minimization of material handling cost

G optimize number and location of elevators

H material flow path design.

2.3.6 Tabu Search-based Method

TS is an iterative meta-heuristic method where at each iteration, current solution moves to the neighborhood point comprising of smallest value with respect to the objective function. One of the main components of TS is its adaptive memory, which creates more flexible search behavior for responsive exploration.

Incorporation of adaptive memory in TS builds its capability that the solution is not stuck into local optima. There are four major building blocks of tabu memory structure, which is referred to as, recency, frequency, quality and influence. Recency and frequency-based memories are complementary in nature, and a combined use helps in recognizing the replica solutions. Quality-based memory helps to judge the goodness of the solution and short moves are taken in neighborhood. Influence-based memory takes care of system learning during the search process.

Intensification and diversification strategies are also highly important components of TS. Intensification is realized by storing in its memory the historically found good or 'elite' solution and the corresponding search move combinations. It helps to concentrate search around the good solution region, and also helps the search process to escape from bad region. Diversification strategy helps to find out new region of solutions, which were not explored during the search and it also helps to get away from sticking to local optimal solutions.

To facilitate these strategies, TS uses a short-term, intermediate-term and long-term memory. Short-term memory stores the variable values of the recently visited points and marks them as tabu, meaning forbidden, to avoid cycling within local region.

Intermediate-term memory stores optimal or near optimal solution to help the intensification operation, while the long-term memory keeps track of the under-explored regions as well as the regions which were already explored exhaustively thereby helping the diversification.

In case of FLP the TS has been widely used in optimizing material handling cost, utilization of space, minimization of re-layout cost for both single- and multi-objective problems [45]. TS is also deployed for optimizing single row FLD problem (a special class of FLP). But difficulties have been encountered when TS was deployed for continuous search space due to the approximation introduced on account of digitization of continuous space [42]. SA is to overcome computational complexities and difficulties of TS proposed a special different intensification and diversification strategies, which shows better convergence of searching to get proper arrangement of facilities. Improved TS algorithm with intensification, reconstruction and solution acceptance operation was proposed by Singh [53], which gives comparative result with respect to some benchmark problems found in QAP-based FLP literature.

Table 2: Survey of TS-based FLP literature

		Objectives			
References	*Modelling technique*	*A*	*B*	*C*	*D*
Abdinnour-Helm and Hadley (2000)	Graph partitioning	√			
Chiang (2001)	QAP	√			
McKendall and Hakobyan (2010)	MILP	√		√	
Scholz et al. (2009)	QAP	√	√		
Seo et al. (2006)	QAP	√			
Sugiyono (2006)	Bond energy algorithm	√			√
Yang et al. (1999)	UMNFP	√			

The survey findings of TS-based facility layout literature undertaken in this work is tabulated in Table 2, wherein four major objectives selected for TS-based solution are tabulated in columns A to D, which is as follows:

A minimizing total material handling or flow cost

B minimizes size of resulting layout or maximizes utilization of area

C minimizing re-layout cost

D total area optimisation with stochastic level of production.

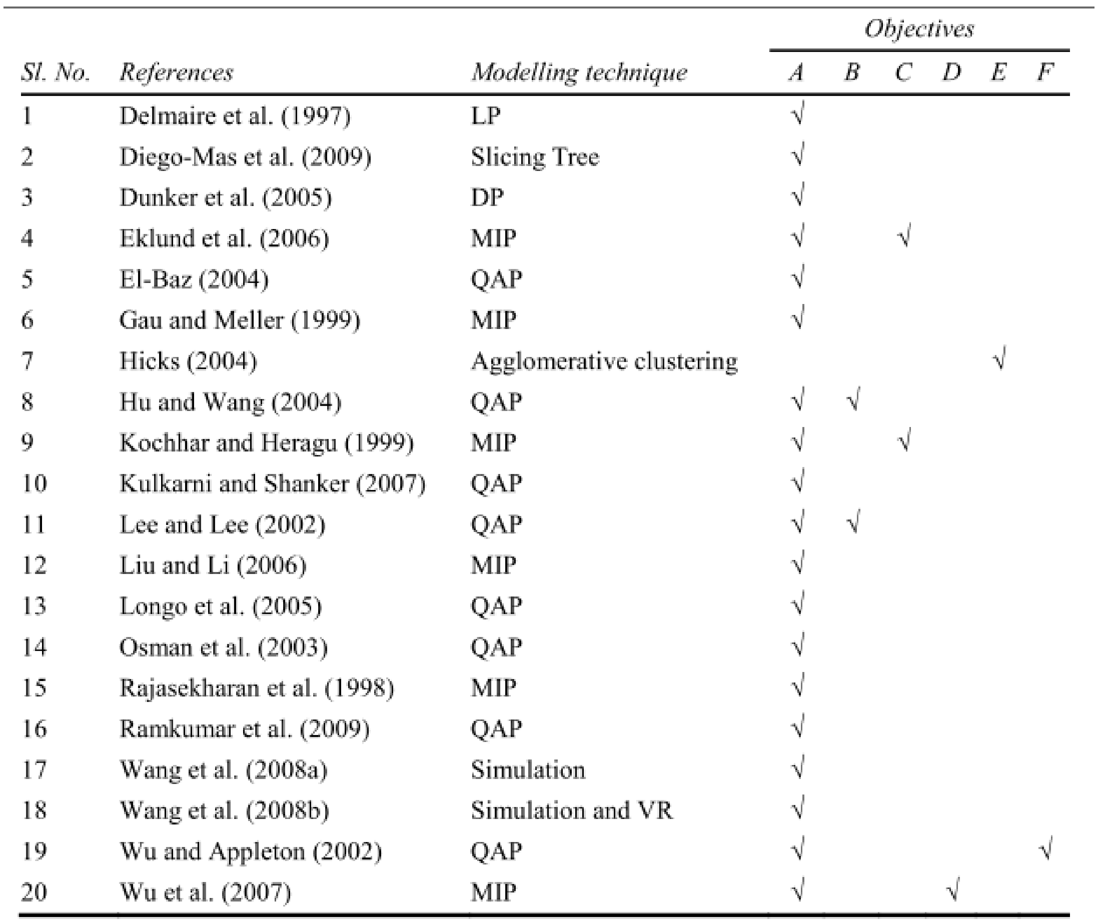
2.3.7 Genetic algorithm-based method

Search mechanism in GA actually is based on the mechanism of natural selection and natural genetics. GA is widely used in optimization problem due to its robustness and is a tool in industrial engineering optimization problems deployed in the recent past. Building block of GA consists of five major elements:

1 a genetic representation of solution
2 a well-defined mechanism to generate initial population
3 a fitness function to evaluate solution quality
4 genetic operators, namely crossover and mutation, analogous to biological operation to generate offspring
5 Parameter values.

An encoded representation of problem parameters, used as chromosome, is generally found in the form of string of binary or real numbers. Each variable is analogous to the gene of biological chromosome, and such gene values are decoded to yield solutions to the problem. Reproduction or selection operator replicates good solutions and eliminates bad solutions from the population, while keeping the population size unchanged. Roulette wheel and tournament selection process are well established for this operation, however tournament selection shows better performance in comparison to other selection operators. Crossover operator is used to generate offspring from parent chromosome by means of interchanging substring(s). No such restriction posed on the exact procedure to crossover in GA; rather they are problem and domain specific. On the other hand, mutation operator is used to change a particular allele value by means of replacing this with its complement in case of binary-coded GA. Flexibility is maintained for the mutation rules for providing robustness to the algorithm. From the literature review, undertaken in the present work, it is observed that GA has rather frequently been used in the recent period as an optimization tool for FLP. As indicated by application of GA was found to be of significant proportion in optimizing QAP formulation for FLP. It is observed during the present study undertaken that in most cases the minimization of material handling cost is considered as an objective function. However, relatively fewer reports addressed flexibility of layout with aisle structure, inter- and intra-cell material handling in cellular layout, maximum utilization of space and minimization of total travelling distance. Facility dispositions and its geometrical orientations are generally encoded as slicing tree, ordered set of facilities, facility number order, x–y coordinate maps to generate initial population. The difficulties of applying classical crossover, mutation operators on FLP were addressed. Different problem specific modified crossover and mutation operation were developed to achieve better efficiency of GA. The operators have also been modified to preserve elite solutions tested response of different GA crossover, mutation rate, population size and maximum number of generation on their problem. Parallel implementation of SA and GA, where GA used for global search and SA applied for local search, is also proposed by developed GA-based software for layout design with an easy GUI.

Table 3: Survey of GA-based FLP literature



			Objectives					
Sl. No.	*References*	*Modelling technique*	*A*	*B*	*C*	*D*	*E*	*F*
1	Delmaire et al. (1997)	LP	√					
2	Diego-Mas et al. (2009)	Slicing Tree	√					
3	Dunker et al. (2005)	DP	√					
4	Eklund et al. (2006)	MIP	√		√			
5	El-Baz (2004)	QAP	√					
6	Gau and Meller (1999)	MIP	√					
7	Hicks (2004)	Agglomerative clustering					√	
8	Hu and Wang (2004)	QAP	√	√				
9	Kochhar and Heragu (1999)	MIP	√		√			
10	Kulkarni and Shanker (2007)	QAP	√					
11	Lee and Lee (2002)	QAP	√	√				
12	Liu and Li (2006)	MIP	√					
13	Longo et al. (2005)	QAP	√					
14	Osman et al. (2003)	QAP	√					
15	Rajasekharan et al. (1998)	MIP	√					
16	Ramkumar et al. (2009)	QAP	√					
17	Wang et al. (2008a)	Simulation	√					
18	Wang et al. (2008b)	Simulation and VR	√					
19	Wu and Appleton (2002)	QAP	√					√
20	Wu et al. (2007)	MIP	√			√		

Ease to hybrid GA with other algorithms and its ability of constraint handling by penalization, as a property has made GA a preferred tool for multi-objective optimization for FLP. Generally, hybrid GA with SA or TS method chosen as algorithm to optimize multiple objectives, such as material handling cost and space utilization, placing facilities around an aisle structure and optimization of material handling cost, minimization of material handling cost and re-layout cost, loop layout in flexible manufacturing system. GA in cellular manufacturing has been applied in cell formation and group layout, involving optimization of inter and intra-cell material handling effort.

GA-based solutions are extended by VR [54] or AutoCAD representation to facilitate visualization of quality of solution.

The survey findings of GA-based facility layout literature, undertaken in this work is tabulated in Table 3, wherein six major objectives selected for GA-based solution are tabulated in columns A to F, which is as follows:

A	minimizing total material handling or flow cost
B	minimizes size of resulting layout or maximizes utilization of area
C	generation of flexible layout
D	minimize material handling cost for inter-cell and intra-cell movement
E	minimizing total rectilinear distance travel for material
F	optimization of aisle structure.

2.3.8 Nature Inspired Algorithms

Real-world optimization problems are often very challenging to solve, and many applications have to deal with NP-hard problems. To solve such problems, optimization tools have to be used, though there is no guarantee that the optimal solution can be obtained. In fact, for NP-problems, there are no efficient algorithms at all. As a result, many problems have to be solved by trial and errors using various optimization techniques. In addition, new algorithms have been developed to see if they can cope with these challenging optimization problems.

Among these new algorithms, many algorithms such as particle swarm optimization, cuckoo search and firefly algorithm, have gained popularity due to their high efficiency. In the current literature, there are about 40 different algorithms. It is really a challenging task to classify these algorithms systematically. Obviously, the classifications can largely depend on the criteria, and there is no easy guideline to set out the criteria in the literature.

Sources of Inspiration

Nature has inspired many researchers in many ways and thus is a rich source of inspiration. Nowadays, most new algorithms are nature-inspired, because they have been developed by drawing inspiration from nature. Even with the emphasis on the source of inspiration, we can still have different levels of classifications, depending on how details and how many sub sources we will wish to use. For simplicity, we will use the highest level sources such as biology, physics or chemistry.

In the most generic term, the main source of inspiration is Nature. Therefore, almost all new algorithms can be referred to as nature-inspired. By far the majority of nature-inspired algorithms are based on some successful characteristics of biological system. Therefore, the largest fractions of nature-inspired algorithms are biology-inspired, or bio-inspired for short.

Among bio-inspired algorithms, a special class of algorithms has been developed by drawing inspiration from swarm intelligence. Therefore, some of the bio inspired algorithms can be called swarm-intelligence-based. In fact, algorithms based on swarm intelligence are among the most popular. Good examples are ant colony optimization, particle swarm optimization, cuckoo search, bat algorithm, and firefly algorithm.

Obviously, not all algorithms were based on biological systems. Many algorithms have been developed by using inspiration from physical and chemical systems. Some may even be based on music [59]. In the rest of thesis, we will briefly divide all algorithms into different categories, and we do not claim that this categorization is unique. This is a good attempt to provide sufficiently detailed references.

Classification of Algorithms

It is worth pointing out the classifications here are not unique as some algorithms can be classified into different categories at the same time. Loosely speaking, classifications depend largely on what the focus or emphasis and the perspective may be. For example, if the focus and perspective are about the trajectory of the search path, algorithms can be classified as trajectory-based and population-based. Simulated annealing is a good example of trajectory-based algorithms, while particle swarm optimization and firefly algorithms are population-based algorithms. If our emphasis is placed on the interaction of the multiple agents, algorithms can be classified as attraction-based or non-attraction-based.

2.3.8.1 Firefly algorithms

Firefly algorithm (FA) is a good example of attraction-based algorithms because FA uses the attraction of light and attractiveness of fireflies, while genetic algorithms are non-attraction-based since there is no explicit attraction used. On the other hand, if the emphasis is placed on the updating equations, algorithms can be divided into rule-based and equation-based. For example, particle swarm optimization and cuckoo search are equation-based algorithms because both use explicit updating equations, while genetic algorithms do not have explicit equations for crossover and mutation. However, in this case, the classifications are not unique. For example, firefly algorithm uses three explicit rules and these three rules can be converted explicitly into a single updating equation which is nonlinear. This clearly shows that classifications depend on the actual perspective and motivations. Therefore, the classifications here are just one possible attempt, though the emphasis is placed on the sources of inspiration.

2.3.8.2 Swarm intelligence based

Swarm intelligence (SI) concerns the collective, emerging behavior of multiple, interacting agents who follow some simple rules. While each agent may be considered as unintelligent, the whole system of multiple agents may show some self-organization behavior and thus can behave like some sort of collective intelligence.
Many algorithms have been developed by drawing inspiration from swarm-intelligence systems in nature.
All SI-based algorithms use multi-agents, inspired by the collective behavior of social insects, like ants, termites, bees, and wasps, as well as from other animal societies like flocks of birds or fish. The classical particle swarm optimization (PSO) uses the swarming behavior of fish and birds, while firefly algorithm (FA) uses the flashing behavior of swarming fireflies.

2.3.8.3 Cuckoo search (CS)

Cuckoo search (CS) is based on the brooding parasitism of some cuckoo species, while bat algorithm uses the echolocation of foraging bats. Ant colony optimization uses the interaction of social insects (e.g., ants), while the class of bee algorithms are all based on the foraging behavior of honey bees.
SI-based algorithms are among the most popular and widely used. There are many reasons for such popularity; one of the reasons is that SI-based algorithms usually sharing information among multiple agents, so that self-organization, co-evolution and learning during iterations may help to provide the high efficiency of most SI-based algorithms. Another reason is that multiple agent can be parallelized easily so that large-scale optimization becomes more practical from the implementation point of view.

2.3.8.4 Bio-inspired, but not SI based

Obviously, SI-based algorithms belong to a wider class of algorithms, called bio-inspired algorithms. In fact, bio-inspired algorithms form a majority of all nature-inspired algorithms. From the set theory point of view, SI-based algorithms are a subset of bio-inspired algorithms, while bio-inspired algorithms are a subset of nature-inspired algorithms. That is SI-based $\subset$ bio-inspired $\subset$ nature-inspired.
Conversely, not all nature-inspired algorithms are bio-inspired, and some are purely physics and chemistry based algorithms as we will see below.
Many bio-inspired algorithms do not use directly the swarming behavior. Therefore, it is better to call them bio-inspired, but not SI-based. For example, genetic algorithms are bio-inspired, but not SI-based. However, it is not easy to classify certain algorithms such as differential evolution (DE). Strictly speaking, DE is not bio-inspired because there is no direct link to any biological behavior. However, as it has some similarity to genetic algorithms and also has a key word `evolution', we tentatively put it in the category of bio-inspired algorithms. For example, the flower algorithm, or flower pollination algorithm, developed by Xin-She Yang in

2012 is a bio-inspired algorithm, but it is not a SI-based algorithm because flower algorithm tries to mimic the pollination characteristics of flowering plants and the associated flower consistency of some pollinating insects.

2.3.8.5 Physics and Chemistry Based

Not all meta-heuristic algorithms are bio-inspired, because their sources of inspiration often come from physics and chemistry. For the algorithms that are not bio-inspired, most have been developed by mimicking certain physical and/or chemical laws, including electrical charges, gravity, river systems, etc. As different natural systems are relevant to this category, we can even subdivide these into many subcategories which is not necessary.
Though physics and chemistry are two different subjects, however, it is not useful to subdivide this subcategory further into physics-based and chemistry. After all, many fundamental laws are the same. So we simply group them as physics and chemistry based algorithms.

2.3.8.6 Other algorithms

When researchers develop new algorithms, some may look for inspiration away from nature. Consequently, some algorithms are not bio-inspired or physics/chemistry-based, it is sometimes difficult to put some algorithms in the above three categories, because these algorithms have been developed by using various characteristics from different sources, such as social, emotional, etc.

2.4 Facility Layout Problem

2.4.1 Facility Layout Problem Objective Functions

In the facility layout problem (FLP) we are to find an efficient non-overlapping planar arrangement of n departments within a given facility. The efficiency of the facility layout is typically measured in terms of material handling costs. In the literature, two common surrogate objectives widely used to approximate material handling costs are given as follows [8]:

2.4.1.1 Closeness Rating Function

A department adjacency-based objective is defined as follows:

$$\max \sum_i \sum_j (r_{ij}) x_{ij}$$

where x_{ij} equals 1 if departments i and j are adjacent, and 0 otherwise. The reward r_{ij} is a numerical value to represent a closeness rating between departments i and j. Such an objective is based on the material handling principle that material handling costs are reduced significantly when two departments are adjacent.

2.4.1.2 Flow Cost Function

An interdepartmental distance-based objective is defined as follows:

$$\min \sum_i \sum_j (f_{ij} c_{ij}) d_{ij}$$

where f_{ij} is the material flow from department i to department j, c_{ij} is the cost to move one unit load one distance unit from department i to department j, and d_{ij} is the distance from department i to department j. This objective is based on the material handling principle that material handling costs increase with the distance the unit load must travel.

There are a variety of ways to measure the distance between a pair of departments (d_{ij}). The following represents commonly-used distance measures for the FLP.

2.4.1.3 Centroid-to-Centroid (CTC) Distance

During the block layout phase where the input/output point and aisle structure are unknown, the distance between two departments is often measured with respect to their centroid locations. The main short-comings of CTC distance include: the mathematically optimal layout may be one with departments represented as concentric rectangles; an algorithm based on CTC attempts to align the department centroids as close as possible, which may make the departments very long and narrow; and L-shaped departments may have a centroid that falls outside of the department [8]. There are some variations to CTC distance measure; e.g., distributed centroid-to-centroid distances (DCTC) and expected distances (EDIST) [9].

2.4.1.4 Contour Distance

Distance may be measured along the aisles between the input/output points of a pair of departments (e.g., see [10,11]). The positive aspect of this measure is that the measured distance is accurate. The major drawback of this accurate measure is that during the block layout design phase one does not know the exact location of input/output points and the aisles, which are to specified during the detailed layout design.

2.4.1.5 Weighted Cost Function

A weighted cost function represents a trade-off between adjacency-based and distance-based objectives. Because there are advantages and disadvantages to adjacency-based and distance-based objectives and the optimal solution under one objective may not be optimal, or even good, under the other objective, some researchers [12] & [13] have combined these two objectives in a weighted criteria approach.

One kind of weighted model is given as follows:

$$\min \alpha \sum_i \sum_j (f_{ij} c_{ij}) d_{ij} - (1-\alpha) \sum_i \sum_j (r_{ij}) x_{ij} \text{ .... (1.3)}$$

where α is a weight with a value between 0 and 1. Such an objective leads to research in the area of the multiple objective facility layout problems.

One of the drawbacks of the above equation is that adjacency-based and distance-based objectives have different scales (mostly since $d_{ij} >> x_{ij}$). Thus, it is difficult to relate the value of the weighing factor α to some physical aspect of the problem. For example, even α = 0.5 does not mean that the adjacency-based and distance-based objectives are weighted equally because they are in different scales. Meller and Gau present a revised objective function to solve this difficulty, which is given as follows:

$$\min \sum_{i}\sum_{j}(f_{ij}c_{ij})d_{ij} + \sum_{i}\sum_{j}(w_{ij}r_{ij})(1 - x_{ij})$$

where the parameter w_{ij} replaces the weighting factor α in (1.3). In order to minimize the impact of setting the weighting factor w_{ij} correctly.

2.4.1.6 Facility Layout Problem Representation

The representation of an FLP solution forms the basis for a mathematical model and greatly impacts the structure and efficiency of the applied optimization algorithms. There are a variety of FLP representation methods, but most of them fall into two main categories: discrete representation and continuous representation.

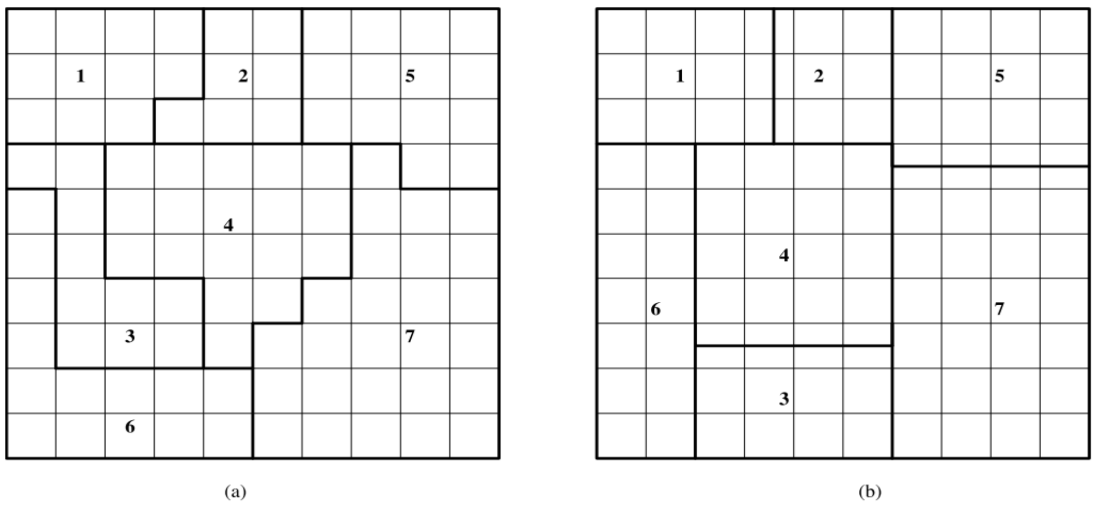


Figure 4: Layout Solutions with (a) Discrete Representation and (b) Continuous Representation.

Discrete Representation: With a discrete representation, the facility is represented by an underlying grid structure with fixed dimensions and all departments are composed of an integer number of grids (see Figure 4(a)). By representing the FLP in a discrete fashion, the FLP is simplified, but at the penalty of eliminating many solutions from consideration.

Of course, the grid size can be chosen sufficiently small such that this penalty is minimized.

However, a smaller grid size (i.e., increasing the resolution) will increase the computational effort as well. Most research on the FLP utilizes a discrete representation.

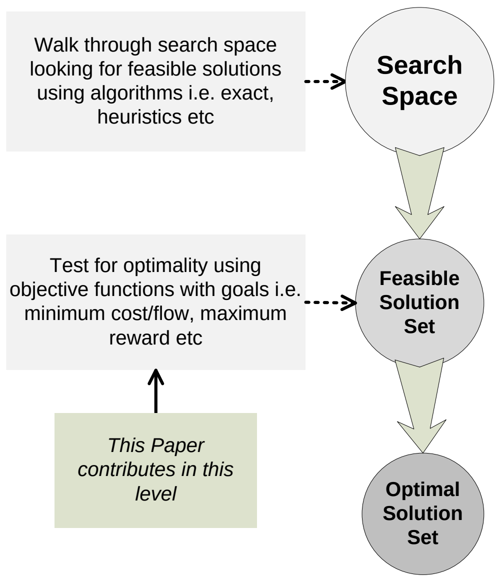
Continuous Representation: In a continuous representation, department dimensions are not restricted to an underlying grid structure, but rather, represented continuously (i.e., department dimensions may take on non-integer values). For example, the discrete layout in Figure 4 (a) could be modeled with a continuous representation as shown in Figure 4 (b).

A continuous representation is more accurate and realistic than a discrete representation, and thus, is capable of finding the “real optimal” final layout solution. However, the continuous representation also increases the complexity of the FLP. As a result, most algorithms based on a continuous representation assume that departments are rectangular in shape. Thus, the “real optimal” layout is restricted as well with most algorithms that utilize a continuous representation.

A mixed-integer programming (MIP) formulation based on a continuous representation was presented by Montreuil [6]. This model uses a distance-based objective with a continuous representation of a layout and considers departments with unequal areas. Both locations and dimensions of departments are decision variables. A number of binary integer variables are used to avoid department overlapping.

CHAPTER 3: SCOPE AND LEVEL OF CONTRIBUTION

In any facility layout problem, the whole algorithm deals with three spaces, namely, search space or problem space, feasible solution space and lastly produce optimal solution set.



Walk through search space looking for feasible solutions using algorithms i.e. exact, heuristics etc
Search Space
Test for optimality using objective functions with goals i.e. minimum cost/flow, maximum reward etc
Feasible Solution Set
This Paper contributes in this level
Optimal Solution Set

Figure 5: Spaces that are dealt with in an FLP Algorithm

After defining the search space or problem space, algorithms are run to find feasible solution space satisfying constraints. For example, exact algorithm or its derivatives tests every possible feasible solution and thus produce a set of feasible solutions. On the other hand, heuristic algorithms abide by some heuristics or rules for taking a probable solution for feasibility checking, and thus works on narrower space compared to exact algorithm. In this manner, the feasible solution set generated from heuristic algorithms are smaller for a particular problem.

After the feasible set is generated, optimality testing is done in order to find the optimum most solution based on well-defined objective functions. Objective functions may be designed in different ways as such focusing on different kinds of objectives i.e. cost of flow, volume of flow, reward value etc. After evaluating feasible solution set, algorithms then provide optimum solution set.

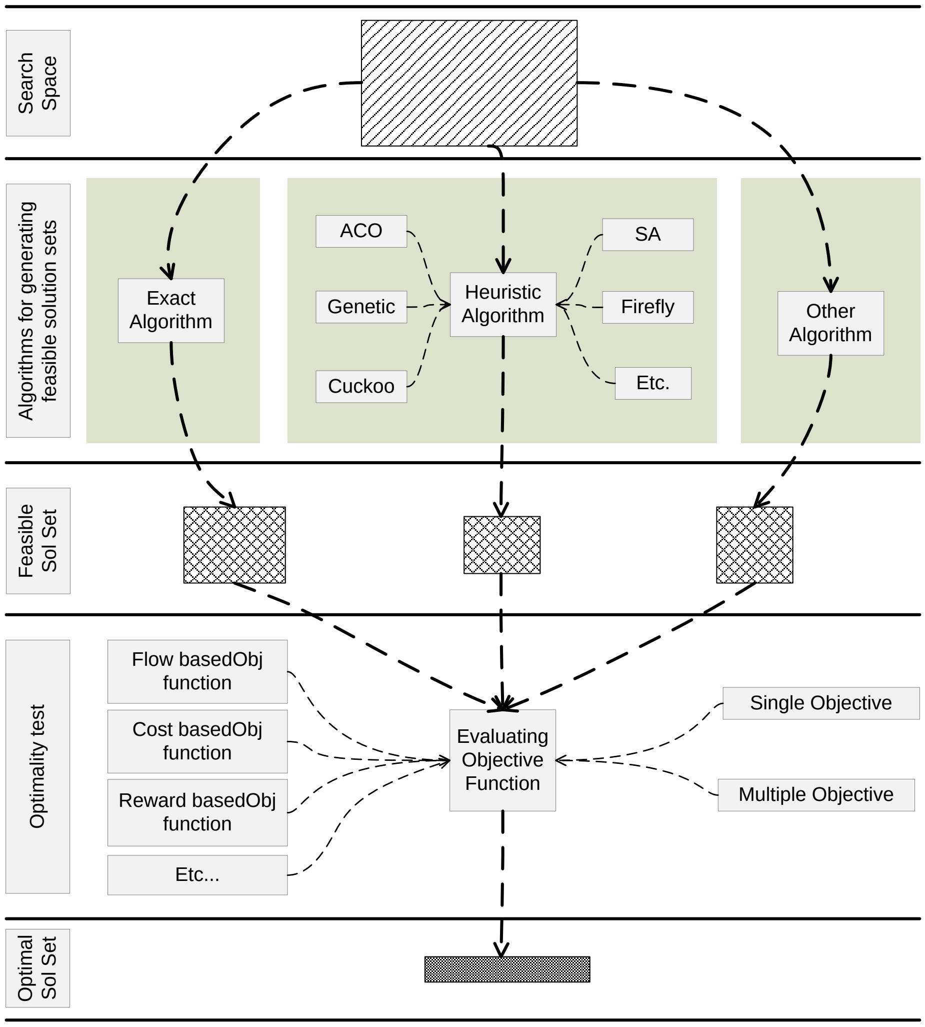


Figure 6: Flow of work of a FLP algorithm

This thesis contributed in the second stage mentioned above, i.e. evaluation stage of feasible solutions' set. More specifically, the problem defined in this thesis is a multi-objective problem, one of which was maximization of reward value taking consideration of adjacency between two departments.

CHAPTER 4: MATHEMATICAL MODEL

4.1 Objective Function

In the facility layout problem (FLP) we are to find an efficient non-overlapping planar arrangement of n departments within a given facility. The efficiency of the facility layout is typically measured in terms of material handling costs. The part of the model that calculates the total cost of flow is the cumulative product of interdepartmental flow, cost of flow and distance between departments. This objective is based on the material handling principle that material handling costs increase with the distance the unit load must travel. A department adjacency-based objective is defined as the product of reward values and adjacency status. Such an objective is based on the material handling principle that material handling costs are reduced significantly when two departments are adjacent.

There are a variety of ways to measure the distance between a pair of departments (d_{ij}). The following represents commonly-used distance measures for the FLP.

A weighted cost function represents a trade-off between adjacency-based and distance-based objectives. Because there are advantages and disadvantages to adjacency-based and distance-based objectives and the optimal solution under one objective may not be optimal, or even good.

So, the actual interpretation of the objective function is:

$$\min \alpha \sum_i \sum_j (f_{ij} c_{ij}) d_{ij} - (1-\alpha) \sum_i \sum_j (r_{ij}) x_{ij}$$

Subject to:

$$d_{ij} = d_{ij}^x + d_{ij}^y ; \forall i, j; i \neq j$$

$$d_{ij}^s = | c_i^s - c_j^s | ; \forall i, j$$

$$l_i^s / 2 \leq c_i^s \leq L^s - l_i^s / 2 ; \forall i, s$$

$$x_{ij} = \begin{cases} 1 & \text{if gap between i \& j is 0} \\ 1/\mathrm{k}_1 \mathrm{e}^{\mathrm{k}_2 gap_{ij}} & \text{other wise} \end{cases}$$

$$gap_{ij}^s = \begin{cases} (c_j - l_j / 2) - (c_i + l_i / 2) & \text{if i preceeds j in s direction} \\ (c_i - l_i / 2) - (c_j + l_j / 2) & \text{if j preceeds i in s direction} \\ 0 & \text{Otherwise} \end{cases}$$

4.2 Parameters and decision variables

α : Weight of objective functions

i, j : Department indices (i, j = 1, 2, …, n)

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

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

c_{ij} : Cost of flow between departments i and j

c^x_i : x co-ordinate of center of department i

c^y_i : y co-ordinate of center of department i

d_{ij} : Rectilinear distance between departments i and j

d^x_{ij} : Rectilinear distance between departments i and j in x axis

d^y_{ij} : Rectilinear distance between departments i and j in y axis

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

l^x_i : length of department i in x axis

l^y_i : length of department i in y axis

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

x_{ij} : binary variable showing whether department i is before department j in the sequence

4.3 Narration of the model

We have implemented exact algorithms for the FLP that represents those algorithms developed to obtain, in theory, an optimal solution to the facility layout problem. The major advantage of an exact algorithm is that it considers the whole solution space and the optimality of the final layout solution can be guaranteed. The limitation of these models is they can only consider very small sized problems (less than 10 unequal sized departments), which are far from the size of common industry-practical problems (30–40 departments). When the size of the problem increases, the algorithms become impossible to solve in a practical sense because of the computational complexity of the FLP. We have opted for this model as because we are not concerned about the feasibility of exact models or comparative analysis of exact models with other models, rather we are interested about the practicality of methodology used for constructing adjacency matrix in adjacency-reward based objective functions.

The objective function is based on a weighted cost function representing a trade-off between adjacency-based and distance-based objectives. Alpha is used for controlling the relative weightage of adjacency-based and distance based objectives.

4.4 Limitation of traditional way of constructing adjacency function

In models that incorporate adjacency matrix, x denoting adjacency is either 1 or 0 (1 for completely adjacent departments and 0 for any gap between departments). This value is then multiplied with closeness ratings provided as input. The problem with this method is, any gap, even if very low, leads the multiplied value (i.e. benefit) to 0 (even in cases with very close proximity).

In practical, the benefit due to adjacency between two departments does not go away for small gaps. So, null value for adjacency does not go with practicality. Rather, we feel that, there might still be benefits of having two departments in very close proximity though benefits may diminish with gap in exponential/non-linear fashion as in depicted in fig(7);

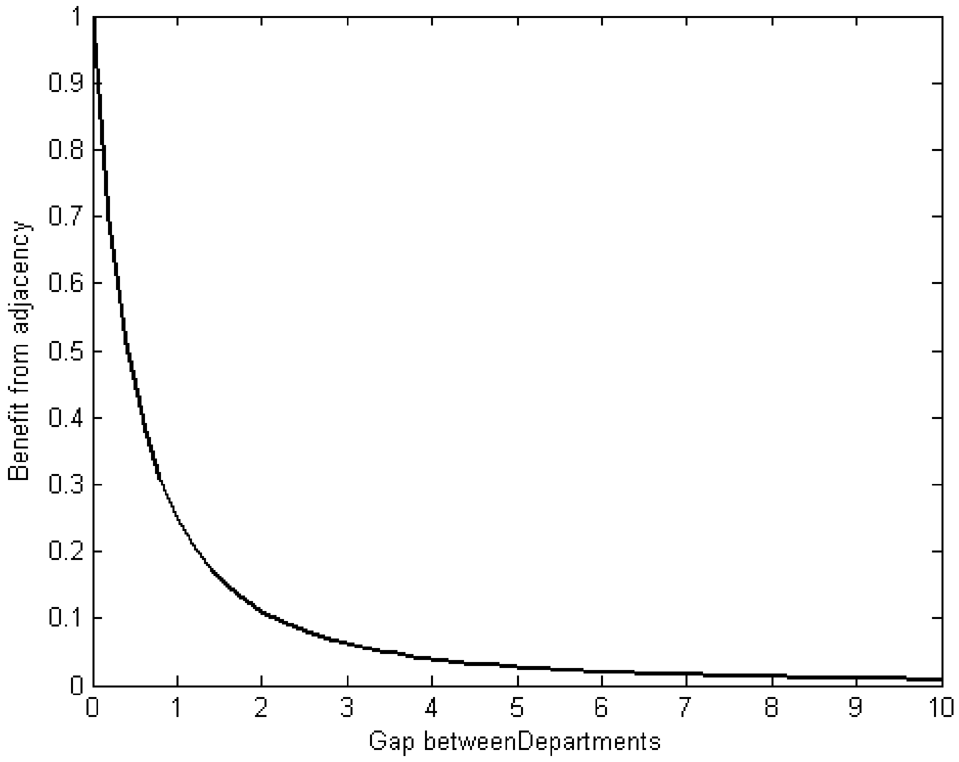


Figure 7: Practical relationship between gaps (between departments) and benefits gained from proximity

(Graph produced by MATLAB)

For this, the thesis suggests rather a nonlinear function for generating values of adjacency matrix.

4.5 Suggested function for constructing adjacency function

According to the proposition that we made in the previous point, adjacency value practically should not be binary one rather should follow non-linear function. Below is provided a outputs from candidate polynomial functions:

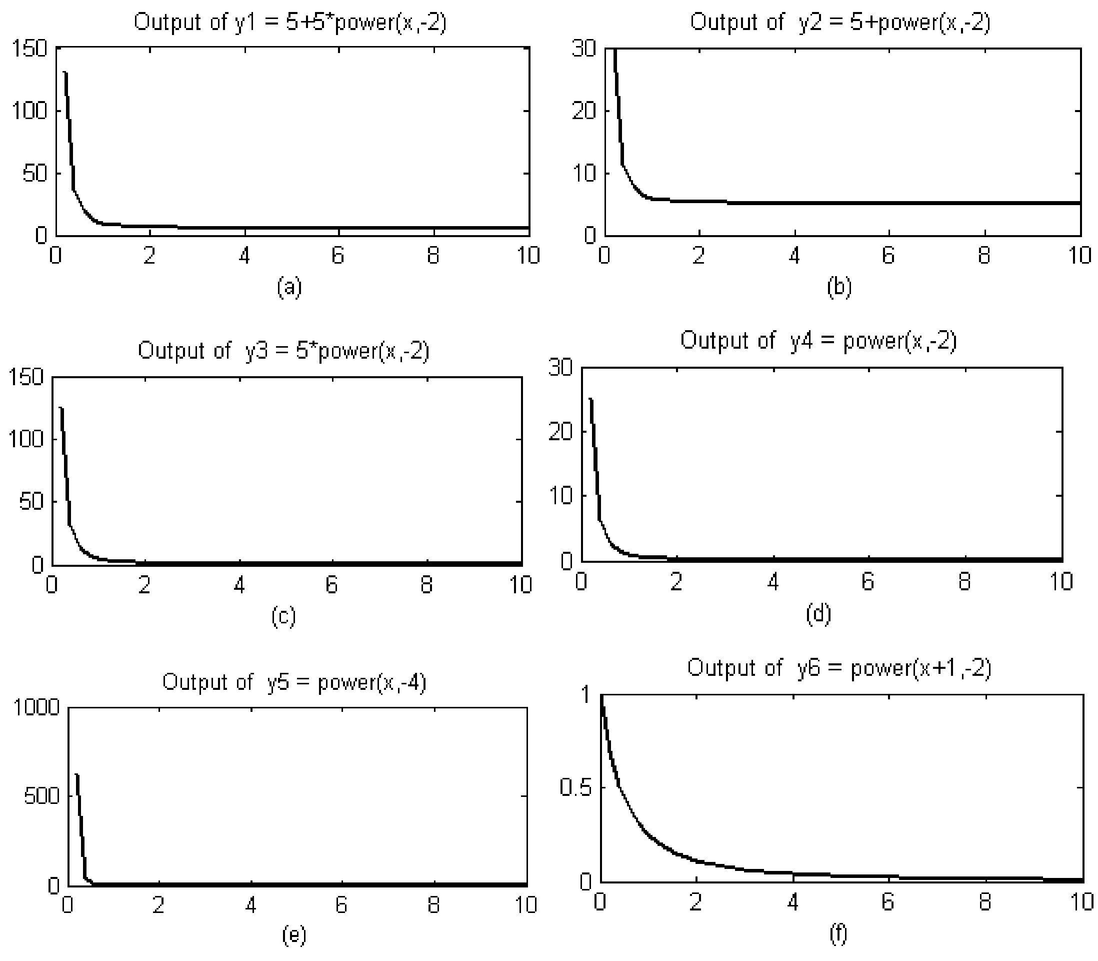


Figure 8: Outputs of nonlinear function $y=k_1+k_2*x^n$ with different co-efficient values (Graph produced by MATLAB)

In the figure(8), the only function that goes best with our proposition is Fig:8. But the problem with this function is, the graph has to be plotted against a function that is not directly a function of x [i.e. power(x+1,k)]

Outputs from some exponential functions are provided below.

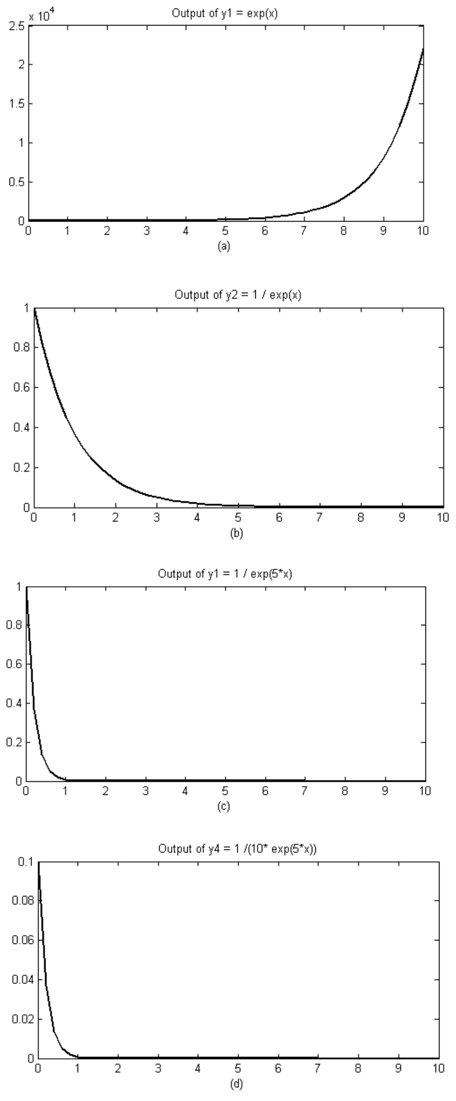


Figure 9: Outputs of exponential function $y=1/k_1*e^{k2*x}$ with different co-efficient values (graph produced by MATLAB)

From the above figure(9), this is evident that, only inverse exponential functions can correctly represent practical nature of diminishing benefits due to adjacency.

So, we can rationally settle with a function like adjacency=$1/k_1*e^{(k_2*x)}$, where k_1 and k_2 are two coefficients that depends on nature of industry and production floor.

CHAPTER 5: MODEL VALIDATION

To illustrate the effectiveness of the model using exponential function we developed a program implementing all the constraints mentioned in the model using MATLAB programming language. For testing purpose we considered 4 departments with fixed dimensions. The details of the input parameters are as follows:

5.1 Input Parameters

Number of Departments: 04 (Four)
Dimension of the facility: 30 X 20 meters
Dimension of the departments: 2 X 14, 20 X 14, 8 X 20, 22 X 6 meters

Flow Matrix: $\begin{bmatrix} 80 & 80 & 80 & 80 \\ 80 & 80 & 760 & 80 \\ 80 & 760 & 80 & 80 \\ 80 & 80 & 80 & 80 \end{bmatrix}$

Cost of Flow matrix: $\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$

Reward matrix: $\begin{bmatrix} 0 & 0 & 200 & 0 \\ 0 & 0 & 400 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$

$K_1=1$ and $K_2=1$;

5.2 Difference in solutions

From the input parameters, this is evident that flow of materials between department 2 & 3 is relatively very much higher than any other pairs as well as reward value (in reward matrix) for this pair is again relatively higher than the pair of department 1 &

3. The rationale behind high reward values for these two pairs may be beyond mere cost of flow of materials. Sometimes administrative requirements may cause higher reward values between two departments.

Now, if we strictly follow binary reward values for adjacency matrix (1 for gap 0 between two departments and 0 for otherwise), the model comes up with a solution where department 2 and 3 are essentially located wall-to-wall. These solutions even leave department #1 at one corner of the facility even if the department id quite thin and have good flow of materials with department #3. The four symmetric solutions are depicted in figure (10) below.

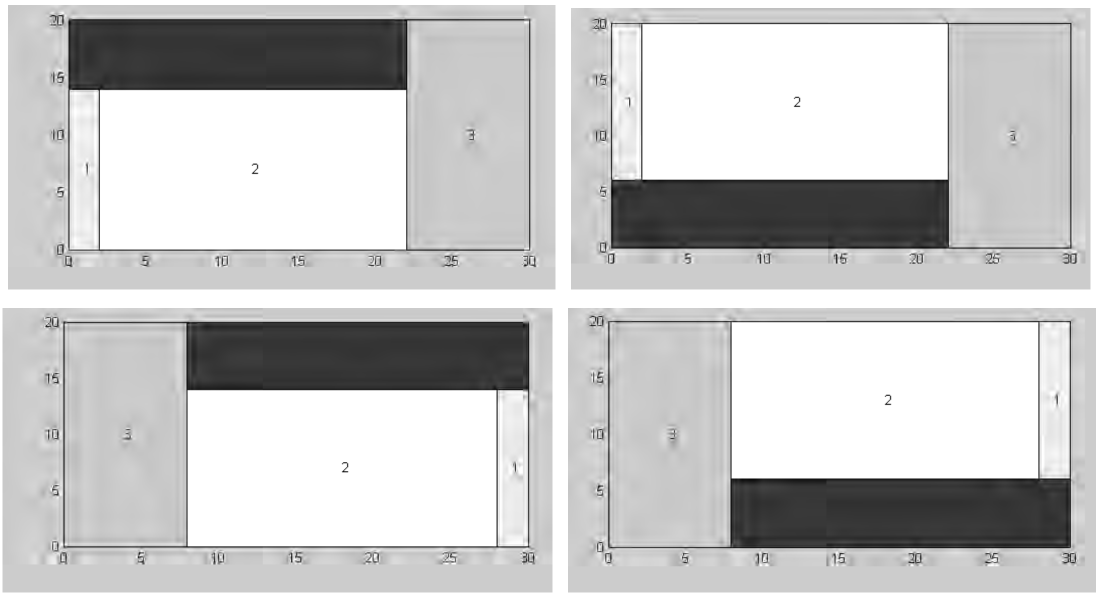


Figure 10: Solution sets using binary adjacency values (graph produced by MATLAB)

Practically thinking, positioning department #1 (width of which is very small) between #2 and #3 does not necessarily cancel out all benefits that are supposed to be there if departments #2 & #3 were in completely adjacent.

After running the same model using proposed exponential function for constructing adjacency matrix we get rather more rational solutions with same sets of input parameters. The solution sets are depicted in figure(11) below:

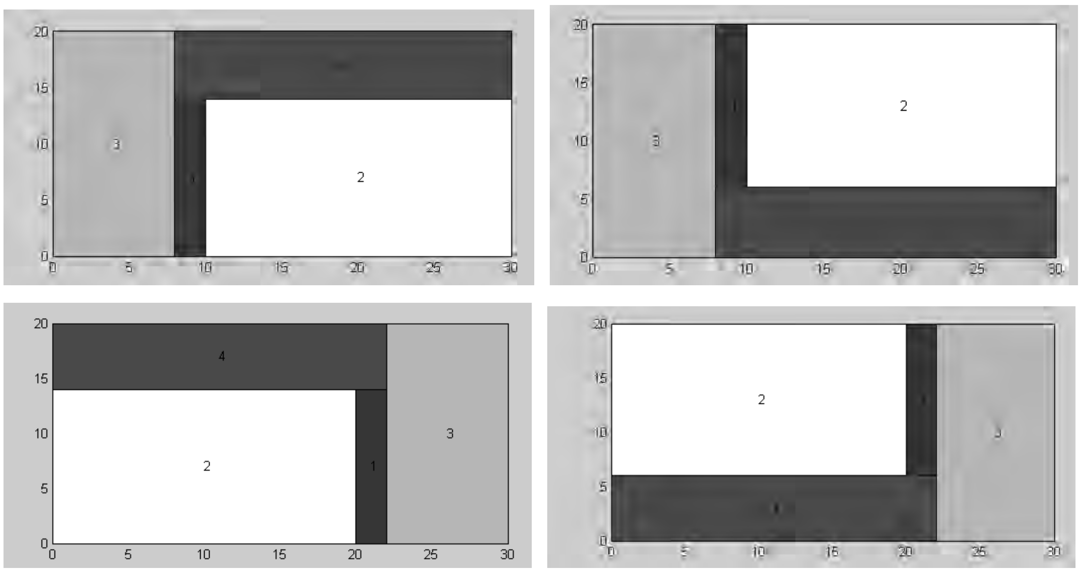


Figure 11: Solution sets using exponential function for generating adjacency values (graph produced by MATLAB)

So, from the above discussion, this is evident that, proposed model with continuous values for adjacency generated from exponential function instead of binary values gives better solution in practical consideration.

CHAPTER 6: FIREFLY ALGORITHM

6.1 Theoretical Background

Firefly Algorithm (FA) was first developed by Xin-She Yang in late 2007 and 2008 at Cambridge University, which was based on the flashing patterns and behavior of fireflies. The theoretical side of this algorithm is explained in Chapter 6. In essence, FA uses the following three idealized rules:

• Fireflies are unisex so that one firefly will be attracted to other fireflies regardless of their sex.

• The attractiveness is proportional to the brightness, and they both decrease as their distance increases. Thus for any two flashing fireflies, the less bright one will move towards the brighter one. If there is no brighter one than a particular firefly, it will move randomly.

• The brightness of a firefly is determined by the landscape of the objective function.

As a firefly's attractiveness is proportional to the light intensity seen by adjacent fireflies, we can now define the variation of attractiveness β with the distance r by $\beta = \beta_0 e^{-\gamma r^2}$, where β_0 is the attractiveness at r = 0.

The movement of a firefly i is attracted to another more attractive (brighter) firefly j is determined by $\boldsymbol{x}_i^{t+1} = \boldsymbol{x}_i^t + \beta_0 e^{-\gamma r_{ij}^2}(\boldsymbol{x}_j^t - \boldsymbol{x}_i^t) + \alpha_t \,\boldsymbol{\epsilon}_i^t,$

where the second term is due to the attraction. The third term is randomization with α_t being the randomization parameter, and ϵ_i^t is a vector of random numbers drawn from a Gaussian distribution or uniform distribution at time t. If $\beta_0 = 0$, it becomes a simple random walk. On the other hand, if $\gamma = 0$, it reduces to a variant of particle swarm optimization. Furthermore, the randomization ϵ_i^t can easily be extended to other distributions such as L´evy flights. A demo version of firefly algorithm implementation by Xin-She Yang, without L´evy flights for simplicity, can be found at Mathworks file exchange website (*www.**mathworks**.co.uk/**matlab**central/**fileexchange**/*).

Parameter Settings

As α_t essentially control the randomness (or, to some extent, the diversity of solutions), we can tune this parameter during iterations so that it can vary with the iteration counter t. So a good way to express α_t is to use, $\alpha_t = \alpha_0 \delta^t, \quad 0 < \delta < 1)$ where α_0 is the initial randomness scaling factor, and δ is essentially a cooling factor. For most applications, we can use $\delta = 0.95$ to 0.97.

Regarding the initial α_0, simulations show that FA will be more efficient if α0 is associated with the scalings of design variables. Let L be the average scale of the problem of interest, we can set $\alpha_0 = 0.01L$ initially. The factor 0.01 comes from the fact that random walks requires a number of steps to reach the target while balancing the local exploitation without jumping too far in a few steps.

The parameter β controls the attractiveness, and parametric studies suggest that $\beta_0 = 1$ can be used for most applications. However, γ should be also related to the scaling L. In general, we can set $\gamma = 1/\sqrt{L}$. If the scaling variations are not significant, then we can set

$\gamma = O(1)$.

For most applications, we can use the population size n = 15 to 100, though the best range is n = 25 to 40.

6.2 Test Problem

Number of Departments: 05 (Five)

Dimension of the facility: 300 X 200 meters

Dimension of the departments: 8 X 4, 1 X 8, 8 X 14, 8 X 20 and 20 X 6 meters

$$\text{Flow Matrix: } \begin{bmatrix} 10 & 10 & 10 & 10 & 10 \\ 10 & 10 & 10 & 10 & 10 \\ 10 & 10 & 10 & 10 & 10 \\ 10 & 10 & 10 & 10 & 10 \\ 10 & 10 & 10 & 10 & 10 \end{bmatrix}$$

$$\text{Cost of Flow matrix: } \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

$$\text{Reward matrix: } \begin{bmatrix} 0 & 0 & 20 & 0 & 0 \\ 0 & 0 & 400 & 50 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

6.3 Performance of the algorithm

6.3.1 Convergence

The figure 12 below shows the convergence rate of the firefly algorithm across different numbers of fireflies. The figures depicts that, the rate is slowest for comparatively higher numbers of fireflies. The rate of convergence with 10 fireflies is approximately double of that with 40 fireflies. But in all the cases, the result is not near to the global optimum, meaning, firefly algorithm is not giving the global optimum value.

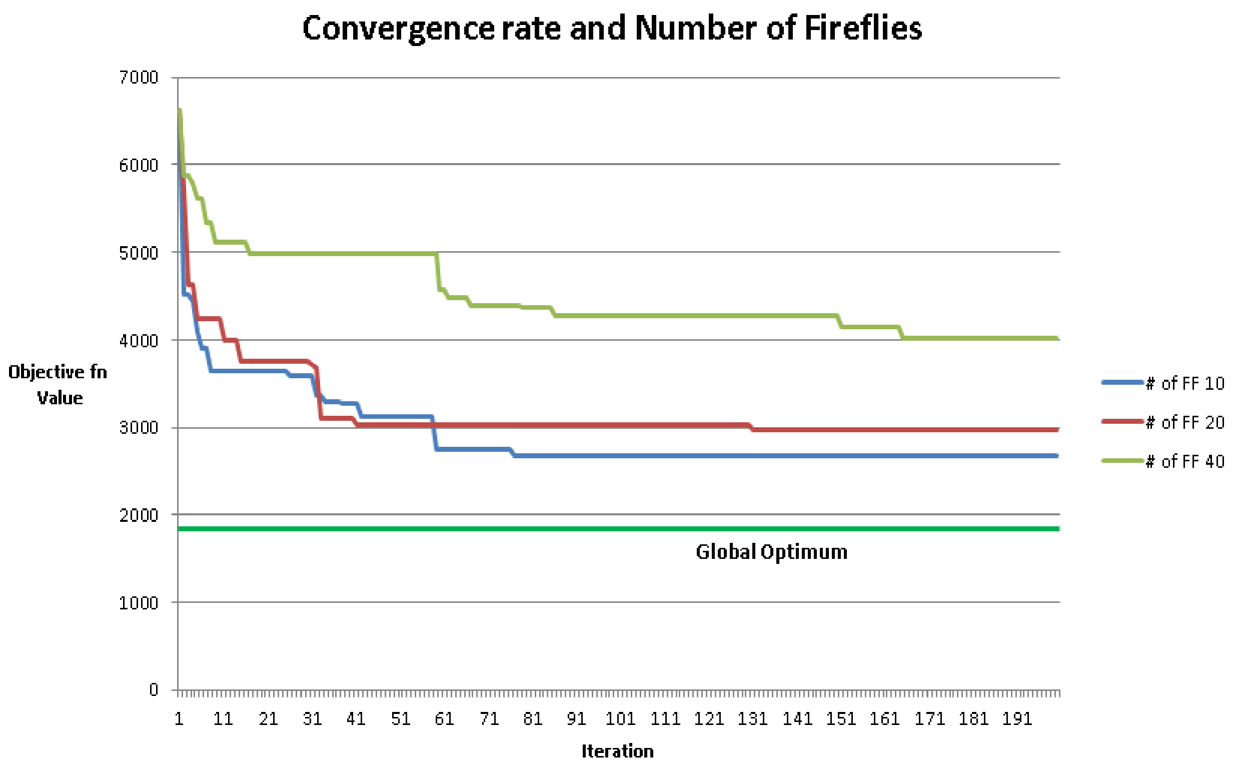


Figure 12: Convergence of the solution with increase of number of fireflies

Figure(13) shows the convergence rate with the increase of number of iterations. The optimum objective function value found with 400 iterations and 200 iterations are quite same (2600), whereas the optimum value is more than 3000 with 100 iterations. Again, the algorithm fails to give values or results near to global optimum found with exact algorithm (1800).

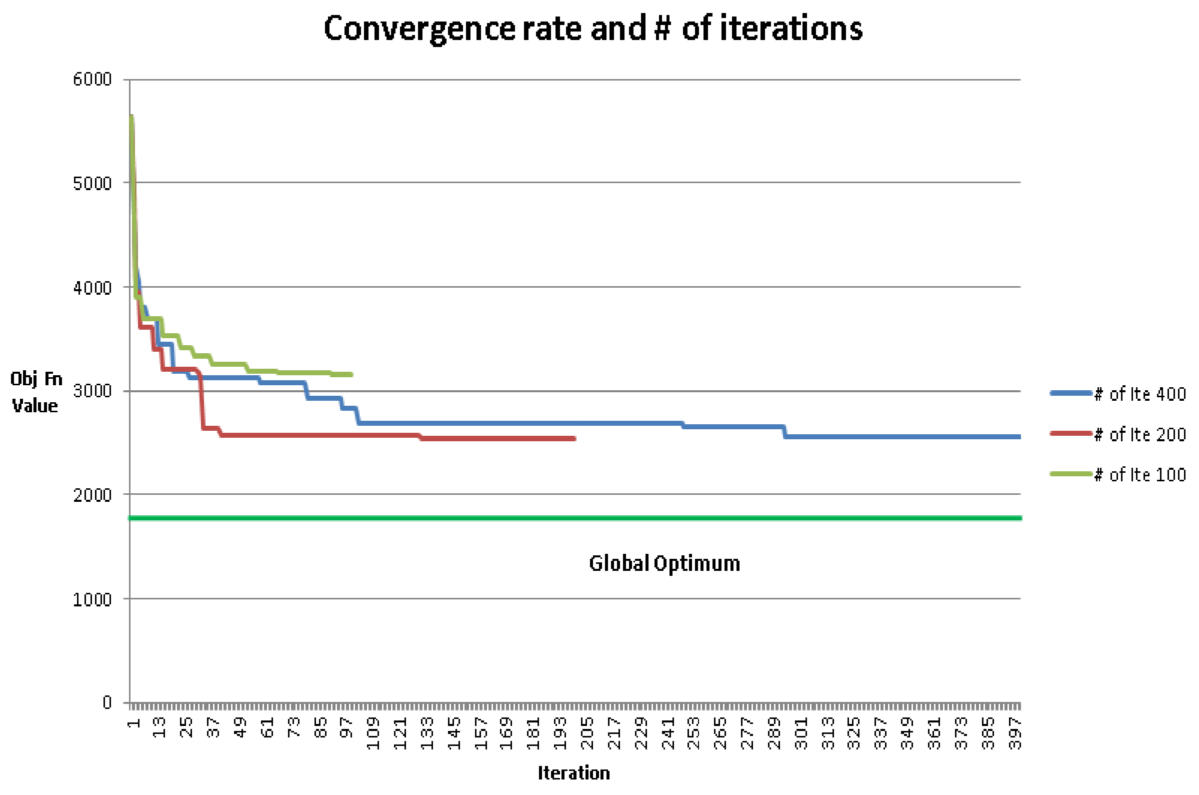


Figure 13: Convergence of the solution with increase of number of iterations

6.3.2 Error rate

Error rate is calculated as the percentage ratio of error (the gap between optimum solution and found solution from the algorithm) and the optimum solution. It is found that the error rate of the algorithm is quite high for the firefly algorithm for solving facility layout problems with different variations. The following table shows the percentage error rates of firefly algorithm for different number of fireflies for 200 iterations. In the previous section, it was shown that, convergence rate is quite slow for higher number of fireflies (40). And thus the error rate with 40 fireflies with 200 iterations is 123% whereas the same for 10 fireflies is 48%. It is to note here that the objective function value of global optimum solution is 1800.

Table 4: Error percentage for different number of Fireflies after 200 iterations

# of fireflies	Optimum Obj Function Value	Error Percentage
10	2671.31	48%
20	2973.97	65%
40	4010.17	123%

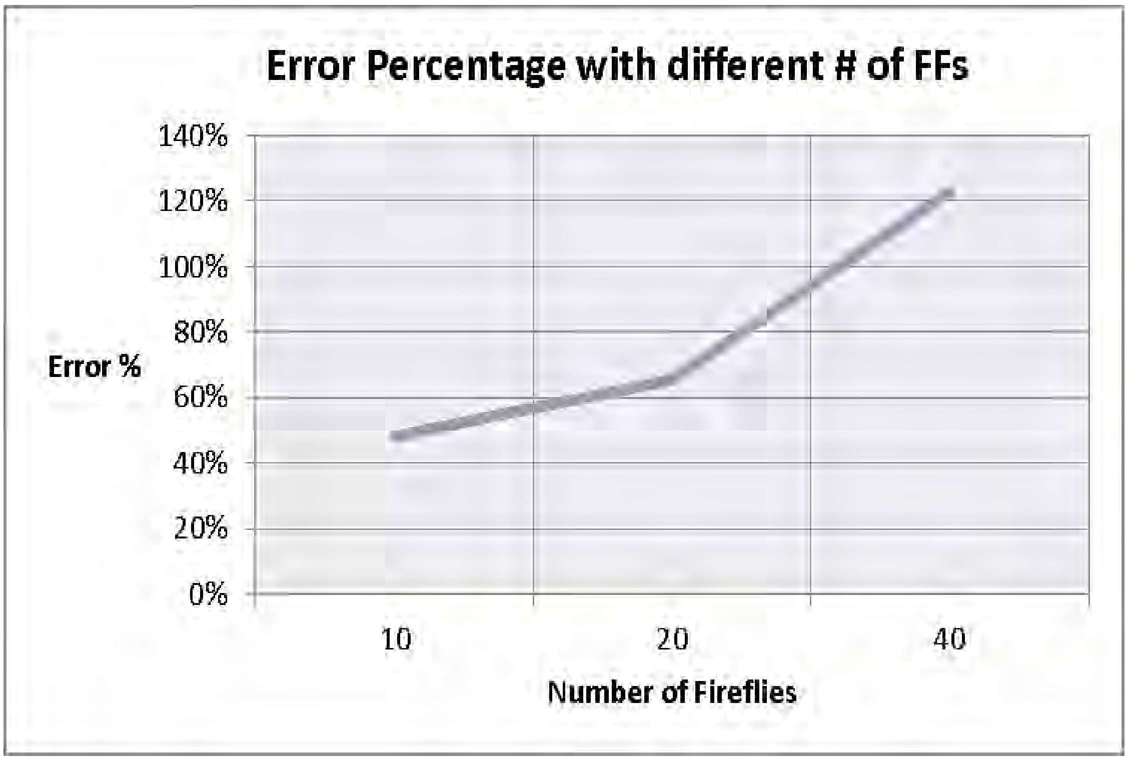


Figure 14: Error Percentage rate with varying number of FFs

If number of iterations is increased, keeping number of fireflies constant, then comparatively the error percentage value is decreased. For example, as depicted in the following table, for 100 iterations with 20 fireflies is 75% whereas the same for higher number of iterations (i.e. 200 and 400 iterations) is around 40%.

Table 5: Error Percentage for different number of iterations with 20 fireflies

# of iterations	Objective function Value	Error Percentage
400	2561.975	42%
200	2533.717	41%
100	3151.209	75%

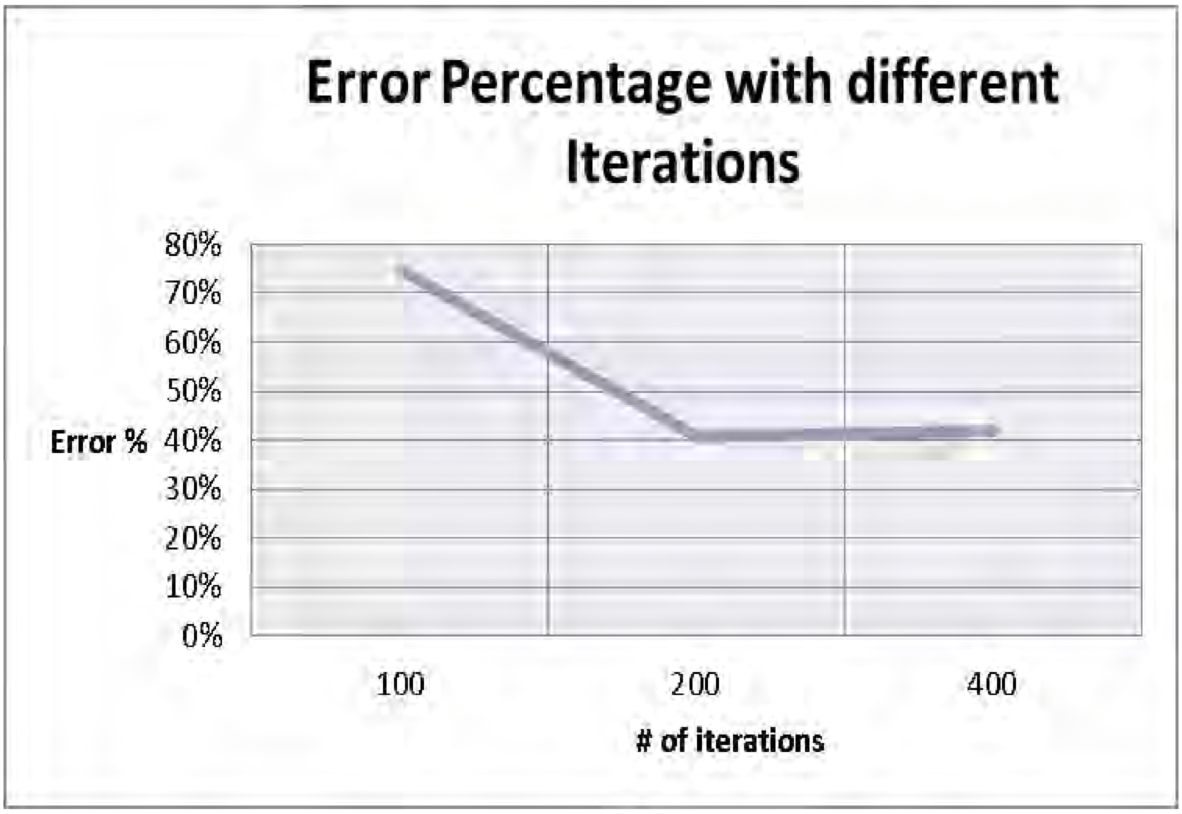


Figure 15: Error Percentage rate with varying number of iterations

CHAPTER 7: APPLICATION OF PROPOSED MODEL FOR FLP

7.1 British American Tobacco of Bangladesh (BATB)

BATB is one of the largest multinational companies in the country and has been operating for over 100 years.

BATB has over 1,000 employees and BATB take pride in being one of the preferred employers in Bangladesh. Responsibility - to shareholders, employees, business partners, customers and any other stakeholders - is at the core of the business and that is why BATB believes “success and responsibility go together”.

BATB is part of the British American Tobacco Group, the world's most international tobacco group with brands sold in more than 180 markets.

7.2 Units at Production Site

The layout problem that we dealt with had 11 units of different dimensions including some maintenance and administrative units. Out of the 11 departments, we ended up with 05 production related units only where the maintenance units were merged within respective units, for example, rather than considering PMD and its maintenance units as two separate units, we considered them as a single unit. This consideration is nonetheless very rational as because the maintenance units are no way should be placed in distance from respective units. And we also excluded the administrative units from our problem as because the units do not have any transfer of goods with the production related units and thus these administrative units can be placed in group or else in any fashion without hampering the main production process.

Another realistic reason for minimizing our problem definition is, the exact algorithm that we used as our base algorithm in this thesis face difficulties as far as time is constrained to come up with solution. This is to emphasize here that, the concern of this thesis was not doing performance analysis of exact algorithm with different size of problem definition, or even not comparative performance analysis between exact algorithm and heuristic algorithms or else. Our findings, rather was at optimality test level, rather than in generating feasible solution space level. So, keeping problem space manageable, we instead concentrated on comparative analysis between traditional binary ways of generating reward values with our proposed function dealing with continuous reward values.

In a word, the algorithm has been applied in getting optimized facility layout for production floor comprising mainly of 5 departments.

In the production floor, as mentioned, are 05 units/departments, dimensions of which are as follows:

Table 6: List of Departments at BATB

No.	Unit	Dimension
1	Leaf Warehouse	L-90’ x W-80’
2	PMD with Maintenance Unit	L-300’ x W-80’
3	CTS	L-96’ x W-110’
4	SMD with Maintenance Unit	L-330’ x W-110’
5	Finish Goods Warehouse	L-148’ x W-71’

Flow of materials is 1→ 2 → → 3 → 4 → → 5

This requirement regarding flow of materials is incorporated in flow of materials matrix:

$$\begin{pmatrix} 0 & 200 & 0 & 0 & 0 \\ 0 & 0 & 200 & 200 & 0 \\ 0 & 0 & 0 & 200 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Relationship requirements

>   1 & 2 must be close together>   2 & 4 must be close together

>   3 must be middle of 2 and 4

This relationship requirement is reflected in Adjacency Reward matrix:

$$\begin{pmatrix} 0 & 80 & 0 & 0 & 0 \\ 0 & 0 & 80 & 0 & 0 \\ 0 & 0 & 0 & 80 & 0 \\ 0 & 0 & 0 & 0 & 80 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

7.3 Solution Using Exact Algorithm

The solutions from the exact algorithm both using adjacency function and without function are same and provided as follow:

Objective Function Value = 2260

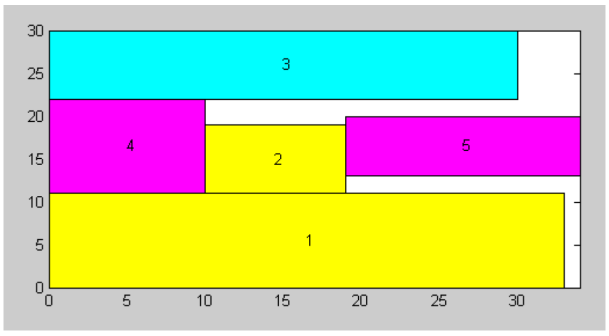
And the co-ordinates are:

(17, 6), (15, 15), (15, 26), (5, 17), (27, 17)

and

(17, 6), (15, 18), (15, 26), (5, 17), (27, 17)

The solutions are provided in the following diagram



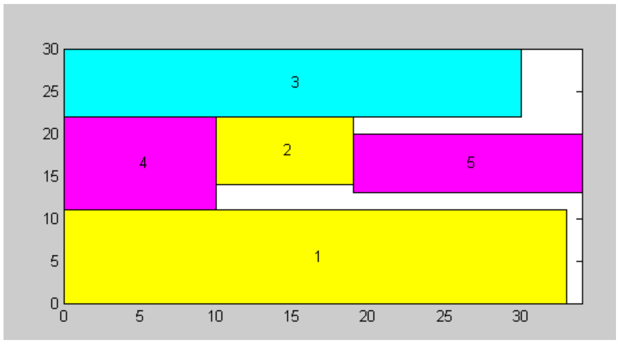


Figure 16: Layout Solutions (in diagram)

So, unlike the test problem, in the case for BATB problem, the proposed solution with adjacency matrix created based on proposed function does not create any difference as far as solutions are concerned.

7.4 Solution Using FF Algorithm

After running the algorithm for several times, no solution could be found by firefly algorithm even with varying number of fireflies. The underlying reason for the phenomenon is: unlike exact algorithm, where every possible feasible solution is tried in order to get the optimum solution, firefly algorithm starts with some randomly generated solutions that in most cases found to be infeasible after checking with constraints. This is truer if the problem is tightly constrained, for example, for facility problems, if the space is tightly constrained likes the BATB problem. In the BATB problem, the dimension of the whole space is 34X30 whereas those of the largest two departments are 33X11 and 30X8.

CHAPTER 8: CONCLUSION AND RECOMMENDATION

8.1 Concluding Remarks

The productivity and efficiency of an organization greatly depends on how people plan, organize and utilize the facilities in that organization. From an upfront investment and recurring project expense, facilities planning are a critical issue in today's competitive manufacturing and service sectors. In addition to the upfront investment involved in facilities planning, there are operational issues that make facilities planning a critical issue. The most obvious impact is on material handling expenses. The impact of the facility layout goes beyond material handling costs. An effective facility layout implies that departments with high flow are close together.

In facility layout problems, objective functions are modeled with different objectives in mind examples of which include minimization of cost or flow of materials, maximizations of closeness rewards etc. The mathematical model with a continuous representation of distance based adjacency matrix proposed in this thesis deems to provide more realistic optimal layout.

8.2 Scope of Future Works

There are many scopes for future works:

- In this thesis, exact algorithm is used for finding feasible solution set from the total solution space. Further research can be done using other heuristic algorithms.
- In the function [adjacency=$1/(k_1*e^{k_2*x})$] proposed in this thesis for generating continuous value adjacency matrix has two co-efficients, namely k_1(Denominator co-efficient) and k_2 (exponential co-efficient). We have assumed unit value for both of the co-efficient, whereas we strongly believe that, these two has industry specific values. There is huge scope of econometrical research to come up with series of values of k_1 and k_2 for different industries.

REFERENCES

[1] Tompkins, J. A., White, J. A., Bozer, Y. A., and Tanchoco, J. M. A., Facilities Planning, Wiley, New York, New York, 3rd edition (2003).

[2] Sule, D. R., Manufacturing Facilities Location, Planning, and Design, PWS Publishing, Boston, MA (1994).

[3] Bozer, Y. A., Meller, R. D., and Erlebacher, S. J., "An Improvement-Type Layout Algorithm for Single and Multiple Floor Facilities," Management Science, 40, 7, 918–932 (1994)

[4] Koopmans, T. C., and Beckman, M., "Assignment Problems and the Location of Economic Activities," Econometrica, 25, 53–76 (1957)

[5] Drezner, Z., Hahn, P. M., and Taillard, E. D., "Recent Advances for the Quadratic Assignment Problem with Special Emphasis on Instances that are Difficult for Meta-heuristic Methods," The Annals of Operations Research: State of the Art and Recent Advances in Integer Programming (2004), accepted for publication.

[6] Montreuil, B., "A Modelling Framework for Integrating Layout Design and Flow Network Design," In Proceedings from the Material Handling Research Colloquium, pp.43–58, Hebron, Kentucky (1990).

[7] Meller, R. D., Narayanan, V., and Vance, P. H., "Optimal Facility Layout Design," Operations Research Letters, 23, 117–127 (1998).

[8] Meller, R. D., and Gau, K.-Y., "The Facility Layout Problem: Recent and Emerging Trends and Perspectives," Journal of Manufacturing Systems, 15, 5, 351–366 (1996).

[9] Bozer, Y. A., and Meller, R. D., "A Reexamination of the Distance-Based Facility Layout Problem," IIE Transactions on Design and Manufacturing, 29, 7, 549–560 (1997).

[10] Norman, B. A., Arapoglu, R. A., and Smith, A. E., "Integrated Facilities Design using a Contour Distance Metric," IIE Transactions on Design & Manufacturing, 33, 4, 337–344 (2001)

[11] Arapoglu, R. A., Norman, B. A., and Smith, A. E., "Locating Input and Output Points in Facilities Design: A Comparison of Constructive, Evolutionary and Exact Methods," IEEE Transactions on Evolutionary Computation, 5, 192–203 (2001)
[12] Rosenblatt, M. J., "The Facilities Layout Problem: A Multi-Goal Approach," International Journal of Production Research, 17, 4, 323–332 (1979)
[13] Dutta, K. N., and Sahu, S., "A Multi-Goal Heuristic for Facilities Design Problems: MUGHAL," International Journal of Production Research, 20, 147–154 (1982).
[14] Rosenblatt, M. J., and Golany, B., "A Distance Assignment Approach to the Facility Layout Problem," European Journal of Operational Research, 57, 253–270 (1992).
[15] Garey, M. R., and Johnson, D. S., Computers and Intractability: A Guide to the Theory of NP-Completeness, W. H. Freeman and Company, New York, New York (1979).
[16] Kettani, O., and Oral, M., "Reformulating Quadratic Assignment Problems for Efficient Optimization," IIE Transactions, 25, 97–107 (1993)
[17] Liao, T. W., "Design of Line Type Cellular Manufacturing Systems for Minimum Operating and Total Material Handling Costs," International Journal of Production Research, 32, 2, 387–397 (1993)
[18] Bozer, Y. A., and Rim, S. C., "A Branch and Bound Method for Solving the Bidirectional Circular Layout Problem," Applied Mathematical Modelling, 20, 342–351 (1996).
[19] Mitchell, M., "An Introduction to Genetic Algorithms", MIT Press, Cambridge, MA (1999)
[20] Giffin, J. W., Graph Theoretic Techniques for Facility Layout, PhD thesis, University of Canterbury, Christchurch, New Zealand (1984)
[21] Hassan, M. M. D., and Hogg, G. L., "A Review of Graph Theory Applications to the Facilities Layout Problem," Omega, 15, 4, 291–300 (1987)

[22] Sherali, H. D., Fraticelli, B. M. P., and Meller, R. D., “Enhanced Model Formulations for Optimal Facility Layout,” Operations Research, 51, 4, 629–644 (2003).

[23] Banerjee, P., Montreuil, B., Moodie, C. L., and Kashyap, R. L., “A Modelling of Interactive Facilities Layout Designer Reasoning Using Qualitative Patterns,” International Journal of Production Research, 30, 3, 433–453 (1992).

[24] Montreuil, B., Ratliff, H. D., and Goetschalckx, M., “Matching Based Interactive Facility Layout,” IIE Transactions, 19, 3, 271–279 (1987).

[25] Hassan, M. M. D., Hogg, G. L., and Smith, D. R., “SHAPE: A Construction Algorithm for Area Placement Evaluation,” International Journal of Production Research, 24, 1283–1295 (1986).

[26] Armour, G. C., and Buffa, E. S., “A Heuristic Algorithm and Simulation Approach to Relative Allocation of Facilities,” Management Science, 9, 294–309 (1963).

[27] DePuy, G. W., Usher, J. S., and Miles, T., “Facilities Layout Using a CRAFT Meta-Heuristic,” In Proceedings of the 2004 Industrial Engineering Research Conference (2004).

[28] Foulds, L. R., and Robinson, D. F., “Graph Theoretic Heuristics for the Plant Layout Problem,” International Journal of Production Research, 16, 1, 27–37 (1978).

[29] Tompkins, Al-Hakim, L. A., “A Modified Procedure for Converting a Dual Graph to a Block Layout,” International Journal of Production Research, 30, 10, 2467–2476 (1992).

[30] Boswell, S. G., “TESSA – A New Greedy Algorithm for Facilities Layout Planning,” International Journal of Production Research, 30, 8, 1957–1968 (1992).

[31] Leung, J., “A New Graph-Theoretic Heuristic for Facility Layout,” Management Science, 38, 4, 594–605 (1992)

[32] Goetschalckx, M., “An Interactive Layout Heuristic Based on Hexagonal Adjacency Graphs,” European Journal of Operational Research, 63, 304–321 (1992)

[33] Tam, K. Y., “A Simulated Annealing Algorithm for Allocating Space to Manufacturing Cells,” International Journal of Production Research, 30, 63–87 (1992).

[34] van Camp, D. J., Carter, M. W., and Vannelli, A., “A Nonlinear Optimization Approach for Solving Facility Layout Problems,” European Journal of Operational Research, 57, 174–189 (1991).

[35] Montreuil, B., Venkatadri, U., and Ratliff, H. D., “Generating a Layout from a Design Skeleton,” IIE Transactions, 25, 1, 3–15 (1993).

[36] Lacksonen, T. A., “Preprocessing for Static and Dynamic Facility Layout Problems,” International Journal of Production Research, 35, 4, 1095–1106 (1997).

[37] Langevin, A., Montreuil, B., and Riopel, D., “Spine Layout Design,” International Journal of Production Research, 32, 2, 429–442 (1994).

[38] Heragu, S. S., and Kusiak, A., “Machine Layout Problem in Flexible Manufacturing Systems,” Operations Research, 36, 2, 258–268 (1988).

[39] Montreuil1, B., Ouazzani, N., Brotherton, E., and Nourelfath, M., “Antzone Layout Metaheuristic: Coupling Zone-Based Layout Optimization, Ant Colony System and Domain Knowledge,” In Proceedings of the 2004 IMHRC (2004).

[40] Kim, J. G., and Kim, Y. D., “Layout planning for facilities with xed shapes and input and output points,” International Journal of Production Research, 38, 4635–4653 (2000).

[41] Irohara, T., and Yamada, T., “Location Matrix Based Design Methodology for the Facility Layout Problem Including Aisles and Door Locations,” In Proceedings of the 2004 IMHRC (2004).

[42] Coello Coello, C.A., Lamont, G.B. and Van Veldhuizen, D.A. (2007) Evolutionary Algorithms for Solving Multi-Objective Problems, Springer, New York.

[43] Castillo, I. and Peters, B.A. (2003) ‘An extended distance-based facility layout problem’, Int. J. Production Research, Vol. 41, No. 11, pp.2451–2479.

[44] Dong, M., Wu, C. and Hou, F. (2009) ‘Shortest path based simulated annealing algorithm for dynamic facility layout problem under dynamic business environment’, Expert Systems with Applications, Vol. 36, No. 8, pp.11221–11232.

[45] McKendall Jr., A.R., Shanga, J. and Kuppusamy, S. (2006) 'Simulated annealing heuristics for the dynamic facility layout problem', Computers and Operations Research, Vol. 33, pp.2431–2444.
[46] Sùahin, R. and Türkbey, O. (2009) 'A simulated annealing algorithm to find approximate Pareto optimal solutions for the multi-objective facility layout problem', Int. J. Advanced Manufacturing Technology, Vol. 41, pp.1003–1018.
[47] Sahin, R., Ertog˘ral, K. and Türkbey, O. (2010) 'A simulated annealing heuristic for the dynamic layout problem with budget constraint', Computers and Industrial Engineering, Vol. 59, pp.308–313.
[48] Souilah, A. (1995) 'Simulated annealing for manufacturing systems layout design', European Journal of Operational Research, Vol. 82, pp.592–614.
[49] Ioannou, G. (2007) 'An integrated model and a decomposition-based approach for concurrent layout and material handling system design', Computers and Industrial Engineering, Vol. 52, pp.459–485.
[50] Sugiyono, A. (2006) 'Cellular manufacturing system application on redesign production layout with using heuristics algorithm', In 2006 IEEE International Conference on Management of Innovation and Technology, pp.940–944.
[51] Matsuzaki, K., Irohara, T. and Yoshimoto, K. (1999) 'Heuristic algorithm to solve the multi-floor layout problem with the consideration of elevator utilization', Computers and Industrial Engineering, Vol. 36, pp.487–502.
[52] Mir, M. and Imam, M.H. (2001) 'A hybrid optimization approach for layout design of unequal-area facilities', Computers and Industrial Engineering, Vol. 39, pp.49–63.
[53] Singh, S.P. (2009) 'Solving facility layout problem: three-level tabu search meta-heuristic approach', Int. J. Recent Trends in Engineering, Vol. 1, No. 1, pp. 73–77.
[54] Wang, G., Yan, Y., Zhang, X., Shangguan, J. and Xiao, Y. (2008a) 'Integrating simulation optimization with VR for facility layout evaluation', Proceedings of the 2008 IEEE IEEM.

[55] Eklund, N.H.W., Embrechts, M.J. and Goetschalckx, M. (2006) 'Efficient chromosome encoding and problem-specific mutation methods for the flexible bay facility layout problem', IEEE Transactions on Systems, Man, and Cybernetics-PART C: Applications and Reviews, Vol. 36, No. 4, pp.495–502.

[56] Wu, Y. and Appleton, E. (2002) 'The optimization of block layout and aisle structure by a genetic algorithm', Computers and Industrial Engineering, Vol. 41, pp.371–387.

[57] Osman, H.M., Georgy, M.E. and Ibrahim, M.E. (2003) 'A hybrid CAD-based construction site layout planning system using genetic algorithms', Automation in Construction, Vol. 12, pp.749–764.

[58] X.S. Yang. A new meta-heuristic bat-inspired algorithm. Nature Inspired Cooperative Strategies for Optimization (NICSO 2010), pages 65–74, 2010.

[59] Zong Woo Geem, Joong Hoon Kim, and GV Loganathan. A new heuristic optimization algorithm: harmony search. Simulation, 76(2):60–68, 2001.

[60] J.H. Holland, Adaptation in Natural and Artificial Systems, University of Michigan Press, Ann Arbor, MI, 1975.

[61] S. Koakutsu, and H. Hirata, "Genetic Simulated Annealing for Floor plan Design", Chiba University, Japan, 1992.

[62] P. Banerjee, and Y. Zhou, "Facilities Layout Design Optimization with Single Loop Material Flow Path Configuration", International Journal of Production Research, 33, 1995, pp. 183-203.

[63] P. Banerjee, Y. Zhou, and B. Montreuil, "Genetically Induced Optimization of Facilities Layout Design", Technical Report TR-92-18, Dept. of Mechanical Engineering, University of Illinois, Chicago, 1992.

[64] D.M. Tate, and A.E. Smith, "Unequal-Area Facility Layout by Genetic Search", IIE Transactions, 27, 1995, pp. 465-472.

[65] H. Muhlenbein, "Parallel Genetic Algorithms, Population Genetics and Combinatorial Optimization", Lecture Notes in Computer Science, 565, 1989, pp. 398-406.

[66] C.L. Huntley, and D.E. Brown, “A Parallel Heuristic for Quadratic Assignment Problem”, Computers and OR, 18, 1991, pp. 275-289.

[67] R. Battiti and G. Tecchiolli, “Parallel Biased Search for Combinatorial Optimization: Genetic Algorithms and TABU”, Microprocessors and Microsystems, 16, 1992, pp. 351-367.

[68] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi, “Optimization by Simulated Annealing”, Science, 220, 1983, pp. 671-680.

[69] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller, “Equation of State Calculations by Fast Computing Machines”, Journal of Chemical Physics, 21, 1953, pp. 1087-1092.

[70] S.S. Heragu, and A.S. Alfa, “Experimental Analysis of Simulated Annealing based Algorithms for the Layout Problem”, European Journal of Operational Research, 57, 1992, pp. 190-202.

[71] S.S. Heragu, and A. Kusiak, “Efficient Models for the Facility Layout Problem”, European Journal of Operational Research, 53, 1991, pp. 1-13.

[72] M.R. Wilhelm, and T.L. Ward, “Solving Quadratic Assignment Problems by Simulated Annealing”, IIE Transactions, 19, 1987, pp. 107-119.

[73] S. Jajodia, I. Minis, G. Harhalakis, and J.M. Proth, “CLASS: Computerized Layout Solutions using Simulated Annealing”, International Journal of Production Research, 30, 1, 1992, pp. 95-108.

APPENDIX

Matlab Code: Exact Algorithm

```
function [N,numSol,L,l,cf]=FLPN_v4()
k1=1;k2=1; %value of co-effecients in the exponential function
% Opening Text file and reading dimensions & initializing respective
% parameters
fileID=fopen('temp.txt','a+');
fprintf(fileID,'\n %s \n',' -- from file --------');
fid=fopen('input4.txt');
N = fscanf(fid, 'N=%d\n', 1);
L(1) = fscanf(fid, 'L=%d', 1);
L(2) = fscanf(fid, '%d\n', 1);

fscanf(fid, '--sizeofDept--\n', 1);
formatSpec = '%d %d';
sizeA = [2 N];
l = fscanf(fid,formatSpec,sizeA);
l=l';
for p=1:1:N
   hl(p,1)=l(p,1)/2;
   hl(p,2)=l(p,2)/2;
end
% Reading Flow matrix
fscanf(fid, '%*s\n', 1);
formatSpec = repmat('%d', 1, N);
sizeA = [N N];
flow = fscanf(fid,formatSpec,sizeA);
```

```
flow=flow';
% Reading Cost of flow matrix
fscanf(fid, '%*s\n', 1);
formatSpec = repmat('%d', 1, N);
sizeA = [N N];
cost_flow = fscanf(fid,formatSpec,sizeA);
cost_flow=cost_flow';
% Reading Reward matrix
fscanf(fid, '%*s\n', 1);
formatSpec = repmat('%d', 1, N);
sizeA = [N N];
reward_adj = fscanf(fid,formatSpec,sizeA);
reward_adj=reward_adj';
fclose(fid);

flag_overlap=0;
cf(:,:,:)=0;  % Matrix for holding final positions of centeroids of departments

%Initilializing matrices for holding intemediate positions of centeroids
for p=1:1:N
    c(p,:)=hl(p,:);
    c_pos(p,:)=hl(p,:);
end
% Initializing Gap matrix
for p=1:1:N
    for q=1:1:N
        d(p,q,:)=[500 500 500];
    end
end
v=0; minval=9999999; sol_num=1;
```

```
walk2(1);

  function walk2(n)  % Function for walking in the solution space
  c_pos(n,:)=hl(n,:);
  while c_pos(n,1)<=L(1)-hl(n,1)
    c_pos(n,2)=hl(n,2);
    while c_pos(n,2)<=L(2)-hl(n,2)
      c(n,:)=c_pos(n,:);
      if (n>1)
        anyoverlap(n);
        if (flag_overlap) % checking if there is any overlap
          c_pos(n,2)=c_pos(n,2)+1; continue;
        end
      end
      if (n==N)
       create_distance_matrix(N)
       chk_solution(n);
      end
      if (n<N) walk2(n+1);end
      c_pos(n,2)=c_pos(n,2)+1; % move in Y direction
    end
    c_pos(n,1)=c_pos(n,1)+1; % move in X direction
  end
  c_pos(n,:)=hl(n,:);
  end

  function anyoverlap(n)  % Function for chcking if there is any overlap
    flag_overlap=0;
    for ov=1:1:(n-1)
      flag_overlap = flag_overlap ||ifoverlap(ov,n,c,hl,L);
```

```
        end
    end

    function create_distance_matrix(N)  % Function that builds the gap matrix
        for p=1:1:N
            for q=1:1:N
                d(p,q,:)=[500 500 500];
            end
        end

        for i=1:1:(N-1)
            for j=(i+1):1:N
                measure_distance(i,j);
            end
        end
    end

    function measure_distance(i,j)   % Function that measures gap between boundaries
of two departments
             if ((c(i,1)+hl(i,1))<=(c(j,1)-hl(j,1)))
                d(i,j,1) = (c(j,1)-hl(j,1))-(c(i,1)+hl(i,1));
             elseif ((c(j,1)+hl(j,1))<=(c(i,1)-hl(i,1)))
                d(i,j,1) = (c(i,1)-hl(i,1))-(c(j,1)+hl(j,1));
             else d(i,j,1)=0;
             end
             if ((c(i,2)+hl(i,2))<=(c(j,2)-hl(j,2)))
                d(i,j,2) =(c(j,2)-hl(j,2))-(c(i,2)+hl(i,2));
             elseif ((c(j,2)+hl(j,2))<=(c(i,2)-hl(i,2)))
                d(i,j,2) =(c(i,2)-hl(i,2))-(c(j,2)+hl(j,2));
             else d(i,j,2)=0;
            end
            d(i,j,3)= d(i,j,1)+d(i,j,2);
    end

    function chk_solution(n)   % check and update solution if new lower objective
value is reached
        alpha=0.5;
        cost=costfun(c,cost_flow,flow,N);
        reward=rewardfun(d,reward_adj,N);
        v= alpha*cost-(1-alpha)*reward;
          if (v==minval)
              sol_num=sol_num+1;
              for i=1:1:n
                  cf(sol_num,i,:)=c(i,:);
              end
          end
          if (v<minval)
              disp(v);
              fileID=fopen('temp.txt','a+');
               fprintf(fileID,'old minval:%0.0f ;new minval= %0.0f\n ',minval,v);
               fclose(fileID);
              cf=[]; sol_num=1;
              minval=v;
              for i=1:1:n
                  cf(1,i,:)=c(i,:);
              end
          end
        end
fileID=fopen('temp.txt','a+');
numSol=sol_num;
```

```
% Writing solution in output file
for i=1:1:sol_num
  for n=1:1:N
  fprintf(fileID,'%2.0f %2.0f ',cf(i,n,1),cf(i,n,2));
  end
fprintf(fileID,'\n');
end
fprintf(fileID,'\n %s \n',' ----- end --------');
fclose(fileID);
fclose('all');
end
%%%%%%%%%%%%%%%%%%%
% Check if there is any overlap between two departments
function isover=ifoverlap(d1,d2,c,hl,L)
isover=-1;
   if ((c(d1,1)+hl(d1,1))<=(c(d2,1)-hl(d2,1)))
      Zijx=1;
      else Zijx=0; end
   if ((c(d2,1)+hl(d2,1))<=(c(d1,1)-hl(d1,1)))
      Zjix=1;
      else Zjix=0; end
   if ((c(d1,2)+hl(d1,2))<=(c(d2,2)-hl(d2,2)))
      Zijy=1;
      else Zijy=0; end
   if ((c(d2,2)+hl(d2,2))<=(c(d1,2)-hl(d1,2)))
      Zjiy=1;
      else Zjiy=0; end

   if (Zijx+Zjix+Zijy+Zjiy>0)
      isover=0;
```

```
        else isover=1; end
end
%%%%%%%%%%%%%%%%%%%%
% Calculate the value of flow cost
function v = costfun(c,cost_flow,flow,N)
    v=0;
        for i=1:1:N
            for j=1:1:N
                if (i~=j)
                    v = v+cost_flow(i,j)*flow(i,j)*(abs(c(i,1)-c(j,1))+abs(c(i,2)-c(j,2)));
                end
            end
        end
end
%%%%%%%%%%%%%%%%%%%%
% Calculate the value of Reward
function r = rewardfun(d,reward_adj,N)
    r=0;k1=1;k2=1;
        for i=1:1:(N-1)
            for j=(i+1):1:N
                if d(i,j,3)==0 adjacency=1;
                else adjacency = 0;end;
                %else adjacency = 1/exp(d(i,j,3));end;
                r = r+reward_adj(i,j)*adjacency;
            end
        end
%disp(r);
end
%%%%%%%%%%%%%%%%%%
```

Matlab Code: Firefly Algorithm

```
function [N,num,L,l,cf]=FLPN_v5_FF()
global g_sol_num;global L; global N; global l; global hl;
global flow; global cost_flow; global reward_adj; global dist;
g_sol_num=0;dist=[];
k1=1;k2=1; %value of co-effecients in the exponential function

% Opening Text file and reading dimensions & initializing respective
% parameters
fid=fopen('input5.txt');
N = fscanf(fid, 'N=%d\n', 1);
L(1) = fscanf(fid, 'L=%d', 1);
L(2) = fscanf(fid, '%d\n', 1);

fscanf(fid, '--sizeofDept--\n', 1);
formatSpec = '%d %d';
sizeA = [2 N];
l = fscanf(fid,formatSpec,sizeA);
l=l';

for p=1:1:N
   hl(p,1)=l(p,1)/2;
   hl(p,2)=l(p,2)/2;
end
% Reading Flow matrix
fscanf(fid, '%*s\n', 1);
formatSpec = repmat('%d', 1, N);
sizeA = [N N];
flow = fscanf(fid,formatSpec,sizeA);
```

```
flow=flow';
% Reading Cost of flow matrix
fscanf(fid, '%*s\n', 1);
formatSpec = repmat('%d', 1, N);
sizeA = [N N];
cost_flow = fscanf(fid,formatSpec,sizeA);
cost_flow=cost_flow';
% Reading Reward matrix
fscanf(fid, '%*s\n', 1);
formatSpec = repmat('%d', 1, N);
sizeA = [N N];
reward_adj = fscanf(fid,formatSpec,sizeA);
reward_adj=reward_adj';
fclose(fid);

ff=20; % number of fireflies
MaxIteration=40; % number of pseudo time steps
% ------------------------------------------------
alpha=0.25;     % Randomness 0--1 (highly random)
betamin=0.20;     % minimum value of beta
gamma=1;        % Absorption coefficient
% ------------------------------------------------

dimensions=zeros(1,2*N); halfLengths=zeros(1,2*N);
i=1;
for p=1:1:N
   dimensions(i)=l(p,1);
   halfLengths(i)=l(p,1)/2;
   dimensions(i+1)=l(p,2);
   halfLengths(i+1)=l(p,2)/2;
```

```
    i=i+2;
end

Ub=zeros(1,2*N);Lb=zeros(1,2*N);

for i=1:2:N*2
    Lb(i)= halfLengths(i);
    Ub(i)= L(1)-halfLengths(i);
end
for i=2:2:N*2
    Lb(i)= halfLengths(i);
    Ub(i)= L(2)-halfLengths(i);
end
noOfVars=N*2;
fbest=2000000;

for m=1:1:200
    [centers]=init_ffa(ff,noOfVars,Lb,Ub);

    objVal=zeros(1,ff);
    for i=1:ff
        objVal(i)=objFunc(centers(i,:));
    end

    for k=1:MaxIteration
        [centers]=ffa_move(ff,noOfVars,centers,objVal,alpha,betamin,gamma,Lb,Ub);
        for i=1:ff
            objVal(i)=objFunc(centers(i,:));
        end
        % Ranking fireflies by their light intensity/objectives
    %sorting of new fire flies
    objVal_temp=objVal;
    [objVal,Index]=sort(objVal_temp);
    centers_temp=centers;
    for i=1:ff,
      centers(i,:)=centers_temp(Index(i),:);
    end

  end
  i=1;
  while (((objVal(i)<fbest) || ((objVal(i)-fbest)< 0.1*fbest)) && objVal(i) ~=
2000000)
    unique=1;
    for j=1:1:g_sol_num
      if isequal(ngbest(j,:),centers(i,:))
        unique=0;
      end %uniqueness check
    end
    if unique==1
      disp(objVal(i));
      g_sol_num=g_sol_num+1;
      ngbest(g_sol_num,:)=centers(i,:);
      tot_zngbest(g_sol_num)=objVal(i);
      fbest=objVal(1);
      i=i+1;
    end
  end

end
fileID=fopen('tempFF.txt','a+');
```

```
fprintf(fileID,'\n %s \n',' -- from file --------');
disp(g_sol_num);
for i=1:1:g_sol_num
  for n=1:noOfVars
     fprintf(fileID,'%2.0f ',ngbest(i,n));
  end
  fprintf(fileID,'\n');
  fprintf(fileID,'%2.0f ',tot_zngbest(i));
  fprintf(fileID,'\n');
end
fprintf(fileID,'\n');
fprintf(fileID,'\n %s \n',' ----- end --------');
fclose(fileID);
fclose('all');
num=g_sol_num;
for i=1:1:g_sol_num
   for j=1:1:N
      cf(i,j,1)= ngbest(i,j*2-1);
      cf(i,j,2)= ngbest(i,j*2);
   end
end
end

%%%%%%%%%%%%%%%%
% Move all fireflies toward brighter ones
function [ns]=ffa_move(n,noOfVars,ns,Lightn,alpha,betamin,gamma,Lb,Ub)
       % Scaling of the system
scale=abs(Ub-Lb);
Lighto=Lightn;nso=ns;
% Updating fireflies
```

```
for i=1:n,
% The attractiveness parameter beta=exp(-gamma*r)
  for j=1:n,
    r=sqrt(sum((ns(i,:)-ns(j,:)).^2));
    % Update moves
    if Lightn(i)>Lighto(j), % Brighter and more attractive
        %disp('in if');
        beta0=1; beta=(beta0-betamin)*exp(-gamma*r.^2)+betamin;
        tmpf=alpha.*(rand(1,noOfVars)-0.5).*scale;
        ns(i,:)=round(ns(i,:).*(1-beta)+nso(j,:).*beta+tmpf);
        %   nsf(i,:)=nsf(j,:);
    end
  end % end for j
end % end for i
end

%%%%%%%%%%%%%%%
function objVal=objFunc(centers)
global N;
for i=1:1:N
   c(i,1)= centers(2*i-1);
   c(i,2)= centers(2*i);
end
create_distance_matrix(c);
isover=0;
for i=1:1:(N-1)
   for j=(i+1):1:N
       isover=isover+ifoverlap(i,j,c);
   end
end
```

```
if isover>0
   objVal=2000000;
else
   alpha=0.5;
   cost=costfun(c);
   reward=rewardfun();
   %v= alpha*cost;
   objVal= alpha*cost-(1-alpha)*reward;
end
end
%%%%%%%%%%%%%%
function create_distance_matrix(c)  % Function that builds the gap matrix
global dist; global N;
   for p=1:1:N
      for q=1:1:N
         dist(p,q,:)=[500 500 500];
      end
   end

   for i=1:1:(N-1)
      for j=(i+1):1:N
         measure_distance(i,j,c);
      end
   end
end

%%%%%%%%%%%%%%%
function measure_distance(i,j,c)   % Function that measures gap between boundaries
of two departments
```

```
   global dist; global hl;
   if ((c(i,1)+hl(i,1))<=(c(j,1)-hl(j,1)))
      dist(i,j,1) = (c(j,1)-hl(j,1))-(c(i,1)+hl(i,1));
    elseif ((c(j,1)+hl(j,1))<=(c(i,1)-hl(i,1)))
      dist(i,j,1) = (c(i,1)-hl(i,1))-(c(j,1)+hl(j,1));
    else dist(i,j,1)=0;
    end
    if ((c(i,2)+hl(i,2))<=(c(j,2)-hl(j,2)))
      dist(i,j,2) =(c(j,2)-hl(j,2))-(c(i,2)+hl(i,2));
    elseif ((c(j,2)+hl(j,2))<=(c(i,2)-hl(i,2)))
      dist(i,j,2) =(c(i,2)-hl(i,2))-(c(j,2)+hl(j,2));
    else dist(i,j,2)=0;
    end
    dist(i,j,3)= dist(i,j,1)+dist(i,j,2);
end
%%%%%%%%%%%%%%%%%
% Calculate the value of Reward
function r = rewardfun()
global dist,global reward_adj,global N;
r=0;k1=1;k2=1;
      for i=1:1:(N-1)
         for j=(i+1):1:N
            if dist(i,j,3)==0 adjacency=1;
            %else adjacency = 0;end;
            else adjacency = 1/exp(dist(i,j,3));end;
            r = r+reward_adj(i,j)*adjacency;
         end
      end
%disp(r);
end
```

```
%%%%%%%%%%%%%%%%%%%%
% Calculate the value of flow cost
function v = costfun(c)
global cost_flow; global flow; global N;
v=0;
   for i=1:1:N
      for j=1:1:N
         if (i~=j)
            v = v+cost_flow(i,j)*flow(i,j)*(abs(c(i,1)-c(j,1))+abs(c(i,2)-c(j,2)));
         end
      end
   end
end

%%%%%%%%%%%%%%%%%%%%
% Check if there is any overlap between two departments
function isover=ifoverlap(d1,d2,c)
global hl;
   if ((c(d1,1)+hl(d1,1))<=(c(d2,1)-hl(d2,1)))
      Zijx=1;
      else Zijx=0; end
   if ((c(d2,1)+hl(d2,1))<=(c(d1,1)-hl(d1,1)))
      Zjix=1;
      else Zjix=0; end
   if ((c(d1,2)+hl(d1,2))<=(c(d2,2)-hl(d2,2)))
      Zijy=1;
      else Zijy=0; end
   if ((c(d2,2)+hl(d2,2))<=(c(d1,2)-hl(d1,2)))
      Zjiy=1;
      else Zjiy=0; end
```

```
    if (Zijx+Zjix+Zijy+Zjiy>0)
        isover=0;
        else isover=1; end
end

%%%%%%%%%%%%%%%%%%%%
% The initial locations of n fireflies (distributions)
function [ns]=init_ffa(ff,noOfVars,Lb,Ub)
  % if there are bounds/limits,
   for i=1:ff,
   ns(i,:)=round(Lb+(Ub-Lb).*rand(1,noOfVars));
   end
end
```