

UNIVERSITY OF BAHRAIN

College of Engineering



Mechanical Engineering Department

Optimum Construction Site Layout Planning using Multi-objective Artificial Bee Colony Algorithm with Levy Flights

A Thesis submitted in Partial Fulfillment of the Requirements for the degree of Master of Science in Engineering Management

Submitted by

Mohamed Abdulla Habib Yahya

**Bachelors of Science in Civil Engineering, University of Bahrain
Kingdom of Bahrain, 2004**

Supervised by

Mehmet P. Saka

**Professor of Civil Engineering
University of Bahrain**

KINGDOM OF BAHRAIN

November, 2013

UNIVERSITY OF BAHRAIN

College of Engineering



Mechanical Engineering Department

Optimum Construction Site Layout Planning using Multi-objective Artificial Bee Colony Algorithm with Levy Flights

A Thesis submitted in Partial Fulfillment of the Requirements for the degree of Master of Science in Engineering Management

Submitted by

Mohamed Abdulla Habib Yahya

Bachelors of Science in Civil Engineering, University of Bahrain
Kingdom of Bahrain, 2004

Supervised by

Mehmet P. Saka

Professor of Civil Engineering
University of Bahrain

KINGDOM OF BAHRAIN

November, 2013

Abstract

Construction site layout planning (CSLP) has been recognized as a critical step in construction planning. The basic function of this process is to find the best arrangement of the temporary facilities according to multiple objectives that may conflicts with each other. Commonly considered objectives are layout cost and safety hazards. The problem is subjected to logical and resource constraints. The formulation of construction site layout planning model as an optimization problem turns out to be nonlinear programming problem where there are conflicting multi-objectives to be achieved. It is shown that the swarm intelligence based meta-heuristic algorithms are quite powerful in obtaining the solution of such hard to solve type of optimization problems. In this study a novel multi objective artificial bee colony (MOABC) via Levy Flights algorithm is proposed. The model takes in account the dynamic nature of the problem where the demand on temporary facilities is changing during various phases of the project life cycle and the presence of obstructions when determining travel distances. It allows the use of facilities of unequal area and the orthogonal rotation of temporary facilities. The performance of the proposed MOABC via levy flights algorithm is verified by solving mathematical benchmark problems, benchmark engineering design problems, and a benchmark CSLP problem. The comparison with most recent meta-heuristic algorithms demonstrated the robustness of the proposed algorithm. The model was also applied to optimize a CSLP problem for a real construction project of a private hospital in the Kingdom of Bahrain.

ملخص الرسالة

الاسم الكامل: محمد عبدالله حبيب يحيى

عنوان الرسالة: أمثل تخطيط لمواقع العمل باستخدام خوارزمية مستعمرة النحل الاصطناعية المتعددة الأهداف مع طيران ليفي

التخصص: ماجستير الإدارة الهندسية

يعتبر تخطيط مواقع العمل خطوة حاسمة في التخطيط للبناء. الوظيفة الأساسية لهذه العملية هي العثور على أفضل ترتيب للمرافق المؤقتة وفقا لأهداف متعددة قد تتعارض مع بعضها البعض. الأهداف التي تؤخذ عادة في الاعتبار هي تكلفة التخطيط والسلامة المهنية. يحد المشكلة عدد من القيود المنطقية و أخرى متعلقة بالموارد. صياغة مشكلة تخطيط المواقع كمسألة أمثلية ينتج عنه مسألة برمجة غير خطية متعددة الأهداف. أكدت الدراسات على فعالية الخوارزميات الميتاهيبروستيكية المبنية على مبادئ ذكاء السرب في إيجاد حل لهذا النوع من المسائل. هذه الدراسة تقدم خوارزمية جديدة و هي مستعمرة النحل الاصطناعية مع طيران ليفي المتعددة الأهداف. النموذج يراعي الطبيعة الديناميكية للمسألة حيث أن الطلب على المرافق المؤقتة يتغير خلال المراحل المختلفة من دورة حياة المشروع. و يميز النموذج حسابه للمسافات المقطوعة بشكل أكثر دقة اخذا في الاعتبار العوائق الموجودة. يسمح النموذج باستخدام مرافق ذات مساحات مختلفة وتدويرها بطريقة متعامدة. تم التحقق من أداء الخوارزمية المقدمة في حل مسائل قياس رياضية ، و مسائل قياس تصميم هندسي ، و مسألة قياس تخطيط المواقع الإنشائية. أظهرت المقارنة مع الخوارزميات المتاهيبروستيكية الحديثة متانة الخوارزمية المقدمة، كما تم تطبيق النموذج لحل مسألة الأمثلية في تخطيط موقع إنشائي حقيقي لمشروع بناء مستشفى خاص في مملكة البحرين .

Acknowledgment

I would like to thank my supervisor, Prof. M. P. Saka for his guidance and support throughout the research work. My thankfulness goes to my family for their patience, encouragement, and support.

Table of Contents

Abstract	ii
Acknowledgment.....	iv
List of tables	viii
List of figures	ix
List of abbreviations	xi
Chapter 1 Introduction.....	1
1.1 Construction Site Layout Planning	1
1.2 Research goals	5
1.3 Research methodology	5
1.3.1 Research topic	5
1.3.2 Literature review	5
1.3.3 Construction site layout planning model	6
1.3.4 Computational experiments and case studies	7
1.3.5 Discussion and conclusion	7
1.4 The main contributions.....	7
1.5 Thesis contents	8
Chapter 2 Literature Review	9
2.1 Single and multi-objective functions.....	9
2.2 Static and dynamic layout problem	10
2.3 Facilities area:	12
2.4 Distances measurement	12
2.5 Optimization algorithms.....	14
2.5.1 Genetic algorithms	14
2.5.2 Particle Swarm Optimization (PSO)	17
2.5.3 Ant Colony Optimization (ACO) Algorithm	19
2.5.4 CAD-based system with GA	21
Chapter 3 Modeling of Construction Site layout Problem	22
3.1 Objective functions:	22
3.2 Dynamic Search:.....	24
3.3 Distance Measurement:	26

3.4 Facilities type and size:	29
3.5 Overlapping constraints	30
3.6 Site boundary constraints:	32
3.7 Tower Crane Access constraints:.....	33
Chapter 4 Multi-objective Artificial Bee Colony Algorithm with Levy flights (MOABC)	35
4.1 History of optimization	35
4.2 Optimization algorithms.....	37
4.3 Bee Algorithm	39
4.3.1 Behavior of honeybees	39
4.4 Artificial Bee Colony (ABC) algorithm	43
4.5 Multi-objective optimization (MOO)	50
4.6 Multi-Objective Artificial Bee Colony algorithms (MOABC)	54
4.7 Levy Flights.....	59
Chapter 5 Computational experiments and Case study.....	62
5.1 Computational experiments.....	62
5.1.1 Test problem ZDT1 function with a convex front.....	62
5.1.2 Test problem ZDT2 function with a non-convex front	63
5.1.3 Test problem ZDT3 function with a discontinuous front.....	66
5.1.4 Sensitivity of control parameters.....	68
5.1.5 Design of a disc brake	73
5.1.6 Design of a cantilever beam	74
5.2 CSLP Case studies.....	77
5.2.1 Case study 1.....	78
5.2.2 Case study 2.....	86
Chapter 6 Conclusion	96
Appendices	98
Appendix A Facilities closeness relationship matrices for case study 1.....	99
Appendix B Facilities closeness relationship matrices for case study 2.....	102
Appendix C MOABC with Levy flights algorithm coded using MATLAB language to solve CSLP problem of case study 2	105
C.1 MOABC with Levy flights algorithm	105
C.2 Get levy flights.....	113

C.3 Application of simple constraints.....	114
C.4 Find the current Pareto	114
C.5 Calculate the fitness	115
C.6 Calculate the fitness of feasible solution only.....	116
C.7 CSLP model of case study 2.....	116
References.....	129

List of tables

Table 3-1 The six-value scale commonly used in industrial facility layout.....	24
Table 5-1 Comparison of error for 1000 iteration	68
Table 5-2 Site facilities and their dimensions of case study 1	79
Table 5-3 Comparison with original site layout of case study 1	84
Table 5-4 Site facilities and their dimensions of case study 2	88
Table 5-5 Proximity relationships for layout cost for phase 1	89
Table 5-6 proximity relationships for safety and environmental concerns for phase 1	89
Table 5-7 Three solutions for CSLP problem for case study 2.....	92
Table 5-8 Comparison with original site layout of case study 2	93

List of figures

Figure 3-1 Continuous dynamic search scheme	26
Figure 3-2 Possible rectangular routes not passing through the obstruction	27
Figure 3-3 Traveling distance measurement.....	28
Figure 3-4 Euclidean distance measurement.....	29
Figure 3-5 Facility alignment alternatives	30
Figure 3-6 Overlapping constraints.....	32
Figure 3-7 Example for boundary constraints	33
Figure 3-8 Tower crane reachable radius.....	34
Figure 4-1 Classification of optimization problems (Yang, 2010b)	37
Figure 4-2 Classification of algorithms (Yang 2010b).....	39
Figure 4-3 Waggle dance of honey bees (Frisch Von 1976)	40
Figure 4-4 Decoding the language of the Bee dance (Barth, 1982).....	41
Figure 4-5 The behaviour of honey bee foraging for nectar (Karaboga, 2005)	43
Figure 4-6 Steps of ABC algorithm (Karaboga 2005)	45
Figure 4-7 Constrained single objective optimization problem.....	47
Figure 4-8 Main body of ABC algorithm (Karaboga and Akay, 2011).....	47
Figure 4-9 Initialization step	48
Figure 4-10 Employed bees algorithm	48
Figure 4-11 Probabilities for onlooker bees.....	49
Figure 4-12 Onlooker bees Algorithm	49
Figure 4-13 Scout bees Algorithm	50
Figure 4-14 A population of five solutions (Deb 2001).....	52
Figure 4-15 Pareto-optimal solution are marked with continuous curves for four combinations of two types of objectives (Deb 2001).....	53
Figure 4-16 Schematic of an ideal multi-objective optimization procedure (Modified from Deb 2001).....	54
Figure 4-17 The flow chart of MOABC with levy flights	58
Figure 4-18 Distribution of 100 consecutive steps (Yang and Deb 2013)	61
Figure 4-19 2D Levy flights in 100 steps (Yang and Deb 2013)	61
Figure 5-1 Pareto front of ZDT1: MOABC with levy flights and the true front.	64
Figure 5-2 Convergence of the proposed MOABC with levy flights for ZDT1.....	64
Figure 5-3 Pareto front of ZDT2: MOABC with levy flights and the true front.	65
Figure 5-4 Convergence of the proposed MOABC with levy flights for ZDT2.....	66
Figure 5-5 Pareto front of ZDT3: MOABC with levy flights and the true front.	67
Figure 5-6 Convergence of the proposed MOABC with levy flights for ZDT3.....	68
Figure 5-7 ZDT1 convergence at four levels of RM	70
Figure 5-8 ZDT2 convergence at four levels of RM	70
Figure 5-9 ZDT1 convergence at four levels of limit	71
Figure 5-10 ZDT2 convergence at four levels of limit	71

Figure 5-11 ZDT1 convergence at four levels of SPP	72
Figure 5-12 ZDT2 convergence at four levels of SPP	72
Figure 5-13 Pareto front for the disc brake design	74
Figure 5-14 Schematic of cantilever beam design (Nakayama , 2005)	76
Figure 5-15 Pareto front for the cantilever beam design	77
Figure 5-16 Original construction site layout for case study 1	82
Figure 5-17 Pareto front for the CSLP problem of case 1	84
Figure 5-18 Construction site layout alternative L1 for case study 1	85
Figure 5-19 Construction site layout alternative L2 for case study 2	85
Figure 5-20 Original construction site layout for case study 2	91
Figure 5-21 Construction site layout alternative L1 for case study 2	93
Figure 5-22 Pareto front for the CSLP problem of design example 2	94
Figure 5-23 Construction site layout alternative L2 for case study 2	94
Figure 5-24 Construction site layout alternative L3 for case study 2	95

List of abbreviations

ABC	Artificial bee colony
ACO	Ant colony optimization
BC	Boundary constrain
CSLP	Construction site layout planning
GA	Genetic algorithm
GDEA	Generalized data envelopment analysis
HBA	honeybee algorithm
HBMA	honeybee-mating algorithm
MCN	Maximum cycle number
MMAS	Max-min ant system
MOABC	Multi-objective artificial bee colony
MOCS	Multi-objective cuckoo search algorithm
MOD	Minimum obstruction distance
MOFA	Multi-objective firefly algorithm
MOO	Multi-objective optimization
MOPSO	Multi-objective particle swarm optimization
MR	Modification rate
NP	Size of population
NSGA II	Elitist non-dominated sorting genetic algorithm
PF ^e	Estimated pareto front
PF ^t	True pareto front
PSO	Particle swarm optimization
QAP	Quadratic assignment problem

sn	Number of solutions
SPP	Scout production period
VBA	Virtual bee algorithm

Chapter 1 Introduction

1.1 Construction Site Layout Planning

Construction site layout planning (CSLP) has been recognized as a critical step in construction planning. The basic function of this process is to find the best arrangement of the temporary facilities in the available space in order to improve productivity, safety and work environment at construction site. Depending on the size and type of the construction project, necessary temporary facilities vary in number and size. Medium to large scale projects commonly includes, site offices, labor rest room, material laydown area, fabrication yard, carpentry workshop, tools shed, tower crane and material hoist. During the planning phase, CSLP decision making process involves determining CSLP objectives, identification of the site facilities and the available site space, generation of the construction site layout alternatives, which fulfill the layout objectives and constraints, and finally evaluating and selecting the best site layout plan for implementation (Ning et al. 2011).

The project manager or planner usually performs the task of preparing the site layout based on his/her knowledge and expertise (Osman et al. 2003) This approach usually produces a single construction site layout alternative that satisfies the layout constraints which is so called a feasible solution, but not necessary to be an optimal solution. However, there is a huge number of feasible alternatives. Moreover, when considering the fact that CSLP problem is multi-objective problem, instead of single optimal solution, the optimal solution consist of numerous trade-offs.

CSLP can be broadly classified into two types, continuous and discrete layout problems (Liggett 2000) Formulating continual layout problem allows locating facilities everywhere in the planar site subject to satisfying relative constraints (Osman et al. 2003). Such constraints typically include boundary constraints and constraints to prevent overlapping of temporary facilities with each other or with other permanent facilities or structures. In the discrete layout problem, 'n' facilities are to be located in 'n' predetermined locations (Jiuping Xu and Zongmin Li 2012, Li and Love 2000, Lien and Cheng 2012, Zhang and Wang 2008). As the continual layout formulation is computationally rigorous because it increases the potential locations for placing the facilities exponentially, many researchers considered the discrete formulation that can be simplified to quadratic assignment problem (QAP).

Several research works treated CSLP as a single objective problem that intends to minimize the cost associated with interactions between facilities, minimize travel distance, or minimize layout cost (Li and Love 2000, Zhang and Wang 2008, Osman et al. 2003). These algorithms neglect other significant aspects taken into account by practitioners in the actual situation, such as safety and noise control. However, such qualitative criteria have been taken in consideration in recent research works. Accordingly, researchers proposed multi-objective algorithms that typically minimize the cost associated with the interaction between facilities and minimize safety and environmental hazards (Jiuping and Zongmin 2012, Ning et al. 2010, Khalafallah and El-Rayes 2011).

Most of previous research considered CSLP problem as static layout, assuming that all temporary facilities are assembled at the beginning and last till completion of the

project, for example (Li and Love 2000, Cheung et al. 2002, Yeh 1995, Lien and Cheng 2012). However in the real situation, the need of temporary facilities varies during different construction phases of the projects and basically depends on activities schedules. Because of its significant effects on the reliability of the results, most recent studies consider the dynamic search scheme for solving the CSLP problem (Jiuping and Zongmin 2012), (Ning et al. 2010).

To simplify the complexity of CSLP problem, former studies assume that temporary facilities have the same size which are referred to as equal-area problems. Later studies acknowledged that the representation of the problem as an equal-area problem limited the utility of the algorithm and proposed alternate models that take into account the actual size of temporary facilities. These models are referred to as unequal-area problem.

In a construction site layout that is composed of multiple facilities, the distance between those facilities is a major parameter in evaluating the effectiveness of that layout. The evaluation of costs related to interaction flows between facilities involves determination of travel distances. Several researchers have measured the travel distances using Euclidean distances or rectangular distances without considering the presence of obstructions (Jiuping and Zongmin 2012), (Ning et al. 2010), and (Lien and Cheng 2012). However, such simplifications would affect the ability of proposed models in representing the actual situation and the reliability of their results. Regarding safety and environmental issues researchers use Euclidean distances that precisely represent the actual situation.

Obtaining accurate values for actual inter-facility transportation can become quite difficult (Osman et al. 2003) which promote the use of proximity weights to reflect the required closeness based on interactions between facilities.

Construction site layout planning is considered as 'NP-hard' problem (Lien and Cheng 2012) for its complexity. The recent meta-heuristic algorithms based on swarm intelligence have demonstrated their power in finding the solution of such type of optimization problems (Saka and Dogan 2012). This has encouraged researchers to employ these modern meta-heuristic algorithms to determine the solution of their proposed CSLP models. Ning et al. (2010) in their work proposed a method to solve dynamic multi-objective CSLP problem using max-min Ant system (MMAS) which is one of the versions of standard ant colony optimization (ACO) algorithms. Lien and Cheng (2012) proposed a hybrid swarm intelligence based particle-bee algorithm for construction site layout optimization under single objective function to locate facilities in predetermined locations. Li and Love (2000) presented an investigation of applying the Genetic Algorithm to attain the optimal solution for single objective CSLP problem to accommodate facilities of un-equal area in predetermined locations. Zhang and Wang (2008) proposed a particle swarm optimization (PSO) based methodology. They modeled CSLP problem to optimize static layout under single objective function to accommodate facilities of un-equal area in predetermined locations. Another study related to particle swarm optimization (PSO) was developed by Jiuping and Zongmin (2012). Their approach uses multi-objective particle swarm optimization (MOPSO) algorithm. The approach was applied to solve multi objective dynamic CSLP problem. Osman et al. (2003) proposed a hybrid cad-based algorithm using genetic algorithm

(GA) in order to optimize the assignment of un-equal area facilities to any unoccupied space at construction site.

1.2 Research goals

The primary objective of the research carried out in this thesis is to develop a multi-objective optimization model to solve constructions site layout planning problem that:

1. Minimizes the total handling cost of interaction flows between facilities and minimizes safety hazards/environmental concerns
2. Employs a continuous dynamic searching scheme
3. Uses more accurate approach to represent the actual travel distances between facilities.
4. Allows unequal area facilities to be aligned vertically or horizontally
5. Considers logical and resources constraints

1.3 Research methodology

In order to fulfill the objectives of the research, the following procedure and methods were employed.

1.3.1 Research topic

The research topic was decided based on investigating both recent research and common practices in the field of construction management.

1.3.2 Literature review

Comprehensive literature review was conducted in the recent research developments in the field of construction site layout planning problem and in the field of modern meta-heuristic algorithms.

1.3.3 Construction site layout planning model

The construction site layout planning problem was formulated as dynamic layout problem that takes into account the temporary existence of facilities among various construction project phases. The model is intended to optimize two objective functions of minimizing the safety hazards /environmental concerns and the total handling cost of interaction flows among facilities.

When an optimization problem involves more than one objective function it is referred as multi-objective optimization problem. Most real-world search and optimization problems naturally involve multiple objectives (Deb 2001). The objectives are often conflicting, consequently, the optimal solution consists of numerous trade-offs.

The proposed model regards CSLP problem as non-linear layout problem with un-equal area facilities that can be aligned horizontally or vertically. Furthermore the model takes into account the presence of obstructions for determining travel distances.

A novel optimization algorithm was developed which is based on Artificial Bee Colony (ABC) algorithm. ABC is one of the most recent swarm intelligence based algorithms introduced by Dervis Karaboga (2005) which mimics foraging behavior of honey bees.

Meta-heuristic algorithms use a combination of randomization and local search. They are regularly based on learning from nature or biological systems. Some examples of nature-inspired meta-heuristic algorithms are ant and bee colony algorithms, genetic algorithms, cuckoo search algorithm, firefly algorithm, harmony search algorithm, bat

algorithm, and particle swarm optimization. Meta-heuristic algorithms demonstrated their power in solving complex optimization problems (Yang 2010a).

In this study a multi objective artificial bee colony (MOABC) algorithm is developed to determine the solution of CSLP problem. The standard MOABC algorithm is enhanced with Levy flights type of random walks for finding new food sources which is conducted by employed bees.

1.3.4 Computational experiments and case studies

In order to verify the efficiency of the developed model, it was demonstrated on three types of benchmark problems which are mathematical, engineering, construction site layout planning problems. The model was also applied to solve CSLP problem for a real construction project of private hospital in the kingdom of Bahrain.

1.3.5 Discussion and conclusion

The outcomes of the proposed model that employs MOABC with Levy Flights were compared with a developed model that employs the standard MOABC, and with previous research results that employed max-min Ant system (MMAS) based method which is one of the ant colony optimization (ACO) algorithms (Ning 2008). Results show that MOABC via Levy Flights performs better than the mentioned algorithms. Using levy flights by employed bees has improved the efficiency of MOABC algorithm.

1.4 The main contributions

Firstly, alternatives generated by the proposed CSLP model are more practical due to its feature of measuring travelling distances in a manner that represents the actual situation more realistically. Secondly, the proposed algorithm allows vertical

and horizontal alignment of temporary facilities. Thirdly, it considers CSLP problem as continuous search problem rather than simplified assignment problem. Finally, it gives the option to maintain minimum distances between the facilities and between facilities and buildings.

1.5 Thesis contents

The outline of the thesis is organized as follows.

1.4.1 Introduction (Chapter 1).

1.4.2 Literature Review on Optimization of Construction Site Layout Planning (Chapter 2).

1.4.3 Artificial Bee Colony Algorithm with Levy Flights (Chapter 3).

1.4.4 Optimization Models Formulation and Development of Supporting Software (Chapter 4).

1.4.5 Computational experiments and case studies (Chapter 5).

1.4.6 Conclusion (Chapter 6).

Chapter 2 Literature Review

This chapter presents a comprehensive literature review on the construction site layout planning problem exploring and analyzing developed modeling approaches and optimization algorithms in the relevant research studies. This Chapter summarizes and organizes the reviewed literature in three main sections: (1) single and multi-objective functions; (2) static and dynamic layout problem; (3) facilities area; (4) Distances measurement; (5) Optimization algorithms.

2.1 Single and multi-objective functions

In regards to the number of objective functions being considered, the optimization models are classified into two types; a single-objective optimization, where only one objective function is involved; and multi-objective optimization, where more than one objective function are involved.

Most efforts in the construction site layout planning problem have been made to develop single-objective optimization algorithms to minimize cost oriented objective function. However, just like most of real-world optimization problems, CSLP problem is multi-objective by nature.

Li and Love (1998) modeled CSLP problem with single objective function of minimizing the total travel distance. Zouein et al. (2002) Mawdesley et al. (2002), Lien and Cheng (2012), and Zhang and Wang (2008) modeled CSLP problem with single objective function of minimizing layout cost.

Elbeltagi et al. (2004), Sanad et al. (2008), and , Ning et al. (2010), Jiuping and Zongmin (2012) modeled CSLP problem with two objective functions of minimizing layout cost and improving safety level at construction site.

2.2 Static and dynamic layout problem

The necessity of facilities in construction site is temporary by nature with respect to time of need. For example a temporary facility necessary at foundation works phase could be no longer required at finishing works phase. Previous research works dealt with that aspect by two main approaches so called static and dynamic layout.

For simplification purposes most of previous studies considered the static layout. The static layout ignores the temporal nature of the CSLP problem and assumed that all temporary facilities are required from first day of construction till completion date of the project.

Zouein and Tommelein (1994) defined dynamic layout construction as the process of creating a sequence of layouts to span the duration of construction of a project when given the activity schedule.

The dynamic layout approach can be classified into two searching schemes, discrete searching scheme and continuous search scheme. The major differences between the two schemes are described as follows. In the continuous search scheme, the temporary facilities will not be relocated during the various project phases. Accordingly, the arrangement of facilities for the whole construction period can be positioned on a single site layout. However, in the discrete searching scheme the locations of temporary facilities in one phase are independent to their locations in the other

phases. This searching scheme involves the re-handling cost which results in more labor cost and time consumption. Moreover, the relocation of the facilities during their demand period will undesirably affect their availability and would cause interruption to ongoing activities. For these reasons, the continuous searching scheme is more favorable.

Elbeltagi et al. (2004) proposed a continuous dynamic search scheme. In their work to determine the needed temporary facilities on site in any time interval, a five-step approach was used: (1) necessary temporary facilities must be identified and sized; (2) a schedule of the construction process must be assembled; (3) the activities' requirements of the temporary facilities are defined, similar to the requirements of labor, equipment, and other site-specific resources. In this sense, the selected temporary facilities are dealt with as resources that are assigned to activities; (4) the service times for each temporary facility are then determined from the schedule; (5) the temporary facilities that serve any time interval can be defined and considered as the needed temporary facilities on site in this interval.

In order to solve the dynamic CSLP problem, Ning et al. (2010) emphasized that the best site layout should suit the whole construction period with the consideration of the interactive relationships in the different construction phases, as well as the facilities used in each phase. Therefore, a continuous dynamic searching scheme, rather than a discrete dynamic searching scheme was employed in their work.

2.3 Facilities area:

The construction site layout problem can be classified into equal-area facilities and unequal-area facilities. This classification is highly dependent on assumptions regarding the possible locations to assign the facilities. Equal-area facilities approach is used when the facilities to be assigned in predetermined locations having that those predetermined locations can accommodate temporary facility of any size as followed by Li and Love (1998), Cheung et al. (2002), Jiuping and Zongmin (2012).

Unequal-area facilities approach is used in the two following situations. When only some predetermined locations can accommodate some of the temporary facilities as followed by Li and Love (2000), Zang and Wang (2008). Unequal-area approach is also employed when no predetermined locations are identified and the facilities can be assigned in any unoccupied space within the boundary lines of construction site as followed by Osman et al. (2003), Elbeltagi et al. (2004), Ning et al. (2010).

2.4 Distances measurement

In construction site layout problem, travel distances between facilities are highly significant element in determining various objective functions. In layout optimization problems, distances are generally measured using either Euclidean distance or rectangular distance. Euclidean distance between two facilities defined as the square root of the sum of the squares of the differences between the corresponding coordinates of the Cartesian coordinates of their centroids. In other words it is the length of a straight line connecting Cartesians coordinate of their centroid.

Mathematically Euclidean (e) distance between two facilities, facility A and facility B, is represented as follows

$$e_{AB} = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \quad (1)$$

where (x_i, y_i) and (x_j, y_j) are the Cartesian coordinates of the centroids of facility i and j respectively

x_1 is the X coordinate of the centroid of facility A

x_2 is the X coordinate of the centroid of facility B

y_1 is the Y coordinate of the centroid of facility A

y_2 is the Y coordinate of the centroid of facility B

Rectangular distance also referred as Manhattan distance, between two facilities is the distance between the Cartesian coordinates of their centroids measured along axes at right angles that is based on a strictly horizontal and/or vertical path. The mathematical representation of the rectangular distance (R) between two facilities, facility A and facility B, is represented as follows:

$$R_{AB} = (|x_1 - x_2| + |y_1 - y_2|) \quad (2)$$

Hegazy and Elbeltagi (1999), Li and Love (2000), Osman et al. (2003), and Ning et al. (2010), and most of earlier research works in construction site layout planning problems have measured the travel distance between facilities using Euclidean distances. Sanad et al. (2008) acknowledged the deficiency of Euclidean distance approach in representing the actual travel distances between temporary facilities, and

consequently propose a new approach so-called actual rout distance where a network of continuous internal routes must be specified. Another attempt to improve the representation of actual operations and resource movements on site by Lien and Cheng (2012) measured the travel distance between locations using the rectangular distance approach. The major limitation of using rectangular distance approach is neglecting the presence of obstructions such as buildings under construction or temporary facilities.

2.5 Optimization algorithms

2.5.1 Genetic algorithms

Genetic algorithms (GA) are the most widely applied evolutionary algorithms. They have been applied to most optimization problems in various domains including the sciences, engineering, and commerce. Developed by John Holland in the 1970s (Holland, 1975), Genetic algorithm is a model of biological evolution based on Charles Darwin's theory of natural selection. Subsequently, several variants of genetic algorithms have been developed, however, genetic algorithms typically uses crossover or recombination, mutation, selection and certain elitism.

Li and Love (1998) presented an investigation of applying the genetic algorithm to attain the optimal solution for construction site layout planning problem. In their work, the CSLP problem was modeled with single objective function of minimizing the total travel distance. The model assumed that the number of facilities is equal to the number of predetermined locations. The model was employed to solve CSLP problem of 11 facilities of equal area which have to be assigned into 11 predetermined locations. The string form of GA optimization model in this work represented the facility number by the

position of the cell, and represented the location of the facility by the value of the cell. The results showed that GA is robust in searching the optimal solution of the modeled CSLP problem. The major model limitation is the assumption of defining predetermined locations to accommodate the temporary facilities.

In 2000 Li and Love have modified the GA model to solve CSLP problem with facilities of unequal-area. In comparison with equal-area facility layout problem, the results from unequal-area layout problem achieved slightly higher objective function values, indicating the presence of some effect on the layout solution. The study demonstrated the robustness of the GA approach in solving layout problem as combinatorial optimization problem and indicated that the strength of the GA system lies in its ability of locating the global optimum using a random yet directed searching operator. Therefore, the GA system is less likely to restrict the search to a local optimum. Moreover, the study indicated that the GA system converged earlier with a medium sized population.

Hegazy and Elbeltagi (1999) made another investigation of using GA to solve CSLP problem under the objective function of minimizing the total travel distance. In this model, a facility is represented as a group of unit areas that can take any user-specified shape. The model accepted any user-specified site shape and incorporated a flexible GA procedure for the optimum placement of facilities. Based on their proposed model, Hegazy and Elbeltagi developed a comprehensive system for site layout planning so-called (EvoSite). EvoSite uses an intuitive spreadsheet representation of the site and the facilities, and automates the evolution of layout solutions.

Elbetagi and Hegazy (2001) improved the basic model to add practicality and supplement the user's subjective judgment. Accordingly, three main components have been incorporated into the basic model to form a hybrid system, they are (1) a knowledge base for facility identification and area determination, (2) a fuzzy quantifier to address the vagueness in the assessment of facilities' interrelationships, and (3) a modified GA to speed the search for an optimal layout solution.

Based on their previous works, Elbetagi et al. (2004) proposed a further improvised GA model that takes into account the dynamic nature of site layout planning. The model consider the reorganization of the location and/or areas of the needed facilities on site at various schedule intervals, along project duration to suit the activities' dynamic need of temporary facilities. Another novelty of that model is to optimize for both improved safety and minimum layout cost.

Zouein et al. (2002) employed a genetic approach for solving the construction site layout problem of unequal-area facilities. The problem involved coordinating the use of limited site space to accommodate temporary facilities so that transportation cost of materials is minimized. The problem was characterized by affinity weights used to model transportation costs between facilities and by geometric constraints that limit their relative positions on site. The GA was tested on different sets of problems to evaluate the algorithm's performance as the total object-to-site-area ratio was varied in the following cases: equal size with equal weight objects, unequal size with unequal weight objects, and 2D constraints between objects. The results indicated that the proposed GA could be used to solve different classes of layout problems and is expected to return

good solutions. However, the results indicated the GA performed better in loosely constrained problems with a small total-objects-to-site-area ratio.

Mawdesley et al. (2002) investigated the capability of GA to solve CSLP problem under the objective function of minimizing the layout cost. In their formulation, the changing nature of the site has been modeled by enabling the system to produce different site layouts at different phases of construction and assigning removal costs to temporary facilities. Results indicated that the technique is able to find solutions to the problems in a small number of generations and hence a reasonably small computational time.

Sanad et al. (2008) also tested the GA application for solving the site layout planning problem considering safety and environmental issues and actual distance between facilities. The performance of the model was evaluated using a real-life construction project. The obtained results proved that satisfactory solutions were obtained.

2.5.2 Particle Swarm Optimization (PSO)

The particle swarm optimization (PSO), developed by Kennedy and Eberhart (1995), is a population based algorithm inspired by swarm behavior such as fish schooling and bird flocking in nature. Particle swarm optimization and genetic algorithms similarly starts with initial random solutions and searches for optima by updating generations. However, PSO is simpler because it does not have evolution operators such as crossover and mutation. Instead, it uses the real-number randomness and the global communication among the swarm particles. Because of its simplicity particle

swarm optimization has been widely applied to optimization problems, several PSO variants have been developed, combined with other existing algorithms to generate more efficient hybrid algorithms.

Zhang and Wang (2008) firstly formulated a particle swarm optimization (PSO-based) methodology to solve the construction site unequal-area facility layout problem. The model employed a priority-based particle representation of the candidate solutions to the layout problem. The particle-represented solution in terms of priorities was transformed to the specific layout plan with consideration of non-overlap and geometric constraints. Moreover, the model proposed a modified solution space boundary handling approach for controlling particle updating with regard to the priority value range. The PSO model intended to determine an optimal set of priority values to minimize the total cost of the resultant layout based on the priority values. The model assumed the temporary facilities to be allocated in predetermined locations. Computational experiments were presented to justify the PSO-based methodology for solving the construction site unequal-area layout problem and the performance of the PSO-based method was compared with that of a GA-based method. The model was employed to solve CSLP problem of 11 facilities of unequal area which have to be assigned into 11 predetermined locations.

The results demonstrated that the PSO-based method requires less iteration to find optimal solutions, and is more efficient than the GA method implemented using the same fitness function, and the modified GA operators of Li and Love (2000).

Jiuping and Zongmin (2012) proposed multi-objective particle swarm optimization algorithm (MOPSO) to solve the construction site layout planning problem. They considered the fact that planning problems are dynamic, multi-objective and uncertain in nature. Aiming to enhance the general practice of dynamic construction site layout planning, this work employed a fuzzy random multi-objective decision making model. In this model, two objectives were considered: (1) minimizing the total cost of site layout; and (2) maximizing the distance between the 'high-risk' facilities and the 'high-protection' facilities to reduce the possibility of safety or environmental accidents.

The MOPSO approach used the concept of pareto dominance to determine the flight direction of a particle and it maintains previously found non-dominated vectors in a global repository that is later used by other particles to guide their own flight. This method provides pareto-optimal solutions allowing decision makers to choose their preferred plan among these non-dominated alternatives.

The approach was applied to a large-scale water conservancy and hydropower construction project. The results demonstrated the effectiveness of the model. The major model limitation is the assumption of defining predetermined locations to accommodate the temporary facilities.

2.5.3 Ant Colony Optimization (ACO) Algorithm

Developed by Marco Dorigo (1992) Ant colony optimization algorithm is one most popular ant algorithms that mimics the foraging behavior of ants colony. The foraging routes in combinatorial optimization are marked by pheromone concentration deposited by the ants. the ants will follow the route with higher pheromone concentration, and

pheromone is enhanced by the increasing number of ants. Accordingly, some favorite routes emerge, often the shortest or more efficient. The most important issues are the probability of choosing a route and the evaporation rate of pheromone. Scientists have developed several ACO algorithms in the various fields and several ACO variants have been developed.

Samdani et al. (2006) developed a CSLP Ant Colony Optimization (ACO) algorithm to solve the problem in which n facilities are to be positioned to n available sites. The model intended to minimize the total cost of construction and interactive cost due to facility layout constraints. The efficacy of the ACO model was demonstrated on a 12 facility problem. The results showed that the proposed model is efficient in producing feasible low-cost solutions. The major model limitation is the assumption of defining predetermined locations to accommodate the temporary facilities.

Lam et al. (2007) made another investigation of using ACO algorithm to resolve construction site layout problem in a hypothetical medium sized project. The objective function aimed to minimize the sum of the interaction flows between facilities. The study applied fuzzy reasoning and entropy technique to calculate the closeness relationship between facilities. The results demonstrated that the ACO algorithm is feasible and efficient method of optimizing and solving site layout problem.

Ning et al. (2010) proposed a new method using continuous dynamic searching scheme to guide the max-min ant system (MMAS) algorithm, which is one of the ant colony optimization (ACO) algorithms. The system aimed to solve the dynamic CSLP problem under the two congruent objective functions of minimizing safety concerns and

reducing construction cost is proposed. Using weighted sum method the MOO problem was solved by the proposed MMAS method. An office building case was used to verify the capability of the proposed method to solve dynamic CSLP problem and the results were promising.

2.5.4 CAD-based system with GA

Osman et al. (2003) developed a hybrid CAD-based construction site layout planning system using genetic algorithms. It utilizes genetic algorithms (GAs), as function optimizers, in determining the temporary facility locations according to the graphical information depicted in a CAD environment. The CAD environment was utilized in space detection of the site layout and in the satisfaction of geometrical constraints dictated by the facility assignment problem. The optimization goal was to minimize the total transportation costs (or its proximity weight equivalents) between facilities. The research concluded that incorporating GA optimization engine with a CAD-based input/output media greatly increased its practicality for use.

Chapter 3 Modeling of Construction Site layout Problem

The proposed construction site layout planning layout model is designed to arrange temporary facilities required at various time intervals (phases) of any construction project where a set of restraints is present and several objectives are required to be achieved. Generally in multi-objective optimization problems pareto optimal solutions are determined so that decision makers can choose their preferred plan among these pareto optimal solutions (Jiuping and Zongmin 2012). The efficiency of CSLP model will be significantly affected by its precision in representing the actual situations.

In this research the construction site layout planning problem has been modeled to be more representative to the actual situation. A comprehensive description of the components of the proposed model is given in seven main sections: (1) The objective functions that the model intends to optimize; (2) the dynamic search approach to reflect the facilities time of need, in another words, at which construction phase a particular facility is required; (3) measurement approaches for travel distance and safety related proximity between temporary facilities; (4) facilities type and size; (5) overlapping constraints; (6) site boundary constraints; (7) tower crane access constraints.

3.1 Objective functions:

Two objective functions are considered in the model. First objective function is to minimize the total handling cost of interaction flows between facilities. The second is to minimize safety hazards/environmental concerns. Minimizing cost is the most important

objective function in most layout planning problems and the focal point of numerous previous research on CSLP problems. Focusing on quantitative objectives such as minimizing the layout cost and neglecting qualitative issues like safety and environmental concerns can increase the likelihood of accidents and environmental hazards. Accordingly, recent researchers emphasized the importance of considering improving safety in addition to cost minimization objective functions (Elbeltagi et al. 2004), (Sanad et al. 2008), (Ning et al. 2010), and (Jiuping and Zongmin (2012). The objective functions, in this research, are mathematically defined as follows:

$$\text{Min } f_1 = \sum_{p=1}^{np} \sum_{i=1}^n \sum_{j=1}^n C_{ijp} r_{ij} P_{ijp} \quad (\text{Eq. 3-1})$$

$$\text{Min } f_2 = \sum_{p=1}^{np} \sum_{i=1}^n \sum_{j=1}^n SE_{ijp} \frac{1}{e_{ij}} P_{ijp} \quad (\text{Eq. 3-2})$$

where (x_i, y_i) and (x_j, y_j) are the Cartesian coordinates of the centroids of facility i and j respectively which are treated as design variables. e_{ij} is the Euclidean distance between centroids of facilities i and j . r_{ij} is the modified rectangular distance between centroids of facilities i and j that considers the presence of obstruction. Both Euclidean and modified rectangular distances will be described later. C_{ijp} represents either the actual transportation cost between facilities i and j at phase p taking into consideration the number of trips made, or a relative proximity weight that reflects the required closeness based on interactions between facilities i and j at phase p . P_{ijp} represents a binary matrix to indicates presence of both facilities i and j at phase p . Obtaining accurate values for actual inter-facility transportation can become quite difficult (Osman et al. 2003) which promotes the use of proximity weights (Ning et al. 2010), (Osman et

al. 2003). Table 3-1 shows a common scale used in industrial facility layout problems (Askin and Standridge 1993). Section 5.2.2.3 presents an example of using this approach.

SE_{ijp} represents closeness relationship values for safety and environmental concerns. In this study, as shown in equation (Eq. 3-2), if two facilities A and B have with safety hazard at phase p, SE_{ABp} , f_2 is minimized by increasing the Euclidean distance e_{AB} .

Table 3-1 The six-value scale commonly used in industrial facility layout (Askin and Standridge, 1993)

Desired relationship between facilities	Proximity weight
Absolutely necessary (A)	81
Especially important (E)	27
Important (I)	9
Ordinary closeness (O)	3
Unimportant (U)	1
Undesirable (X)	0

3.2 Dynamic Search:

A continuous dynamic searching scheme (Elbeltagi et al. 2004) rather than discrete dynamic search is employed in this work to minimize the total values of

objective functions during whole construction periods (Ning et al. 2010). In continuous dynamic searching scheme, locations of facilities will be the same during the different construction phases i.e. there will be a single site layout for the whole construction period. On the contrary, in discrete dynamic searching scheme the layout of each phase is independent to the other phases. For example if in phase 1 the steel yard is located at location A, in phase 2 the model will search for a new location to the steel yard facility without any consideration to its location in phase 1 and may locate another facility at location A. Accordingly a re-handling cost i.e. more labor cost and time consumed is incurred when discrete dynamic searching scheme is used. Another disadvantage of this searching scheme is acute when a relocated facility became out of use causing interruption to ongoing activities depending on that facility. Figure 3-1 illustrates the continuous dynamic search scheme for locating four temporary facilities in a hypothetical site with three phases. Facility A is required in the three construction phases, facility B is needed in phases 1 and 2, facility C is needed in phase 2 only, and facility 4 is needed in phases 2 and 3. Figure 3-1 also illustrates that the optimum layout, when using continuous dynamic searching scheme, can be represented in single plan. In this searching scheme the location of any facility is determined from optimizing the layout in all phases together. For example, when minimizing for single cost based objective function, the location of facility B is determined to have minimum layout cost in all phases and not only for phase 1 which is the time facility B is needed.

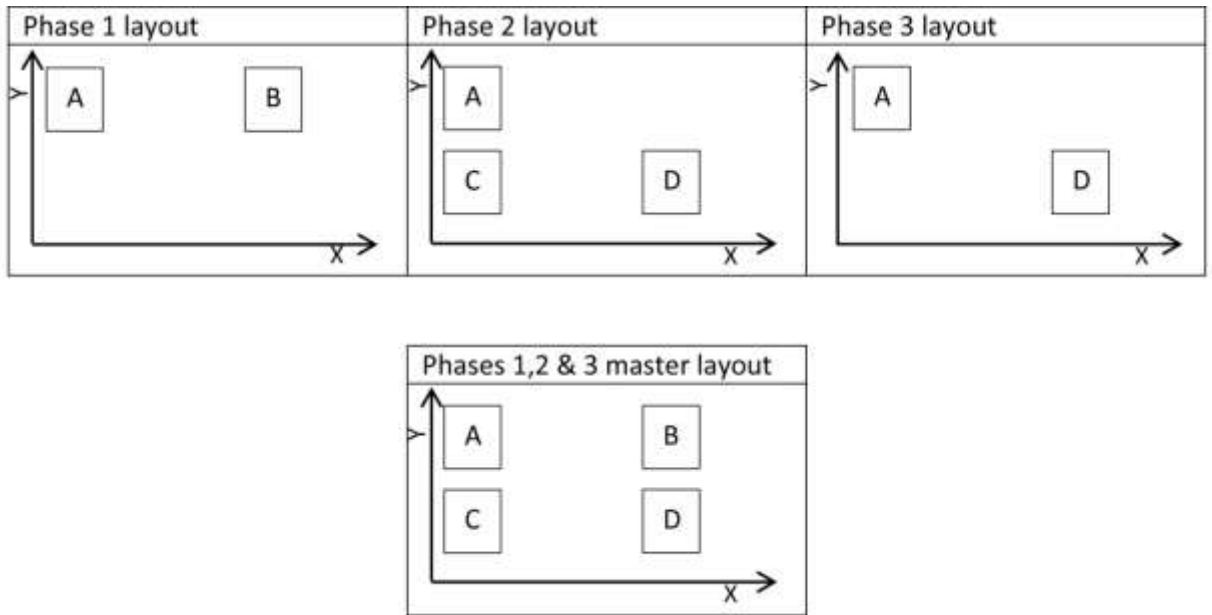


Figure 3-1 Continuous dynamic search scheme

3.3 Distance Measurement:

For determining the first objective function that minimizes total handling cost of interaction flows between facilities this study proposes more accurate approach to represent the actual travel distances between facilities. Neglecting the presence of obstruction can significantly affect the reliability of the results. The rectangular distance is modified to consider the presence of obstructions. Obstruction can be a building under construction or any other facility.

In the modified rectangular distance measurement approach, if an obstruction exists in the rectangular route between two facilities, the route is modified to avoid passing through the obstruction via the shortest path. For example, in Figure 3-2 there are four possible routes to avoid passing through the obstruction. The first route A, c, d, e, f, B is the shortest among the four routes, therefore the proposed model measures the travel distance between Facility A and Facility B by calculating the length of route A,

c, d, e, f, B. The distance added to the basic rectangular distance to prevent passing through the obstruction is referred here as “obstruction distance”. The modified rectangular distance R_{ij} is the result of adding the minimum obstruction distance (MOD) to the basic rectangular distance and mathematically represented as follows

$$r_{ij} = (|x_i - x_j| + |y_i - y_j|) + MOD \quad (\text{Eq. 3-3})$$

where (x_i, y_i) and (x_j, y_j) are the Cartesian coordinates of the centroids of facility i and j respectively, and MOD is the minimum obstruction distance.

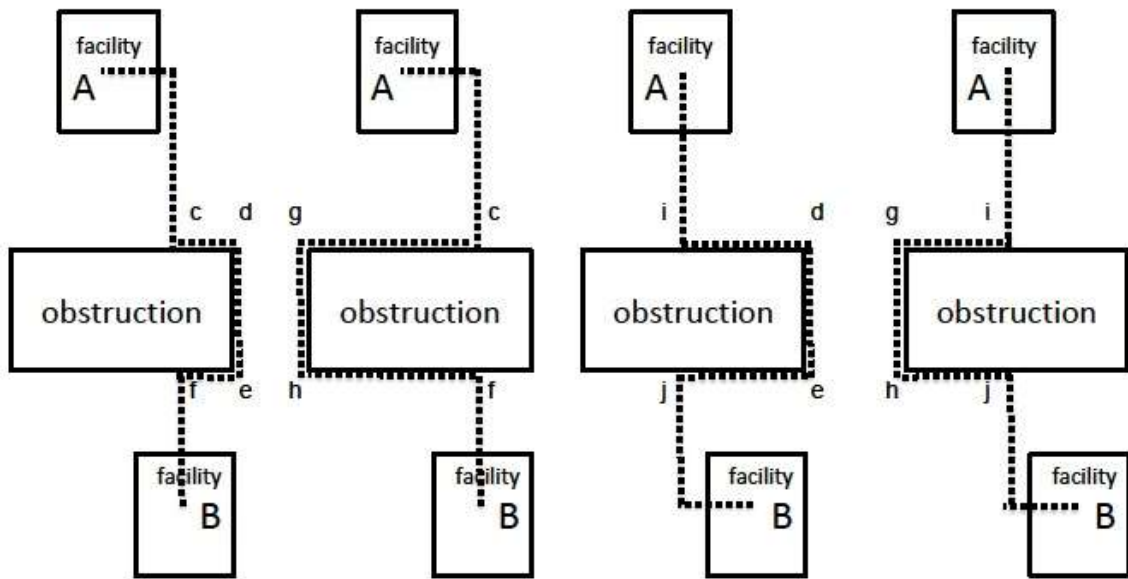


Figure 3-2 Possible rectangular routes not passing through the obstruction

Figure 3-3 shows both the modified and basic approach for calculating rectangular travel distance between facilities A and B.

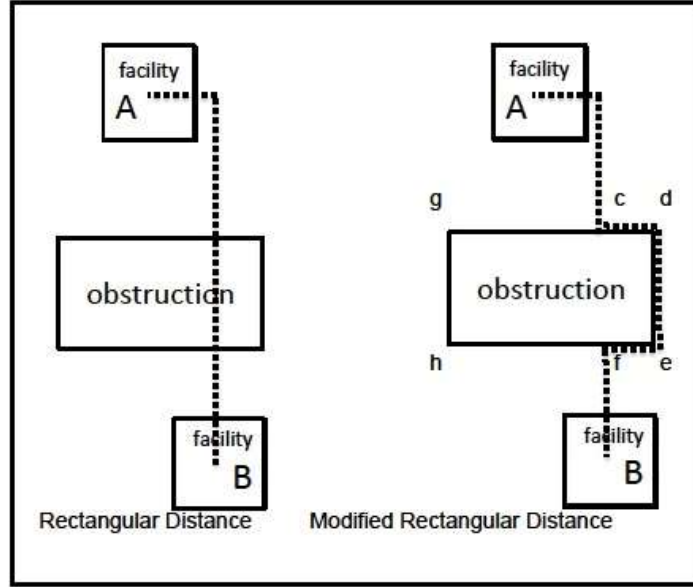


Figure 3-3 Traveling distance measurement

For determining the second objective function that minimizes safety hazards /environmental concerns the Euclidean distance is considered as the safety and environmental measure depends on closeness of facilities rather than travel distance.

Euclidean distance between two facilities is defined as the square root of the sum of the squares of the differences between the corresponding coordinates of the Cartesian coordinates of their centroids. In other words it is the length of a straight line connecting Cartesian coordinate of their centroid as illustrated in Figure 3-4

Mathematically Euclidean (e) distance between two facilities, is represented as follows

$$e_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \quad (\text{Eq. 3-4})$$

where (x_i, y_i) and (x_j, y_j) are the Cartesian coordinates of the centroids of facility i and j respectively

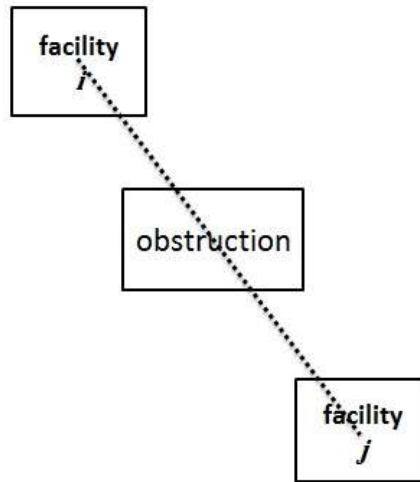


Figure 3-4 Euclidean distance measurement

3.4 Facilities type and size:

Fixed facilities are those facilities whose locations are pre-determined and fixed during the process of assigning facilities to free locations. For example, the security hut is commonly assigned to the main gate, and materials hoist is dependent on the structural elements to which they are tied, as such, planners usually freeze this type of facility in a set location (Ning et al. 2011).

In order to improve practicality of the model, this work allows unequal area facilities to be aligned vertically or horizontally, as shown in Figure 3-5. This approach overcomes the deficiency of commonly used approach that considers facilities of fixed orientation. The deficiency of fixed orientation approach is more acute when the width of a free space is smaller than the width of a temporary facility to be located in that space, the approach will not allow locating the facility. However, the proposed approach will rotate the facility at 90 degrees and check the possibility of locating the facility in the

rotated orientation. Such flexibility allows more efficient utilization of the available area and other resources.

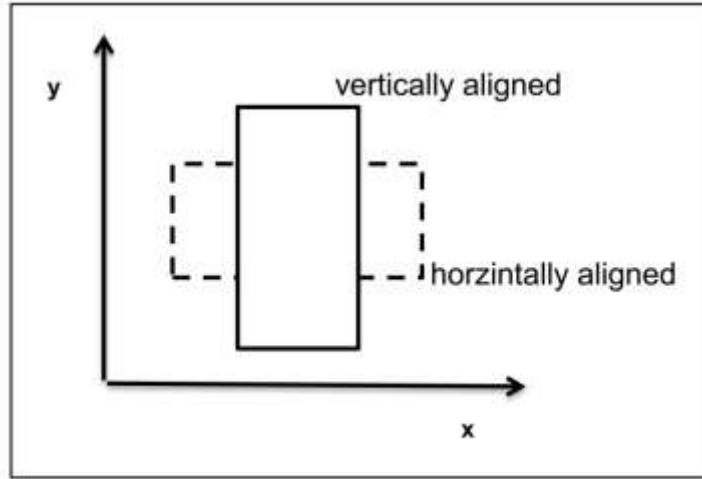


Figure 3-5 Facility alignment alternatives

3.5 Overlapping constraints

Modeling CSLP problem as a continuous search problem, as opposed to discrete quadratic assignment model, necessitates setting constraints to prevent temporary facilities from overlapping with each other or with other permanent structures such as buildings under construction. In addition to overlapping, in some situations, there must be a minimum horizontal (h_{ij}) or vertical (v_{ij}) distance to be provided between two temporary facilities or between a temporary facility and a building. Overlapping constraints are mathematically represented as follows

$$0.5(b_i + b_j) + h_{ij} - |x_i - x_j| \leq 0; i \neq j \text{ (horizontal overlapping)} \quad (3-5)$$

$$0.5(l_i + l_j) + v_{ij} - |y_i - y_j| \leq 0; i \neq j \text{ (vertical overlapping)} \quad (3-6)$$

$$violation(i, j) = \min \{0.5(b_i + b_j) + h_{ij} - |x_i - x_j|, 0.5(l_i + l_j) + v_{ij} - |y_i - y_j|\}$$

(Eq. 3-7)

Equation (3-7) ensures that only one of the two constraints will hold, in that it allows multiple facilities to be assigned in the same row or column. b_i and l_i are the horizontal and vertical dimensions of facility i , h_{ij} and v_{ij} are minimum allowable horizontal and vertical distances between facilities i and j . Each facility is tested against all other facilities and other structures. Each constraint equation is evaluated by substituting the Cartesian coordinates of two temporary facilities centroid and the dimensions of these facilities. Positive value indicate a violation to constraint, in other words if the result is larger than zero, it indicates that the temporary facility is overlapped with the other facility. The larger the value of the positive result means the more violation to that constraint. The summation of positive results yielded from testing a particular facility against all other facilities and structures is calculated. This summation represents the violation to overlapping constraints and results from locating the temporary facility in that particular location. If the result of the summation is negative or zero, this indicates that all overlapping constraints are satisfied and the facility is not overlapping with any other facility or structure.

Illustrated in Figure 3-6, the facilities A and B are overlapping on each other, while facility C is overlapping with a permanent structure.

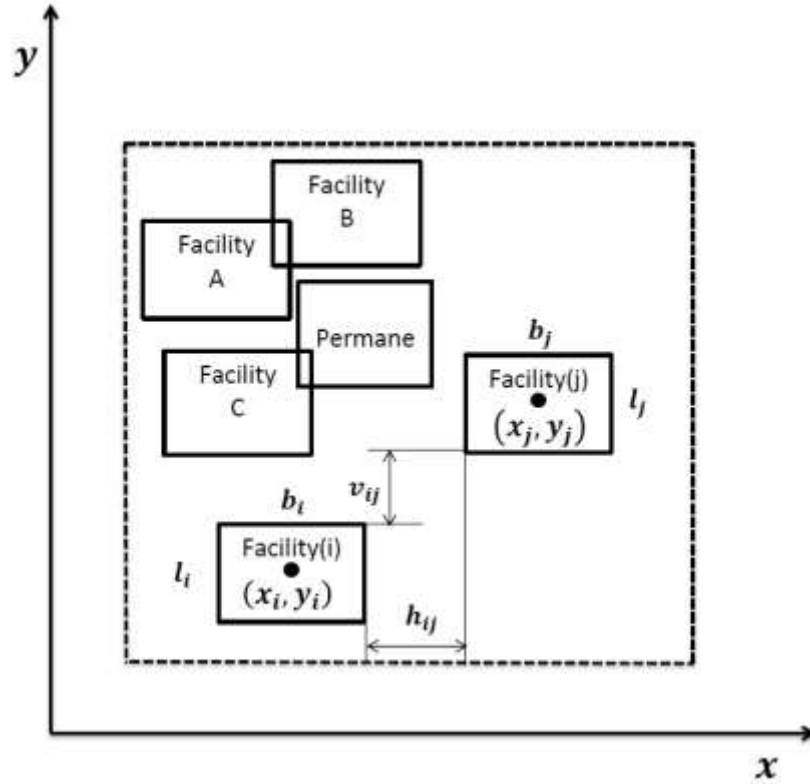


Figure 3-6 Overlapping constraints

3.6 Site boundary constraints:

Facilities have to be located within the boundaries of construction site. In the proposed model each boundary line of the construction site is represented by a mathematical equation. Each temporary facility is tested against all boundary constraints as follows. Each constraint equation is evaluated by substituting the Cartesian coordinates of the temporary facility centroid and the dimensions of that facility. Positive value indicate a violation to constraint, in other words if the result is larger than zero, it indicates that the temporary facility or part of it is laid beyond the boundary line. The larger the value of the positive result means the more violation to that constraint. The summation of positive results yielded from evaluating all boundaries constraints equations for that facility is calculated. This summation represents the violation to

boundary constraints and results from locating the temporary facility in that particular location. If the result of the summation is negative or zero, this indicates that all boundary constraints are satisfied and the facility is located within the site boundaries. The boundary constraints are represented mathematically in the form of the equation of straight lines as shown in Figure 3-7 . For example, facility A violates one boundary constraint and facility B violates two boundary constraints.

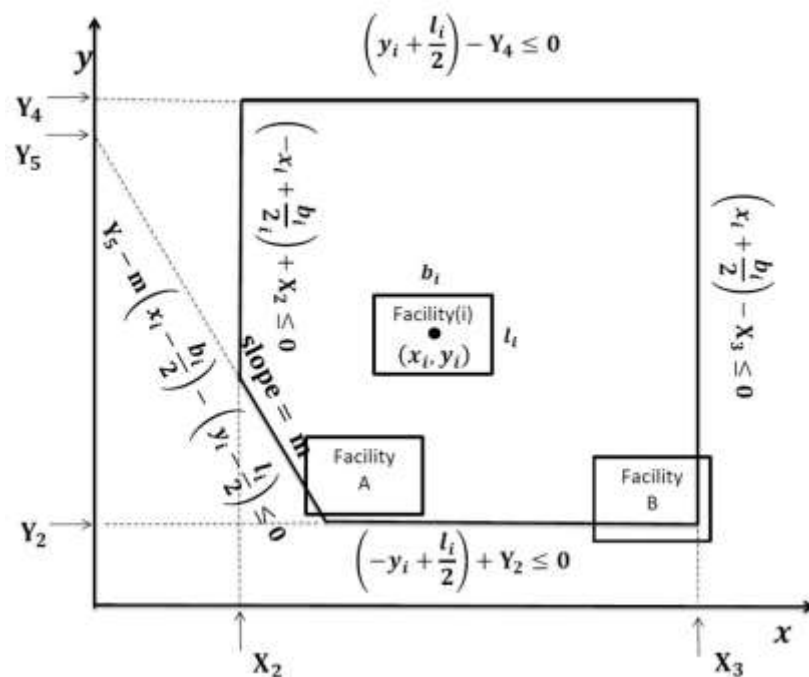


Figure 3-7 Example for boundary constraints

3.7 Tower Crane Access constraints:

Some of the temporary facilities like steel yards, rebar bending yards, and material lay down areas have to be positioned within the reachable radius of tower

cranes as illustrated in Figure 3-8. Therefore it is very important in the CSLP model that such facilities are accessible by a tower crane. For facility (i) and tower crane (t) with reachable radius R , this constraint is mathematically represented as follows:

$$\sqrt{(FCX_i - TX_t)^2 + (FCY_i - TY_t)^2} - R \leq 0 \quad (\text{Eq. 3-8})$$

FCX_i and FCY_i represent the Cartesian coordinates of most far corner of facility (i) from the tower crane. TX_t and TY_t represents the Cartesian coordinates of tower crane (t).

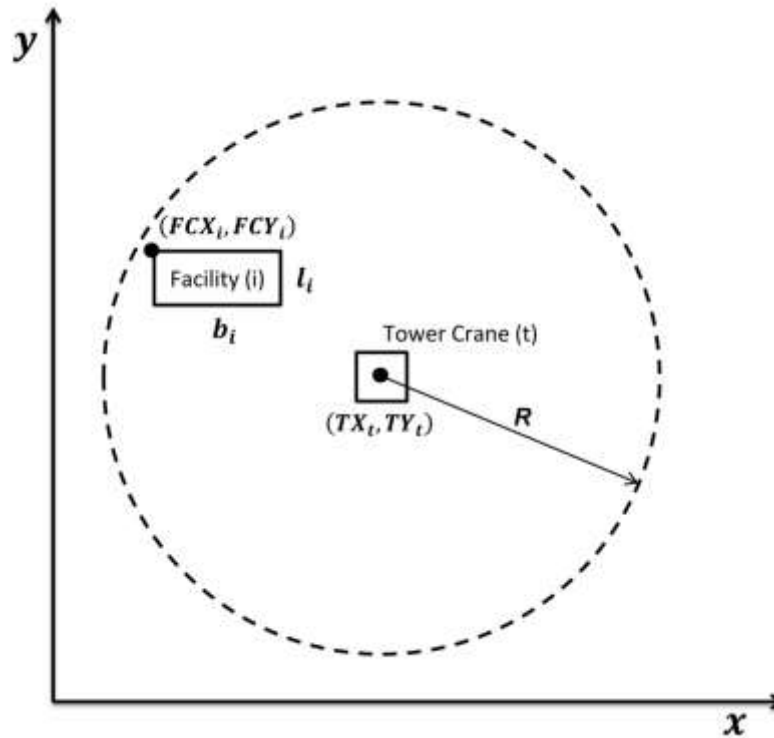


Figure 3-8 Tower crane reachable radius

Chapter 4 Multi-objective Artificial Bee Colony Algorithm with Levy flights (MOABC)

4.1 History of optimization

Optimization involves in finding one or more feasible solutions which correspond to extreme values of one or more objectives. The need for finding such optimal solutions in a problem arises mostly from the extreme purpose of either designing a solution for minimum possible cost of fabrication, or for maximum possible reliability, or others (Deb 2001).

The study of optimization problems is as old as science itself. It is known that the ancient Greek mathematicians solved many optimization problems. Euclid in around 300BC proved that a square encloses the greatest area among all possible rectangles with the same total length of four sides (Yang 2010b).

Optimization problems can be classified in too many ways according to the in terms of the number of objectives, number of constraints, function forms, landscape of the objective functions, type of design variables, uncertainty in values, and computational effort as summarized by (Yang 2010b) in Figure 4-1. According to the number of considered objectives, the optimization problem is classified into single objective problem if only one objective is considered and multi-objective problem if more than one objective is considered. The multi-objective is also referred to as multi-criteria or multi-attributes optimization. Regarding the number of constrains, the optimization problem with no constrains is classified as unconstrained. Optimization problem with one or more constrains are referred to as constrained problem.

According to the functions form the optimization problem can be either linear or non-linear. The optimization is said to be linear programming if both objective and constraints functions are all linear. However, if any of these functions is non-linear, the optimization problem becomes non-linear.

The optimization can be classified in terms of the landscape of the objective functions. For a single objective function, the shape or the landscape may vary significantly for the nonlinear case. If there is only a single valley or peak with a unique global optimum, then the optimization problem is said to be unimodal. In this case, the local optimum is the only optimum, which is also its global optimum (Yang 2010b).

According to the type of the values of the design variables the optimization problem is classified into the following categories. Integer optimization deals with optimization problems where all design variables are integers. On the contrary, in continuous optimization the design variables are continuous or they take real values in some interval. Discrete optimization finds the solution of optimization problems where all design variables are discrete. The discrete optimization is also referred to as the combinatorial optimization. If an optimization problem contains both discrete and continuous variables, the optimization is referred to as the mixed type.

According to the certainty involved, the optimization problem is divided into two categories: deterministic and stochastic. The optimization problems is said to be deterministic if for any given set of design variables, the values of both objective functions and the constraint functions are determined exactly. If there is any uncertainty

and noise in the design variables and the objective functions and/or constraints, then the optimization is said to be a stochastic optimization problem.

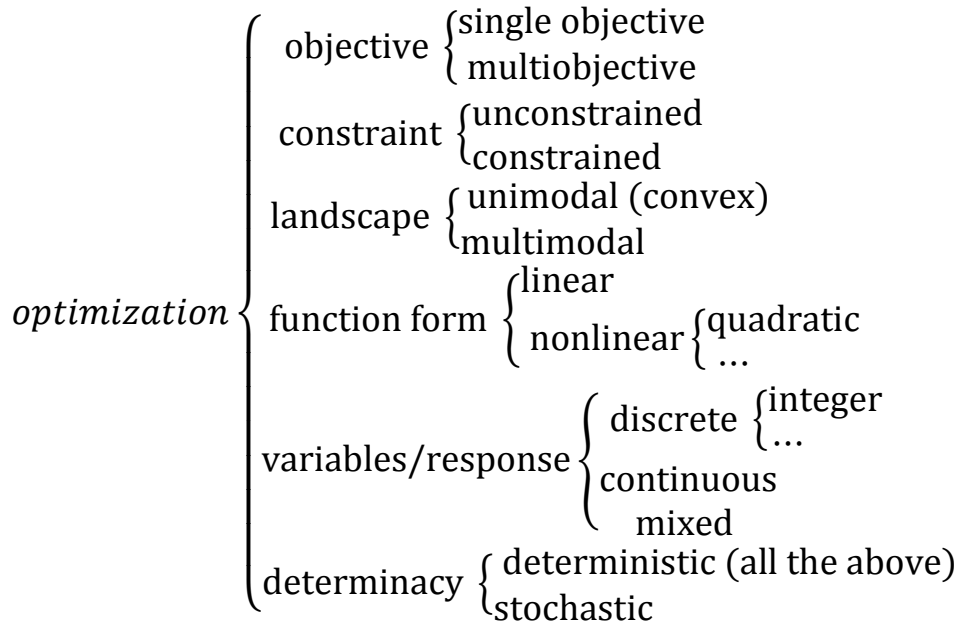


Figure 4-1 Classification of optimization problems (Yang, 2010b)

4.2 Optimization algorithms

After an optimization problem is formulated, the main task is to find the optimal solution in case of single objective function or set of optimal solutions, i.e. Pareto front by some solution procedure using the right mathematical techniques (Yang, 2010a). For different types of optimization problems, we often have to use different optimization techniques because some algorithms are more suitable for certain types of optimization than others (Yang, 2010b). The field of search and optimization has changed over the last few years by the introduction of a number of non-classical, unorthodox and stochastic search and optimization algorithms (Deb, 2001).

Optimization algorithms generally can be classified into two categories according to the nature of the problem. First, deterministic algorithms which follow a rigorous

procedure and its path and values of both design variables and the functions are repeatable. Such algorithms does not have random component, therefore at any time starting with a particular initial conditions will produce exactly the same results. Hill climbing is an example for deterministic algorithms. Second, stochastic algorithms in which always there is a random component, hence each time one runs the program, the algorithm produces different result. Genetic Algorithms is an example for stochastic algorithms. The optimization algorithms are summarized by (Yang 2010b) in Figure 4-2.

For deterministic algorithms there are several types. The most common are (1) linear programming like simplex method as an example; (2) nonlinear programming; (3) gradient based algorithms that use the function values and their derivatives like Newton-Raphson algorithm; (4) Non-gradient-based or gradient-free algorithms which do not use any derivative, but only the function values. Nelder-Mead downhill simplex is an example of non-gradient algorithms.

Stochastic algorithms can be generally divided into two main categories that are slightly differing from each other: heuristic and meta-heuristic algorithms. Heuristic algorithms can be defined as optimization by trial-and-error. In other hand, meta-heuristic algorithms, *meta-* means 'beyond' or 'higher level' use a combination of randomization and local search. They are usually inspired from nature and biological systems. Ant algorithms, bee algorithms, genetic algorithms, and particle swarm optimization are the most widely known meta-heuristic algorithms.

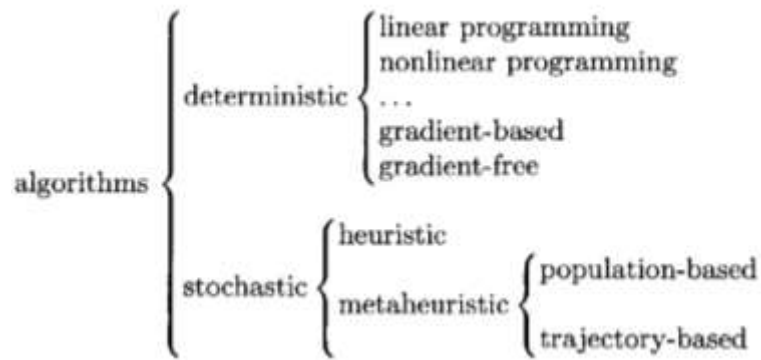


Figure 4-2 Classification of algorithms (Yang 2010b)

4.3 Bee Algorithm

Almost all bee algorithms are inspired by the foraging behavior of honeybees in nature. Interesting characteristics, such as waggle dancing, polarization, and nectar maximization, are often used to simulate the allocation of the foraging bees along flower patches, and thus in different regions of the search space. Several variants of bee algorithms have been formulated, including the honeybee algorithm (HBA), the virtual bee algorithm (VBA), the artificial bee colony (ABC), the honeybee-mating algorithm (HBMA) and others (Yang, 2010a).

4.3.1 Behavior of honeybees

Honeybees are social insects by nature; hence they live in colonies sharing and dividing responsibilities in a well-organized manner. The basic three components of a foraging model are: food sources, employed foragers and unemployed foragers. The model defines two leading modes of the behavior: the recruitment to a nectar source and the abandonment of a source. Food Sources such as flower patches are evaluated

based on several factors such as its proximity to the nest, its richness or concentration of its energy, and the ease of extracting this energy. Employed foragers are associated with a particular food source which they are currently exploiting. When returned to the nest, employed bees will communicate information regarding this particular food source. Such information includes its distance and direction from the nest, the profitability of the source. Unemployed foragers are continuously looking out for a food source to exploit. They are classified into two types: scouts, searching for new food sources in the environment neighboring the nest and onlookers who are waiting in the nest and establishing a food source through the information shared by employed foragers.

At dancing area in the nest, honeybees communicate information related to the quality of food sources and its location via dancing. This dance is called a waggle dance. The dance consists of two loops with a straight run in the middle. The direction of the straight run determines the direction of the food source. The rate of looping and duration of buzzing indicate the distance to the food supply as illustrated in the Figure 4-3 (Frisch Von, 1976) and Figure 4-4 (Barth, 1982).

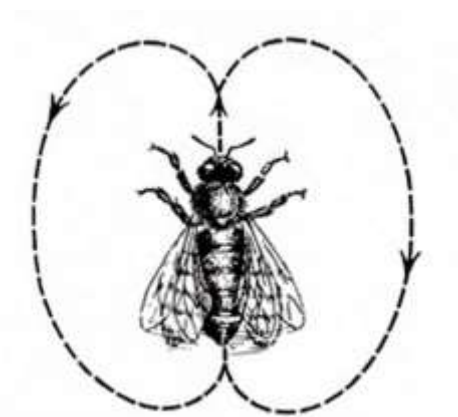


Figure 4-3 Waggle dance of honey bees (Frisch 1976)

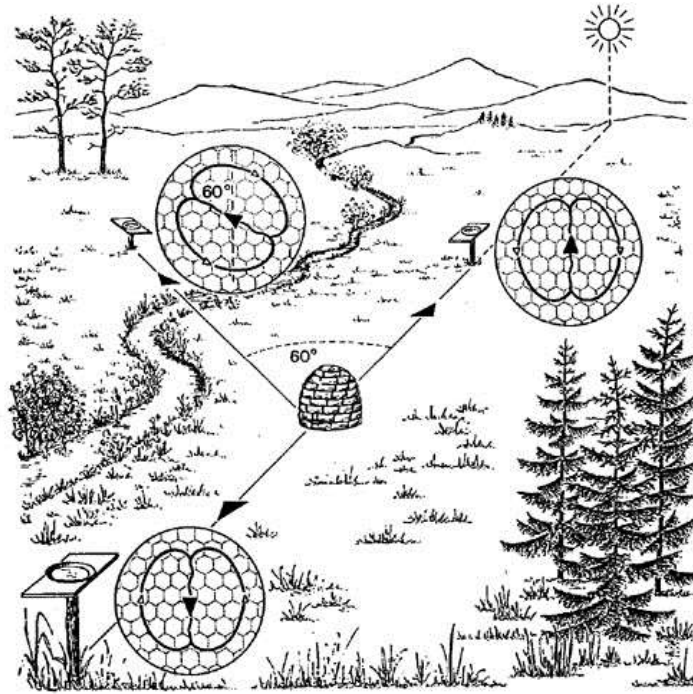


Figure 4-4 Decoding the language of the Bee dance (Barth, 1982).

Since information about all the current rich sources is available to an onlooker on the dance floor, probably she can watch numerous dances and decides to employ herself at the most profitable source. There is a greater probability of onlookers choosing more profitable sources since more information is circulated about the more profitable sources. Employed foragers share their information with a probability proportional to the profitability of the food source, and the sharing of this information through waggle dancing is longer in duration. Hence, the recruitment is proportional to the profitability of the food source.

Karaboga (2005) in his research well explained the characteristics of basic behavior of foragers as illustrated in Figure 4-5 and detailed as follows. For two discovered food sources: A and B. At the very beginning, a potential forager will start as unemployed forager. That bee will have no knowledge about the food sources around

the nest. The two possible options for such a bee are: (1) it can be a scout and starts searching around the nest spontaneously for a food due to some internal motivation or possible external clue (S on Figure 4-5); (2) it can be a recruit after watching the waggle dances and starts searching for a food source (R on Figure 4-5).

After locating the food source, the bee utilizes its own capability to memorize the location and then immediately starts exploiting it. Hence, the bee will become an “employed forager”. The foraging bee takes a load of nectar from the source and returns to the hive and unloads the nectar to a food store. After unloading the food, the bee has the following three options: (1) it becomes an uncommitted follower after abandoning the food source (UF); (2) it dances and then recruits nest mates before returning to the same food source (EF1)); (3) it continues to forage at the food source without recruiting other bees (EF2).

The experiments confirmed that new bees begin foraging at a rate proportional to the difference between the eventual total number of bees and the number of present foraging (Karaboga, 2005)

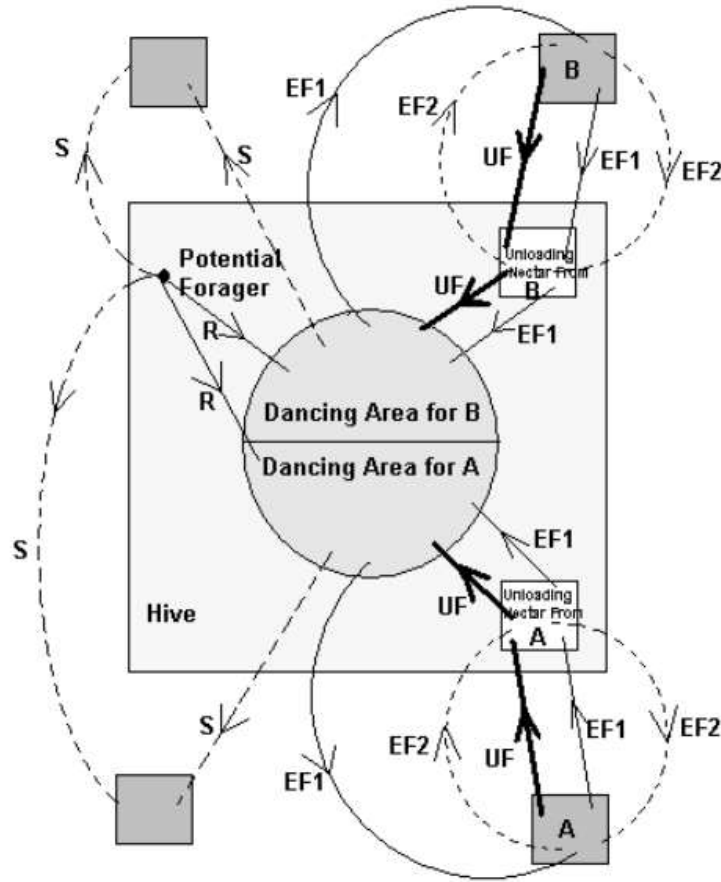


Figure 4-5 The behaviour of honey bee foraging for nectar (Karaboga, 2005)

4.4 Artificial Bee Colony (ABC) algorithm

Introduced by Dervis Karaboga in 2005, ABC algorithm simulated the foraging behavior of a bee colony. Bees aim to maximize the nectar amount unloaded to the food stores in the hive. In the ABC algorithm a honey bee swarm is classified to three categories: employed bees, onlooker bees, and scout bees. Half of the colony bees are employed bees and each food source is exploited by only one employed bee that carries information about this particular food source and share information with other bees waiting in the hive. The information is shared via wiggle dancing. The other half includes the onlooker bees who try to find a food source by means of the information given by the

employed bees. Exhausted food sources are abandoned and the employed bees whose food source has been abandoned become a scout bee. Scout bees randomly search the environment in order to find new food sources. The main steps of the basic version of ABC algorithms are presented in Figure 4-6.

ABC has been widely applied to solve real world optimization problems. A comprehensive survey for ABC algorithm applications was presented by Karaboga et al. (2012). Results show that ABC algorithm is better than or similar to those of other population based algorithms (Karaboga and Akay, 2009). Karaboga and Akay (2011) proposed new version of ABC algorithm to solve constrained optimization problem under single objective. Number of studies extended the standard version of ABC to search for optimality under multi-objective problems (Akbari et al. 2011), (Zou et al. 2012).

In the modified ABC algorithm developed by Karaboga and Akay (2011) to solve single objective constrained problems, the mathematical representation of the optimization problem and modified ABC algorithms are presented in Figure 4-7 to Figure 4-13.

Send the scouts onto the initial food sources

REPEAT

Send the employed bees onto the food sources and determine their nectar amounts

Calculate the probability value of the sources with which they are preferred by the onlooker bees

Send the onlooker bees onto the food sources and determine their nectar amounts

Stop the exploitation process of the sources exhausted by the bees

Send the scouts into the search area for discovering new food sources, randomly

Memorize the best food source found so far

UNTIL (requirements are met)

Figure 4-6 Steps of ABC algorithm (Karaboga 2005)

In the initialization step, ABC algorithm generates randomly an initial population of sn solutions. The number of solutions (sn) equals to half of the size of population (NP) and D is the number of optimization parameters. The initial solution is set within the upper and lower boundaries Figure 4-9.

After evaluating the initial population, it is subjected to repeated cycles of the search processes of the employed bees, the onlooker bees and scout bees. Employed bee procedure of the ABC algorithm is presented in Figure 4-10.

After producing a new food source (solution), ABC algorithm makes a selection. The modified ABC algorithm to solve constrained optimization problems is using Deb's rules instead of using greedy selection in selection process. Structure of the algorithm directs the solutions to feasible region in searching process. Applying Deb's rules, the bee either memorizes the new position by forgetting the old one or keeps the old one.

Deb's method uses a tournament selection operator, where two solution vectors u and v are compared at a time by applying following criteria:

- Any feasible solution ($violation_u \leq 0$) is preferred to any infeasible solution ($violation_v > 0$), solution vector u is dominant;
- Among two feasible solutions ($violation_u \leq 0, violation_v \leq 0$), the one having better objective function value is preferred ($f_u < f_v$, *solution $_u$ is dominant*);
- Among two infeasible solutions ($violation_u > 0, violation_v > 0$), the one having smaller constraint violation is preferred ($violation_u < violation_v$),

After all employed bees complete the search process; they share the nectar information of the food sources and their position information with the onlooker bees on the dance area by calculating probability values as presented in Figure 4-11.

Based on calculated probability values onlooker bee evaluates the information taken from all employed bees and produces a new solution as presented in Figure 4-12, then the selection process is conducted using Deb's rule.

If solution cannot be improved through scout production period (SPP) the food source is abandoned by its bee and a new food source is randomly initialized by scout bee as presented in Figure 4-13.

$$\begin{aligned}
& \text{minimize } f(\bar{x}), \quad \bar{x} = (x_1, \dots, x_n) \in \mathbb{R}^n \\
& \text{subject to :} \quad l_i \leq x_i \leq u_i, \quad i = 1, \dots, n \\
& \quad \quad \quad g_j(\bar{x}) \leq 0, \quad \text{for } j = 1, \dots, q \\
& \quad \quad \quad h_j(\bar{x}) = 0, \quad \text{for } j = q + 1, \dots, m
\end{aligned} \tag{1}$$

Figure 4-7 Constrained single objective optimization problem

Algorithm 1 (*Pseudo-code of main body of ABC algorithm*).

1:	Initialization
2:	Evaluation
3:	cycle = 1
4:	repeat
5:	Employed Bees Phase
6:	Calculate Probabilities for Onlookers
7:	Onlooker Bees Phase
8:	Scout Bees Phase
9:	Memorize the best solution achieved so far
10:	cycle = cycle + 1
11:	until cycle = Maximum Cycle Number

Figure 4-8 Main body of ABC algorithm (Karaboga and Akay, 2011)

Algorithm 2 (Initialization step)

```

for  $i = 1$  to  $sn$  do
    for  $j = 1$  to  $D$  do
        Generate  $\vec{x}_i$  solution
         $x_i^j = x_{min}^j + rand(0, 1)(x_{max}^j - x_{min}^j)$ 
        where  $x_{min}^j$  and  $x_{max}^j$  are lower and upper bound of the parameter  $j$ ,
        respectively.
    end for
     $failure_i = 0$ 
end for

```

Figure 4-9 Initialization step

Algorithm 3 (Employed bees phase)

```

for  $i = 1$  to  $sn$  do
    for  $j = 1$  to  $D$  do
        Produce a new food source  $\vec{v}_i$  for the employed bee of the food source
         $\vec{x}_i$  by using
        
$$v_{ij} = \begin{cases} x_{ij} + \phi_{ij}(x_{ij} - x_{kj}), & \text{if } R_j < MR \\ x_{ij}, & \text{otherwise} \end{cases}$$

        where  $k \in \{1, \dots, sn\}$  is randomly chosen index,  $\phi$  is
        uniformly distributed random real number in the range  $[-1, 1]$ ,
        and  $R_j$  is uniformly distributed number in the range  $[0, 1]$ .
         $MR$  is control parameter of ABC algorithm in the range  $[0, 1]$ 
        which control the number of parameters to be modified.
    end for
    If no parameter is changed, one random parameter to be changed by:
    
$$v_{ij} = x_{ij} + \phi_{ij}(x_{ij} - x_{kj})$$


    where  $j$  is randomly distributed integer  $[1, D]$ ,  $D$  is number of parameters.
    Evaluate the quality of  $\vec{v}_i$ 
    Apply the selection process between  $\vec{v}_i$  and  $\vec{x}_i$  based on Deb's method
    If solution  $\vec{x}_i$  does not improve  $failure_i = failure_i + 1$ 
    , otherwise  $failure_i = 0$ 
end for

```

Figure 4-10 Employed bees algorithm

Algorithm 4 (Probabilities for onlooker bees)

for $i = 1$ to sn **do**

Calculate the probability values p_i for the solutions using fitness and/or violation of the solutions by:

$$p_i = \begin{cases} 0.5 + \left(\frac{fitness_i}{\sum_{j=1}^{sn} fitness_j} \right) \times 0.5 & \text{if solution is feasible} \\ \left(1 - \frac{violation_i}{\sum_{j=1}^{sn} violation_j} \right) \times 0.5 & \text{if solution is infeasible} \end{cases}$$

where $violation_i$ is the penalty value of the solution $\vec{x}_i$ and $fitness_i$ is the fitness value of the solution $\vec{x}_i$ which is proportional to the nectar amount of that food source. The fitness is determined by :

$$fitness_i = \begin{cases} 1/(1 + f_i) & \text{if } f_i \geq 0 \\ 1 + abs(f_i) & \text{if } f_i < 0 \end{cases}$$

where f_i is the cost value of the solution $\vec{x}_i$

end for

Figure 4-11 Probabilities for onlooker bees

Algorithm 5 (Onlooker bees phase)

$t = 0, i = 1$

Repeat

if $random < p_i$ **then**

$t = t + 1$

for $j = 1$ to D **do**

produce a new food source $\vec{v}_i$ for the onlooker bee of the food source $\vec{x}_i$ by using

$$v_{ij} = \begin{cases} x_{ij} + \phi_{ij}(x_{ij} - x_{kj}), & \text{if } R_j < MR \\ x_{ij}, & \text{otherwise} \end{cases}$$

end for

Apply the selection process between $\vec{v}_i$ and $\vec{x}_i$ based on Deb's method

If solution $\vec{x}_i$ does not improve $failure_i = failure_i + 1$

, otherwise $failure_i = 0$

end if

$i = i + 1$

$i = i \bmod ((sn + 1) + 1)$

until $t = sn$

Figure 4-12 Onlooker bees Algorithm

```

Algorithm 6 (Scout bees phase)
if  $cycle \bmod SPP = 0$  then
    if  $\max(failure_i) > limit$  then
        replace  $\vec{x}_i$  with a new randomly produced solution by
         $x_i^j = x_{min}^j + rand(0,1)(x_{max}^j - x_{min}^j)$ 
    end for
     $failure_i = 0$ 
end for

```

Figure 4-13 Scout bees Algorithm

4.5 Multi-objective optimization (MOO)

Multi-objective optimization problems are intended to achieve two goals. First goal is to find a number of optimal solutions, referred as pareto-optimal front. Second goal is to maintain a diverse set of solutions in that pateto-optimal front (Deb 2001).

The general form for multi-objective optimization problem is mathematically represented as follows:

$$\text{Minimize/Maximize } f_m(x), \quad m = 1, 2, \dots, M; \quad (\text{Eq. 4-1})$$

Subject to j inequality constraints

$$g_j(x) \geq 0, \quad j = 1, 2, \dots, J; \quad (\text{Eq. 4-2})$$

And k equality constraints

$$H_k(x) = 0, \quad k = 1, 2, \dots, K; \quad (\text{Eq. 4-3})$$

And restricting the decision variables within the upper and lower limits

$$x_i^{(L)} \leq 0 \leq x_i^{(U)}, \quad i = 1, 2, \dots, n; \quad (\text{Eq. 4-4})$$

MOO solutions are in such a way that the performance of each objective cannot be improved without sacrificing the performance of at least another one. Hence, the solution to a MOO problem exists in the form of an alternate trade-off known as a Pareto optimal set. The Pareto optimal set is defined based on Pareto dominance.

For minimization problems, solution vector $\mathbf{u}$ is said to dominate another vector $\mathbf{v}$ if and only if $u_i \leq v_i$ for $\forall i \in \{1, \dots, n\}$ and $\exists i \in \{1, \dots, n\} : u_i < v_i$. In other words, no component of $\mathbf{u}$ is larger than the corresponding component of $\mathbf{v}$, and at least one component is smaller.

For two-objective optimization problem, maximizing f_1 and minimizing f_2 , with five different solutions in the objective space, the dominance concept is described as follows as presented by Deb (2001) in Figure 4-14. Solution 1 produces higher value of f_1 than solution 2 and solution 1 produces lower value of f_2 than solution 2, therefore solution 1 is said to dominate solution 2, as it is better for both functions. In the same way, solutions 3 and 5 are said to dominate solutions 1, 2, and 4. In another hand when comparing solutions 3 and 5, we observe that solution 3 produces better value for the second objective function, while solution 5 produces better value for the first objective function. Therefore, none of these solutions is dominating the other. From the above analysis we conclude that solutions 3 and 5 are non-dominated solutions.

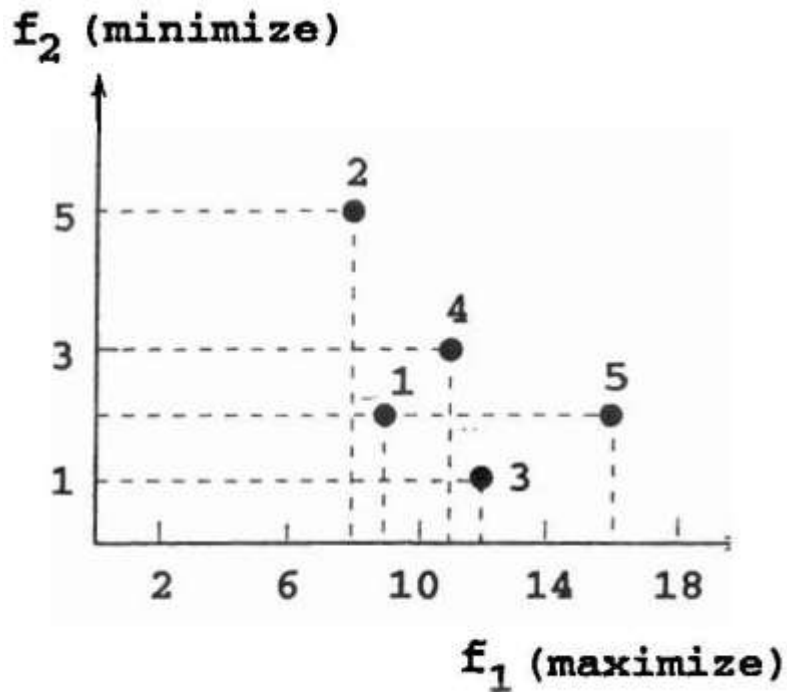


Figure 4-14 A population of five solutions (Deb 2001)

Figure 4-15 illustrate the four different scenarios with two objectives optimization f_1 and f_2 in term of minimizing or maximizing the objective function. The shaded area shows the search space and the solid curves marks the pareto-optimal solution set which can be established using the domination concept.

Deb (2001) emphasized that in a multi-objective optimization; ideally the effort must be made in finding the set of trade-off optimal solution by considering all objective to be important. Then, higher-level qualitative information has to be considered to choose the suitable single solution among this set. Accordingly, Deb suggested the following principle for an ideal multi-objective optimization procedure illustrated in Figure 4-16. In first step find multiple trade-off optimal solutions with a wide range of values of

objectives. In second step choose a single solution of the obtained solutions using higher-level information.

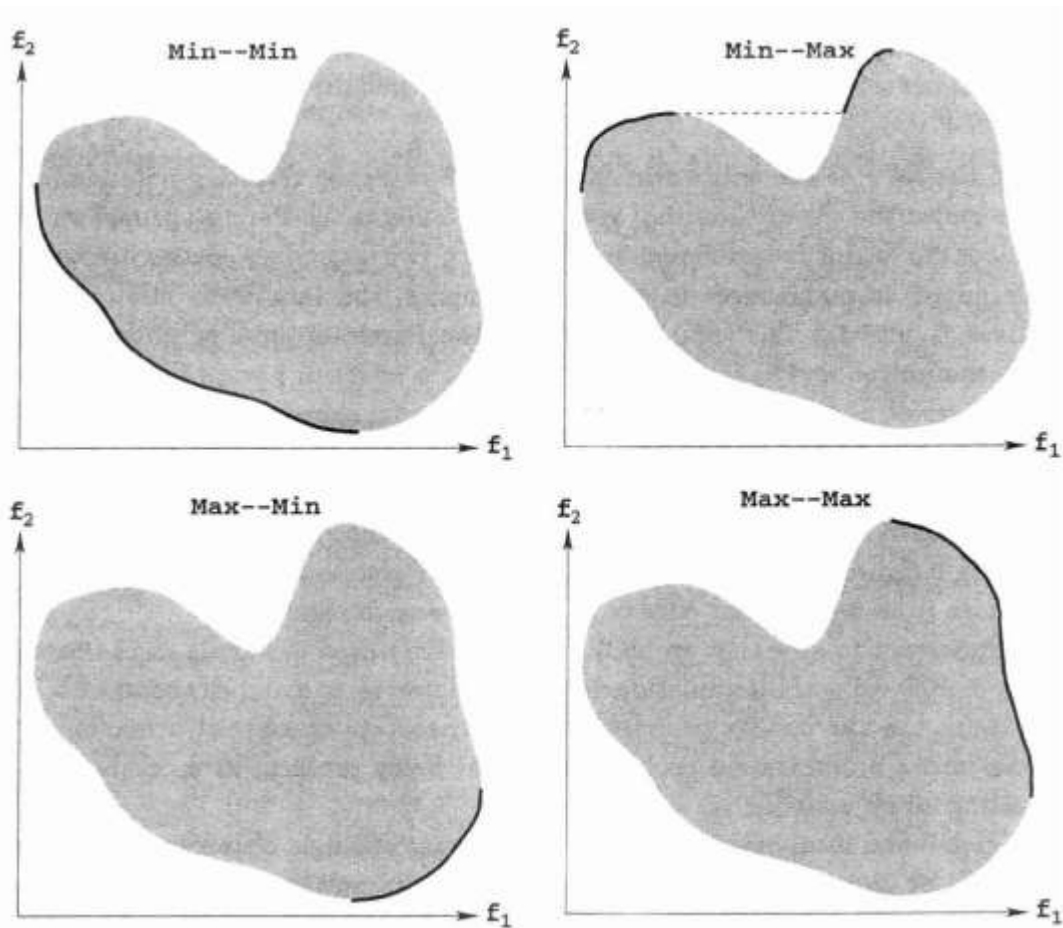


Figure 4-15 Pareto-optimal solution are marked with continuous curves for four combinations of two types of objectives (Deb 2001)

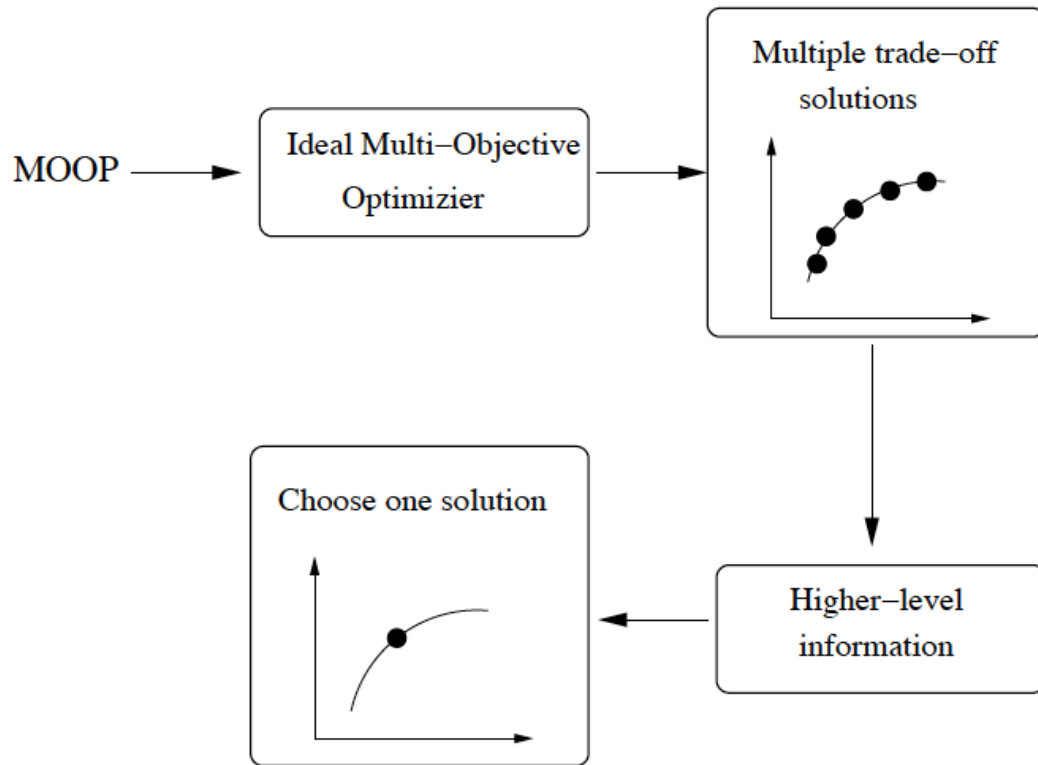


Figure 4-16 Schematic of an ideal multi-objective optimization procedure (Modified from Deb 2001)

4.6 Multi-Objective Artificial Bee Colony algorithms (MOABC)

Based on ABC algorithm developed by Karaboga and Akay (2011) two MOABC algorithms, Basic-MOABC and MOABC via Levy Flights are developed in this study to solve for multi-objective optimization problems. The flowchart of MOABC via Levy Flights is shown in Figure 4-17 and steps of MOABC algorithms are described as follows:

Step 1: Generate randomly an initial population of sn solutions, where the number of solutions (sn) equals to half of the size of population (NP). The initial solutions are set within the upper and lower boundaries.

Step 2: Calculate the value of objective functions, violation to constraints, and fitness of evaluated solution. The fitness of initial food sources is calculated based on non-domination as follows.

$$fitness_i = ND_i/sn \quad (Eq. 4-5)$$

$fitness_i$ refers to the fitness of solution i and ND_i refers to number of other solutions that do not dominate solution i .

For minimization problems, solution vector $\mathbf{u}$ is said to dominate another vector $\mathbf{v}$ if and only if $u_i \leq v_i$ for $\forall i \in \{1, \dots, n\}$ and $\exists i \in \{1, \dots, n\} : u_i < v_i$. In other words, no component of $\mathbf{u}$ is larger than the corresponding component of $\mathbf{v}$, and at least one component is smaller.

Step 3: Find pareto optimal by applying modified Deb's rules which search for feasible solutions on assumption that any feasible solution is better than any infeasible one (Deb, 2000). Deb's method uses a tournament selection operator, where two solution vectors $\mathbf{u}$ and $\mathbf{v}$ are compared at a time by applying following criteria:

- Any feasible solution ($violation_u \leq 0$) is preferred to any infeasible solution ($violation_v > 0$), solution vector $\mathbf{u}$ is dominant;
- Among two feasible solutions ($violation_u \leq 0, violation_v \leq 0$), for minimization problems, apply domination criteria mentioned in Step 2 i.e. solution vector $\mathbf{u}$ is said to dominate another vector $\mathbf{v}$ if and only if $u_i \leq v_i$ for $\forall i \in \{1, \dots, n\}$ and $\exists i \in \{1, \dots, n\} : u_i < v_i$;

- Among two infeasible solutions ($violation_u > 0, violation_v > 0$), the one having smaller constraint violation is preferred ($violation_u < violation_v$), solution vector u is dominant.

The following steps of the search processes of the employed bees, the onlooker bees and scout bees are carried out till reaching a maximum number of cycles or a stop criterion.

Step 4: Produce a new food source s_i for the employed bee of food source x_i .

For Basic-MOABC algorithms developed a new food source is generated as follows:

$$s_{ij} = \begin{cases} x_{ij} + \phi_{ij}(x_{ij} - x_{kj}), & \text{if } R_j < MR \\ x_{ij}, & \text{otherwise} \end{cases} \quad (\text{Eq. 4-6})$$

where $k \in \{1, \dots, sn\}$ is randomly chosen index, ϕ is uniformly distributed random real number in the range $[-1, 1]$, and R_j is uniformly distributed number in the range $[0, 1]$. MR is control parameter of ABC algorithm in the range $[0, 1]$ which control the number of parameters to be modified. If no parameter is changed, one random parameter to be changed by

$$s_{ij} = x_{ij} + \phi_{ij}(x_{ij} - x_{kj}) \quad (\text{Eq. 4-7})$$

j is randomly distributed integer $[1, D]$, D is number of parameters.

For MOABC via Levy Flight algorithm in order to maximize the efficiency of food source searches conducted by employed bees, this work proposes to use Levy flights to produce a new food source s_i for the employed bee of food source x_i as follows:

$$s_i = x_i + \alpha_0 \text{Levy}(\beta)(x_i - x_j) \quad (\text{Eq. 4-8})$$

where α_0 is a constant and the function $\text{Levy}(\beta)$ is explained in section 4.7 which refers to generating random walk with Levy Flights.

Step 5: Evaluate the new solution, compare to current solution and keep the dominant solution by rules described in (Step 3).

Step 6: Calculate probabilities of sending onlooker bees.

$$P_i = \begin{cases} 0.5 + \left(\frac{\text{fitness}_i}{\sum_{j=1}^{sn} \text{fitness}_j} \right) \times 0.5 & \text{if solution is feasible} \\ \left(1 - \frac{\text{violation}_i}{\sum_{j=1}^{sn} \text{violation}_j} \right) \times 0.5, & \text{if solution is infeasible} \end{cases} \quad (\text{Eq. 4-9})$$

Step 7: Based on probability value calculated in (Step 6) onlooker bee evaluates the information taken from all employed bees and produce a new solution by equations (Eq. 4-6) and (Eq. 4-7) [in Step 4].

Step 8: Evaluate the solution and apply selection process rules in (Step 3).

Step 9: Add the food sources to archive, compare and keep only non-dominated solutions in the archive.

Step 10: If solution cannot be improved through scout production period (SPP) the food source is abandoned by its bee and a new food source is randomly initialized by scout bee.

Step 11: terminate if the maximum cycle number is reached.

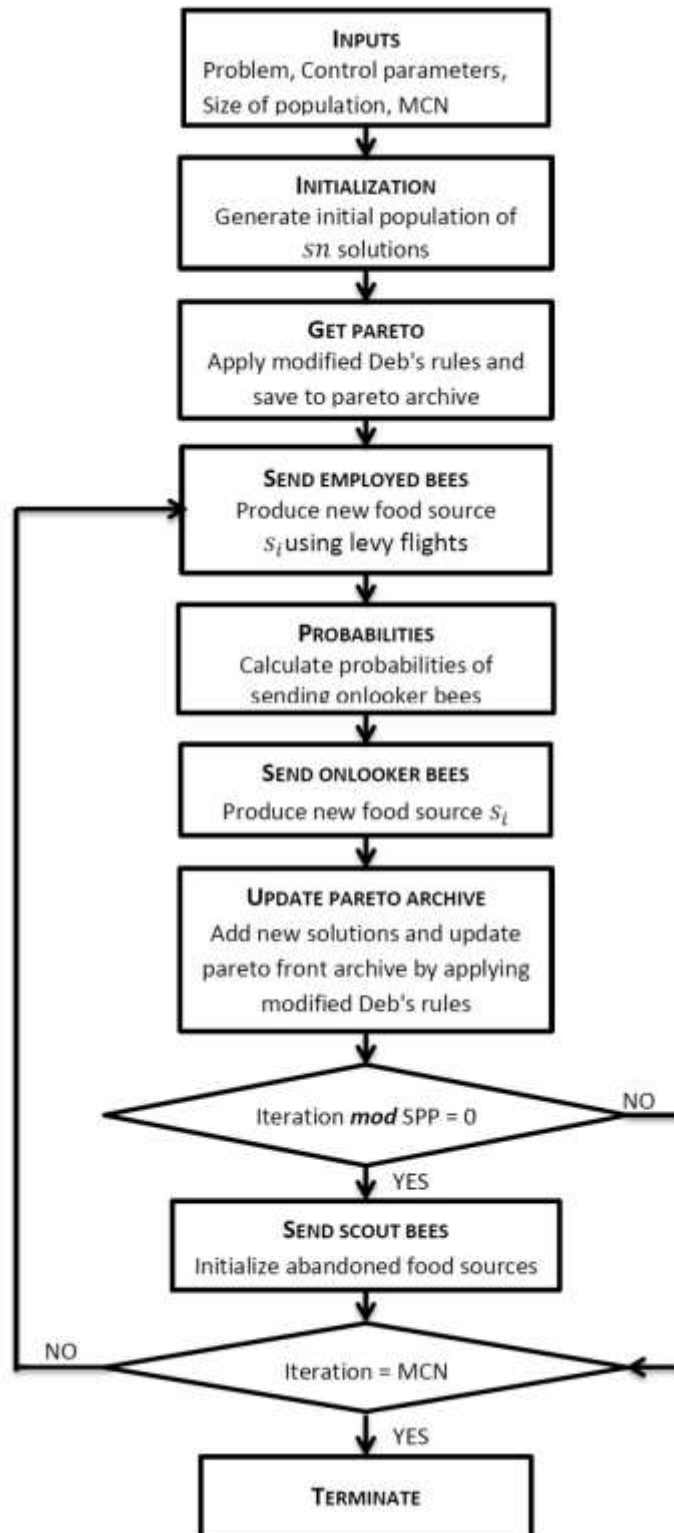


Figure 4-17 The flow chart of MOABC with levy flights

4.7 Levy Flights

Randomization plays an important role in both exploration and exploitation, or diversification and intensification. In meta-heuristic algorithms randomization process is performed by means of random walks. Random walks are defined as a stochastic process which consists of a series of random steps, forming a trajectory of consecutive random steps (Yang, 2010a). Random variables are generated to obey a particular distribution. Examples of such distributions are normal or Gaussian distribution, Poisson distribution, and levy distribution.

A recent study by Reynolds and Frye shows that fruit flies or *Drosophila melanogaster*; explore their landscape using a series of straight flight paths punctuated by a sudden 90° turn, leading to a Levy-flight-style intermittent scale-free search pattern. Subsequently, such behavior has been applied to optimization and optimal search, and preliminary results show its promising capability (Yang and Deb, 2009).

Levy Flights are random walks whose step length is drawn from the levy distribution.

$$Levy \sim u = t^{-1-\beta}. (0 < \beta < 2) \quad (\text{Eq. 4-10})$$

Various studies have shown that flight behavior of many animals and insects has demonstrated the typical characteristics of Levy flights (Yang and Deb, 2009). When generating new solutions s_i , a new food source for a bee in sources x_i a Levy flight is performed as follows:

$$s_i = x_i + \alpha * Levy(\beta) \quad (\text{Eq. 4-11})$$

Where $\alpha > 0$ is the step size which should be related to the scales of the problem of interest. In most cases, we can use $\alpha = O(1)$. To accommodate the difference between solutions quality, we use

$$\alpha = \alpha_0 (x_i - x_j) \quad (\text{Eq. 4-12})$$

where α_0 is a constant, while the term in the bracket corresponds to the difference of two randomly generated solutions. Yang (2010b) discussed in details the scheme of generating step size by levy flights and summarized as follows:

$$step = \alpha_0 (x_i - x_j) * Levy(\beta) \sim \alpha_0 \frac{u}{|v|^{1/\beta}} (x_i - x_j) \quad (\text{Eq. 4-13})$$

where u and v are drawn from normal distributions. That is

$$u \sim N(0, \sigma_u^2), \quad v \sim N(0, \sigma_v^2), \quad (\text{Eq. 4-14})$$

With

$$\sigma_u = \left\{ \frac{\Gamma(1+\beta) \sin(\pi\beta/2)}{\Gamma[(1+\beta)] \beta 2^{(\beta-1)/2}} \right\}, \quad \sigma_v = 1 \quad (\text{Eq. 4-15})$$

Here Γ is the standard Gamma function.

In their study, Yang and Deb (2013) presented an example of drawing 100 steps which obey a Levy distribution in Figure 4-18, where the number of rises represents the number of steps and the heights represent the length of steps. The path of these steps during a Levy flight is illustrated in Figure 4-19. The high efficiency of Levy flights in exploring unknown, large-scale search space is due to the fact that the variance of Levy

flights increases much faster than linear relationships of other random walks (Yang and Deb 2013).

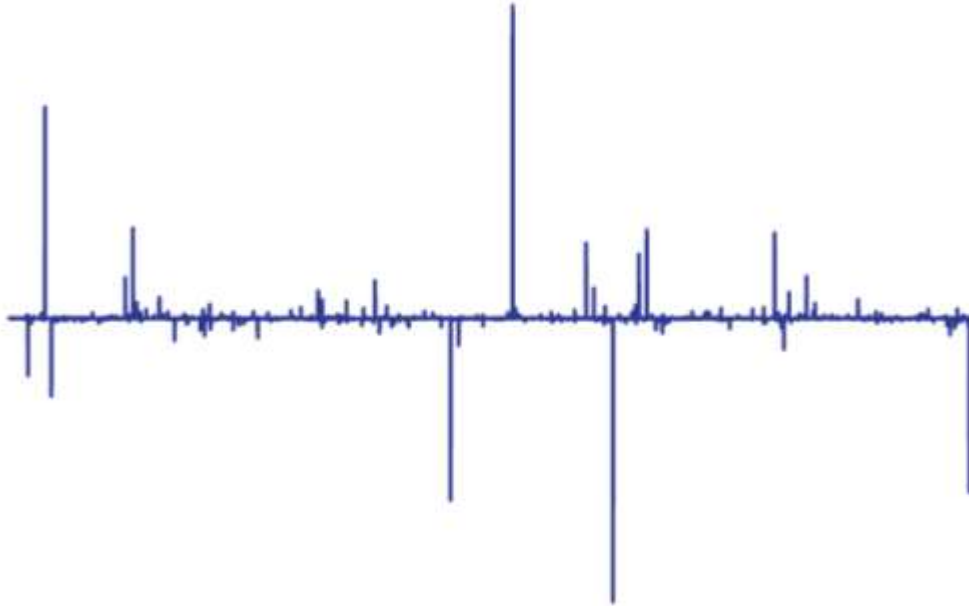


Figure 4-18 Distribution of 100 consecutive steps (Yang and Deb 2013)

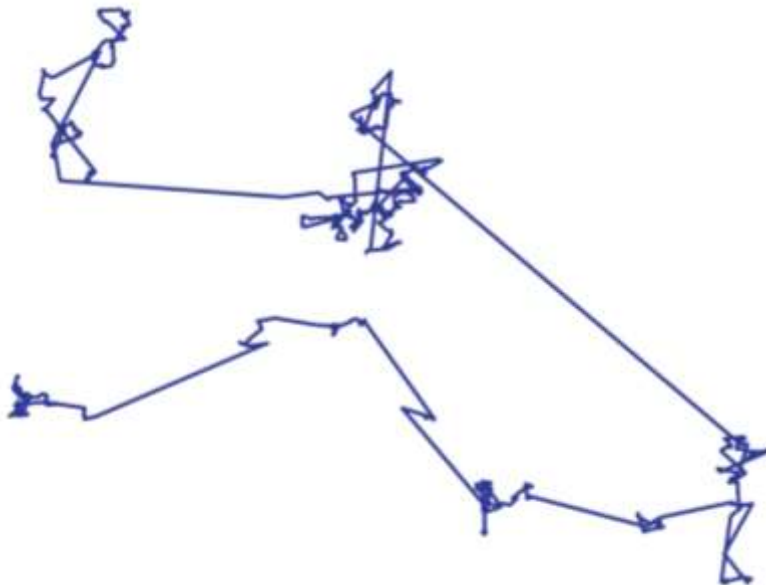


Figure 4-19 2D Levy flights in 100 steps (Yang and Deb 2013)

Chapter 5 Computational experiments and Case study

5.1 Computational experiments

In order to verify the performance of the multi-objective artificial bee colony with levy flights, the proposed algorithm is used to solve some popular mathematical and engineering benchmarked problems.

The first subset of test problems includes three scalable, unconstrained, algebraic, multi-objective problems with two objective functions chosen from ZDT series designed by Zitzler, Deb, and Thiele (Deb 2001). The solutions obtained by the proposed method are comparable with those attained by some of the most recent algorithms, multiobjective firefly algorithm (MOFA), multiobjective cuckoo search algorithm (MOCS), and the basic multiobjective artificial bees colony algorithm (MOABC-basic) and results are summarized in Table 5-1 (Yang 2013, Yang and Deb 2013). The second subset of problems includes two engineering design optimization benchmark problems which are the cantilever beam design, and disc brake design.

5.1.1 Test problem ZDT1 function with a convex front

$$f_1(x) = x_1, \quad f_2(x) = g(1 - \sqrt{f_1/g}) \quad (\text{Eq. 5-1})$$

$$g = 1 + \frac{9 \sum_{i=2}^d x_i}{d-1}, x_i \in [0, 1], i = 1, \dots, 30,$$

where $d = 30$ is the number of dimensions (variables). The Pareto optimality is reached when $g = 1$.

To find the Pareto front, the control parameters were set as follows: size of population $NP = 40$ which makes the number of solutions (sn) = 20; Maximum Cycle Number (MCN) = 1000; Scout production period (SPP) = $0.1 \cdot D \cdot NP$; Modification rate (MR) = 0.8, and (D) = 30.

A comparison of the front found by the proposed MOABC with levy flights and the true Pareto front is presented in Figure 5-1. In order to verify the convergence of the proposed algorithm, the ZDT1 problem was solved 30 independent times and the average error between estimated Pareto (PF^e) and true Pareto (PF^t) is calculated using mean squared error approach and presented in Figure 5-2 in both normal and log scale. Figure shows that MOABC with levy flight converge in almost exponential manner. The error is calculated as follows:

$$error = Ef/N, \quad (Eq. 5-2)$$

$$where \ Ef = \|PF^e - PF^t\|^2 = \sum_{j=1}^N (PF_j^e - PF_j^t)^2$$

And N is the number of points in the Pareto front.

5.1.2 Test problem ZDT2 function with a non-convex front

$$f_1(x) = x_1, \quad f_2(x) = g(1 - \frac{f_1}{g})^2, \quad (Eq. 5-3)$$

$$g = 1 + \frac{9 \sum_{i=2}^d x_i}{d-1}, \quad x_i \in [0, 1], i = 1, \dots, 30,$$

where $d = 30$ is the number of dimensions (variables). The Pareto optimality is reached when $g = 1$.

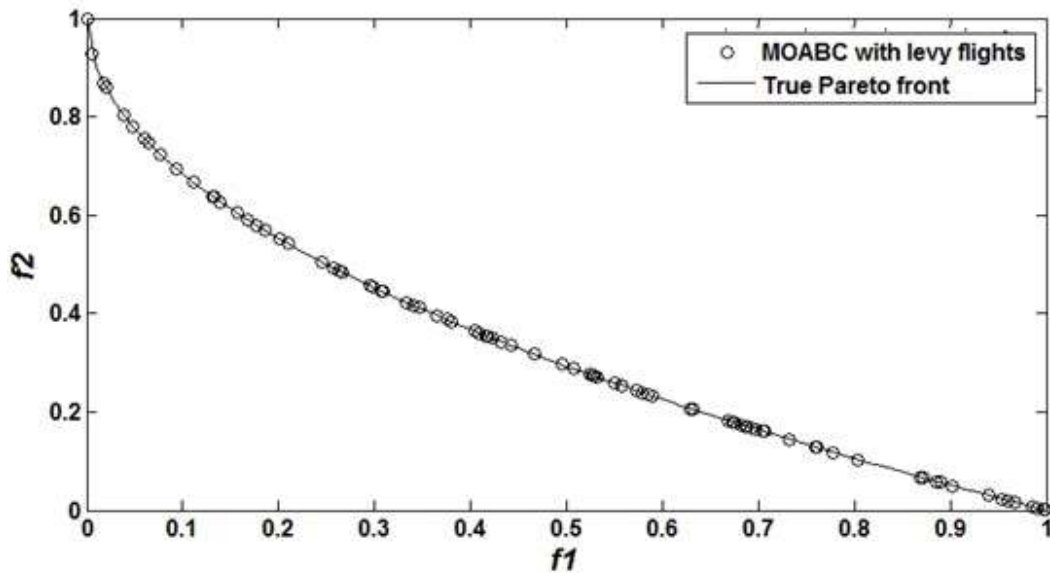


Figure 5-1 Pareto front of ZDT1: MOABC with levy flights and the true front.

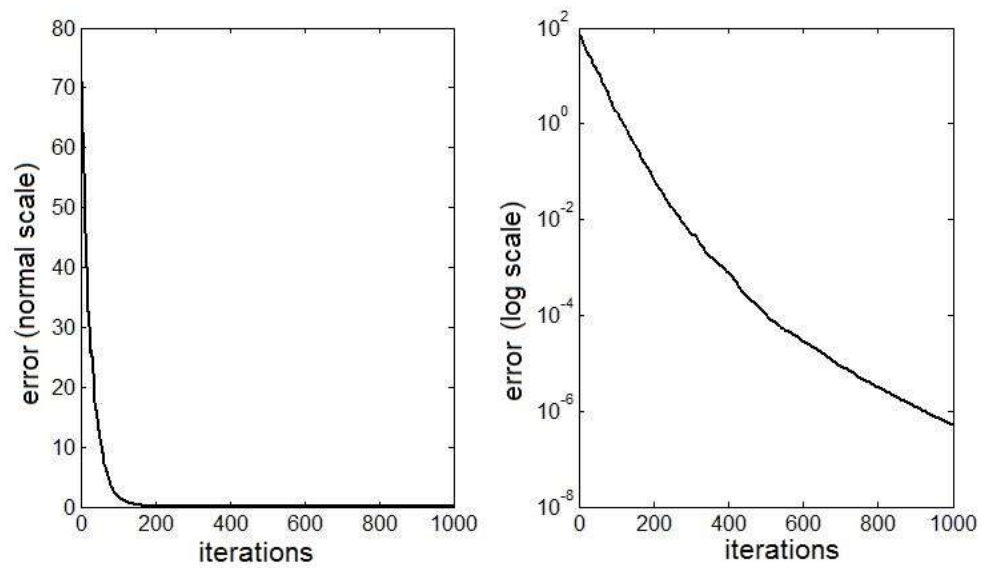


Figure 5-2 Convergence of the proposed MOABC with levy flights for ZDT1

To find the Pareto front, the control parameters were set as follows: size of population $NP = 40$ which makes the number of solutions (sn) = 20; Maximum Cycle Number (MCN) = 1000; Scout production period (SPP) = $0.1 \cdot D \cdot NP$; Modification rate (MR) = 0.8, and (D) = 30.

A comparison of the front found by the proposed MOABC with levy flights and the true Pareto front is presented in Figure 5-3. In order to verify the convergence of the proposed algorithm, the ZDT2 problem was solved 30 independent times and the average error between estimated Pareto (PF^e) and true Pareto (PF^t) is calculated using mean square error approach and presented in Figure 5-4 in both normal and log scale. Figure shows that MOABC with levy flight converge in almost exponential manner.

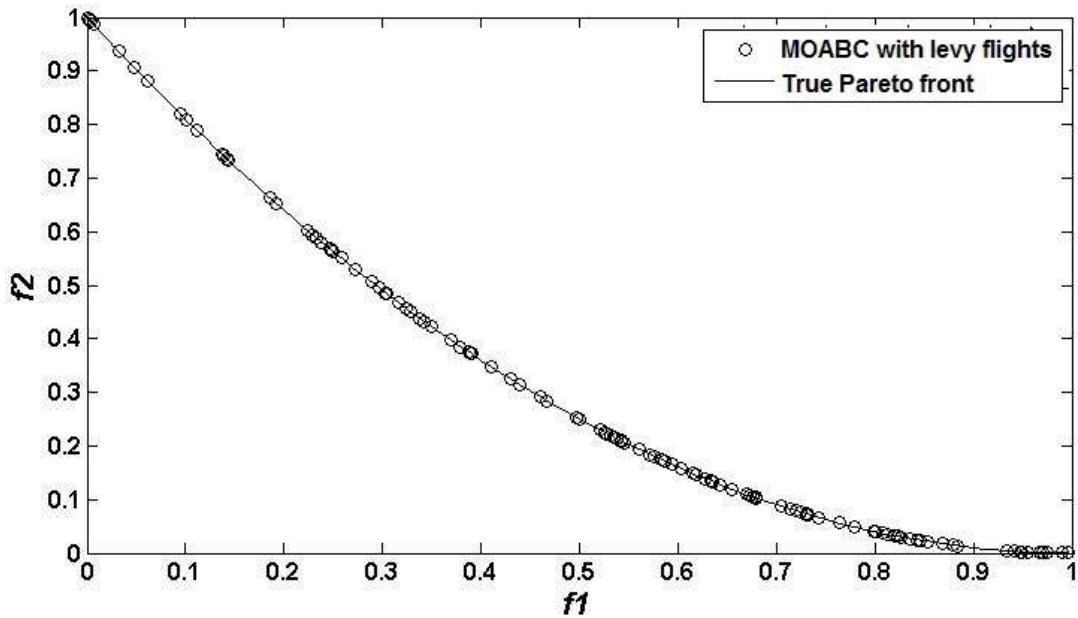


Figure 5-3 Pareto front of ZDT2: MOABC with levy flights and the true front.

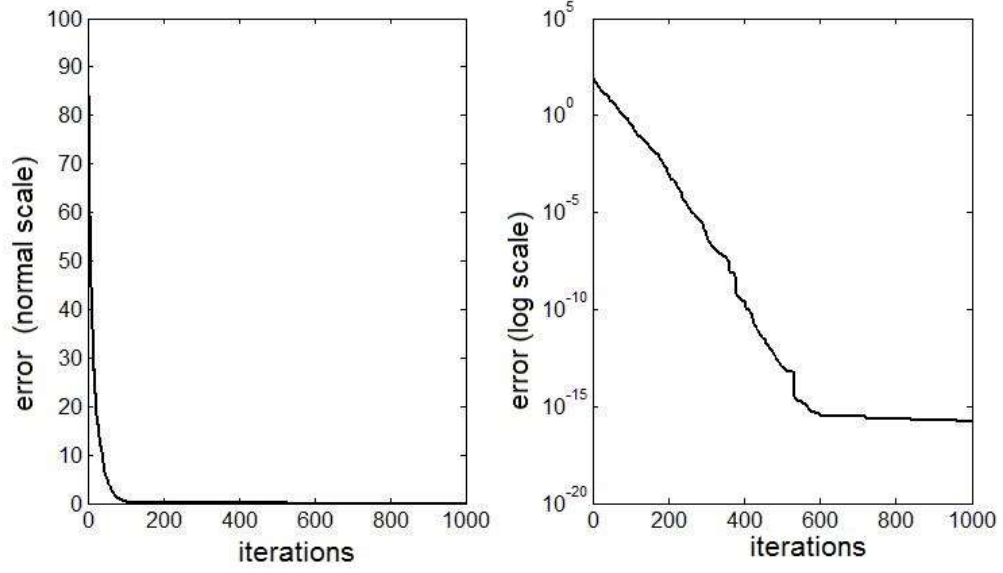


Figure 5-4 Convergence of the proposed MOABC with levy flights for ZDT2

5.1.3 Test problem ZDT3 function with a discontinuous front

$$f_1(x) = x_1, \quad f_2(x) = g \left[1 - \sqrt{\frac{f_1}{g}} - \frac{f_1}{g} \sin(10\pi f_1) \right], \quad (\text{Eq. 5-4})$$

$$g = 1 + \frac{9 \sum_{i=2}^d x_i}{d-1}, \quad x_i \in [0, 1], i = 1, \dots, 30,$$

where $d = 30$ is the number of dimensions (variables). The Pareto optimality is reached when $g = 1$.

To find the Pareto front, the control parameters were set as follows: size of population $NP = 40$ which makes the number of solutions (sn) = 20; Maximum Cycle Number (MCN) = 1000; Scout production period (SPP) = $0.1 \cdot D \cdot NP$; Modification rate (MR) = 0.8, and $D = 30$.

A comparison of the front found by the proposed MOABC with levy flights and the true Pareto front is presented in Figure 5-5. In order to verify the convergence of the proposed algorithm, the ZDT3 problem was solved 30 independent times and the average error between estimated Pareto (PF^e) and true Pareto (PF^t) is calculated using least square approach and presented in Figure 5-6 in both normal and log scale. The figure shows that MOABC with levy flight converge in almost exponential manner.

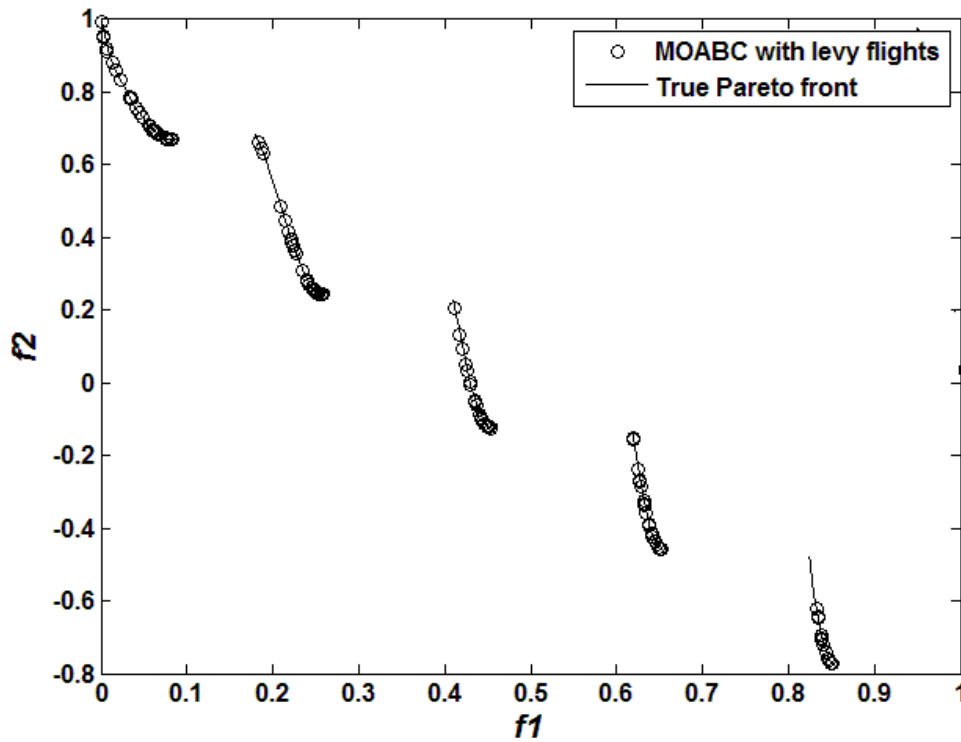


Figure 5-5 Pareto front of ZDT3: MOABC with levy flights and the true front.

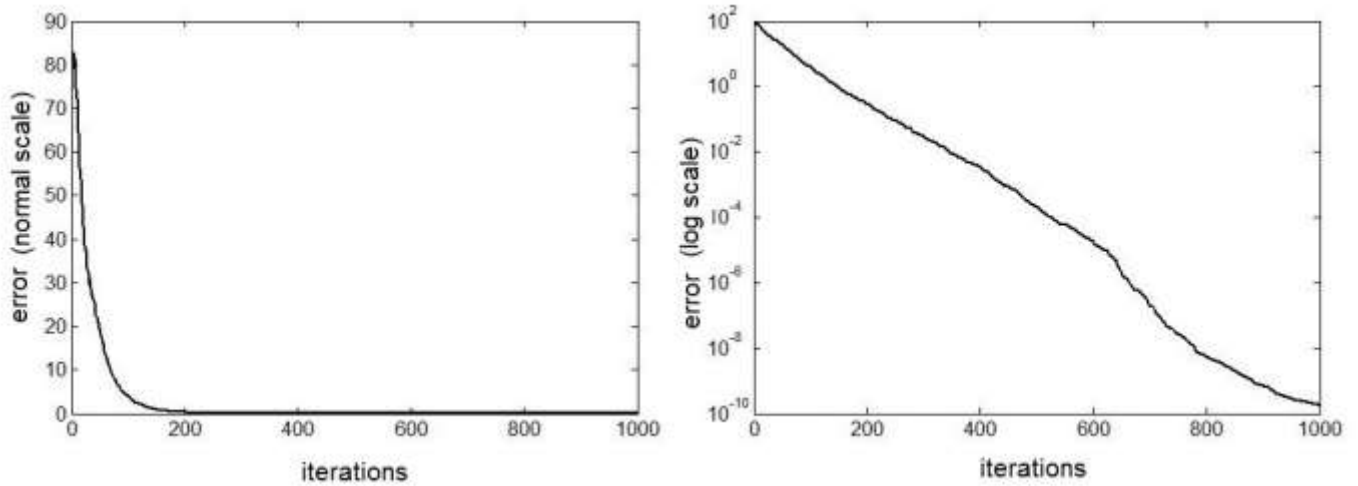


Figure 5-6 Convergence of the proposed MOABC with levy flights for ZDT3

Table 5-1 Comparison of error for 1000 iteration

	ZDT1	ZDT2	ZDT3
MOFA	1.15E-08	4.45E-08	1.85E-07
MOCS	6.00E-09	3.65E-08	1.10E-07
MOABC-basic	5.05E-08	3.89E-03	1.98E-06
MOABC-Levy	5.01E-07	1.56E-16	1.83E-10

5.1.4 Sensitivity of control parameters

In order to examine the effects of the values of control parameters MR, Limit and SPP, the three test problems ZDT1, and ZDT2 are solved with various sets of parameters and the results are presented in illustrative comparison of their convergence.

The results in Figure 5-7 and Figure 5-8 are produced by four levels of $MR = \{0.2, 0.4, 0.6, 0.8\}$. The other control parameters were set as follows: size of population $NP = 40$; $limit = 0.1 * D * NP$; Maximum Cycle Number (MCN) = 1000; Scout production period (SPP) = $0.1 * D * NP$

Similarly, Figure 5-9 and Figure 5-10 are produced by four levels of $limit = \{0.1, 0.5, 1, 2\}$. The other control parameters were set as follows: $MR = 0.6$, size of population $NP = 40$; Maximum Cycle Number (MCN) = 1000; Scout production period (SPP) = $0.1 * D * NP$.

Figure 5-11 and Figure 5-12 are produced by four levels of Scout production period (SPP) = $\{0.1, 0.5, 1, 2\}$. The other control parameters were set as follows: $MR = 0.6$, size of population $NP = 40$; Maximum Cycle Number (MCN) = 1000; $limit = 0.1 * D * NP$.

From the illustrated figures, it is clear that the effects of control parameters varies from problem to another, no clear relationship could be observed, and no best set of their values could be concluded. However, for both test functions we observe that regardless the values of control parameters; the convergence for the first 300 iterations is good and almost same i.e. no significant effects of control parameter are observed in this range of iterations.

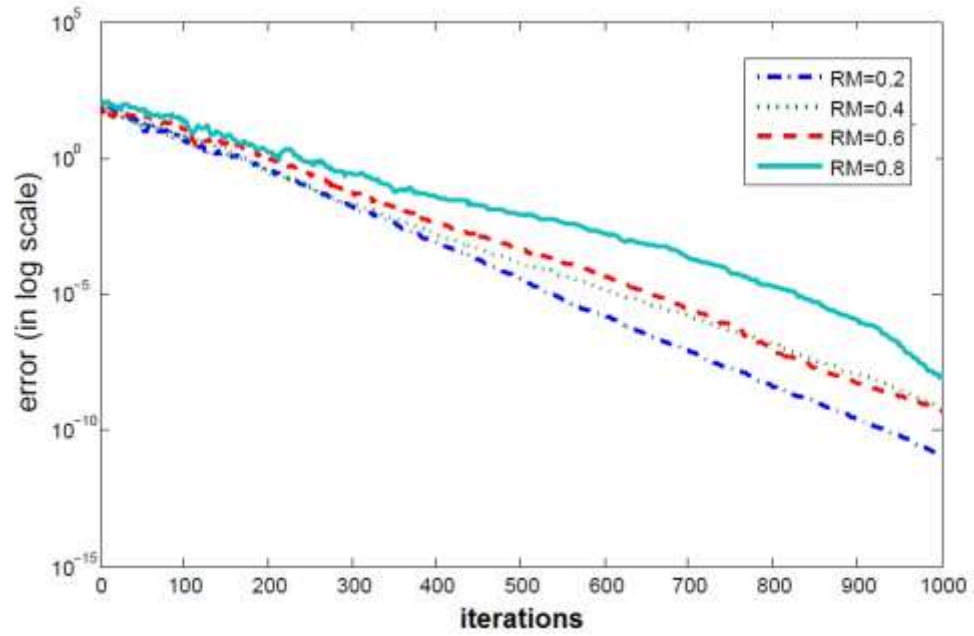


Figure 5-7 ZDT1 convergence at four levels of RM

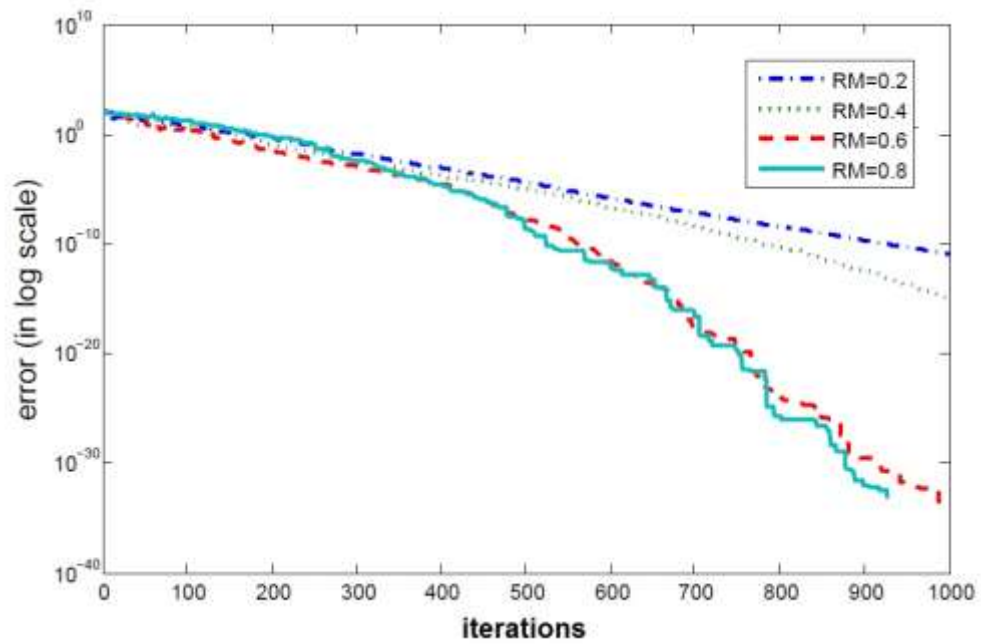


Figure 5-8 ZDT2 convergence at four levels of RM

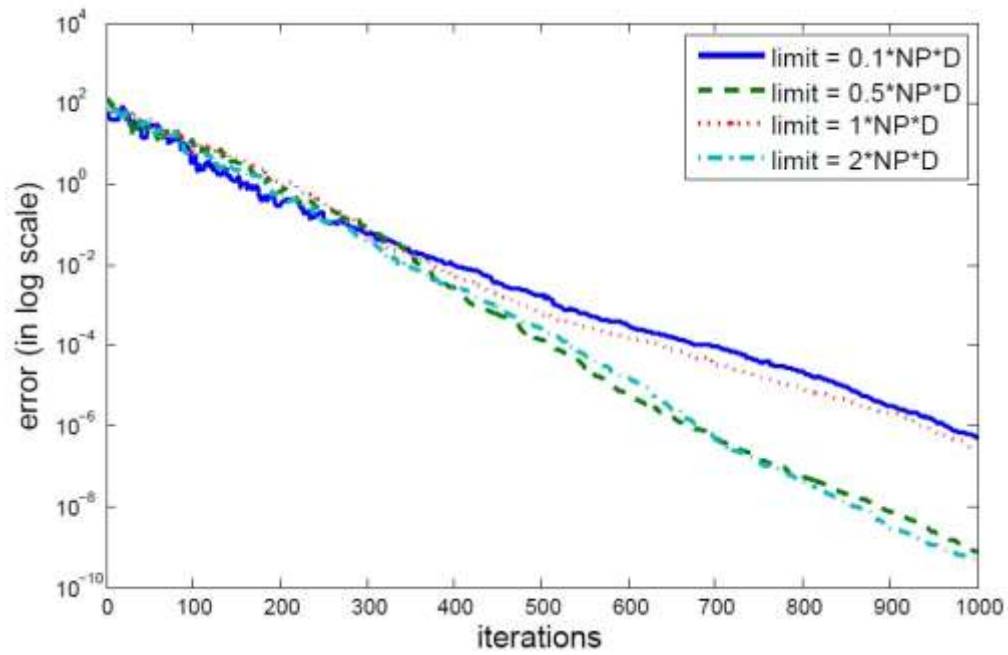


Figure 5-9 ZDT1 convergence at four levels of limit

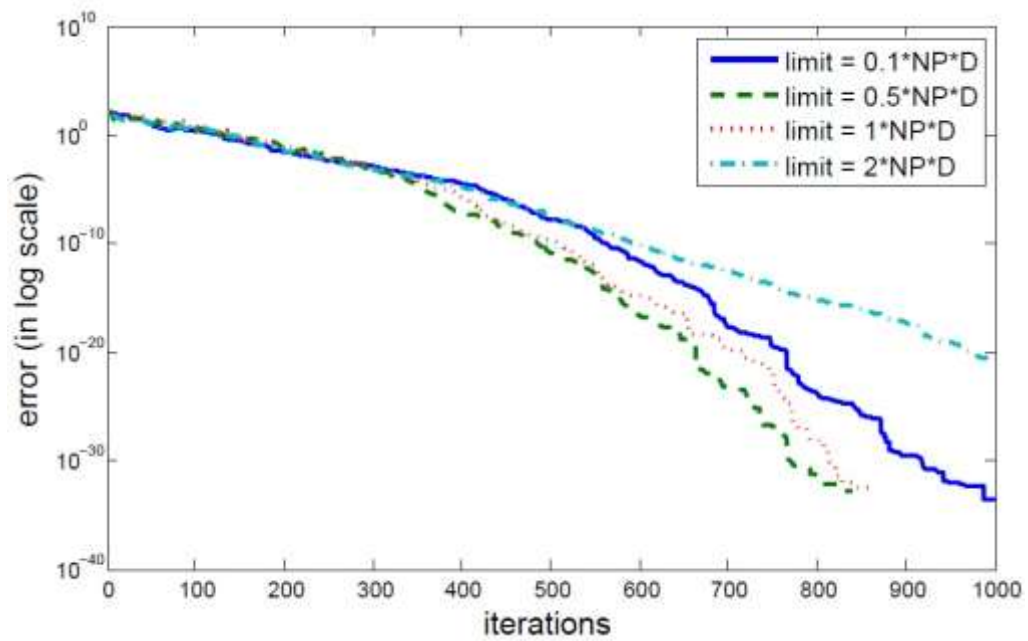


Figure 5-10 ZDT2 convergence at four levels of limit

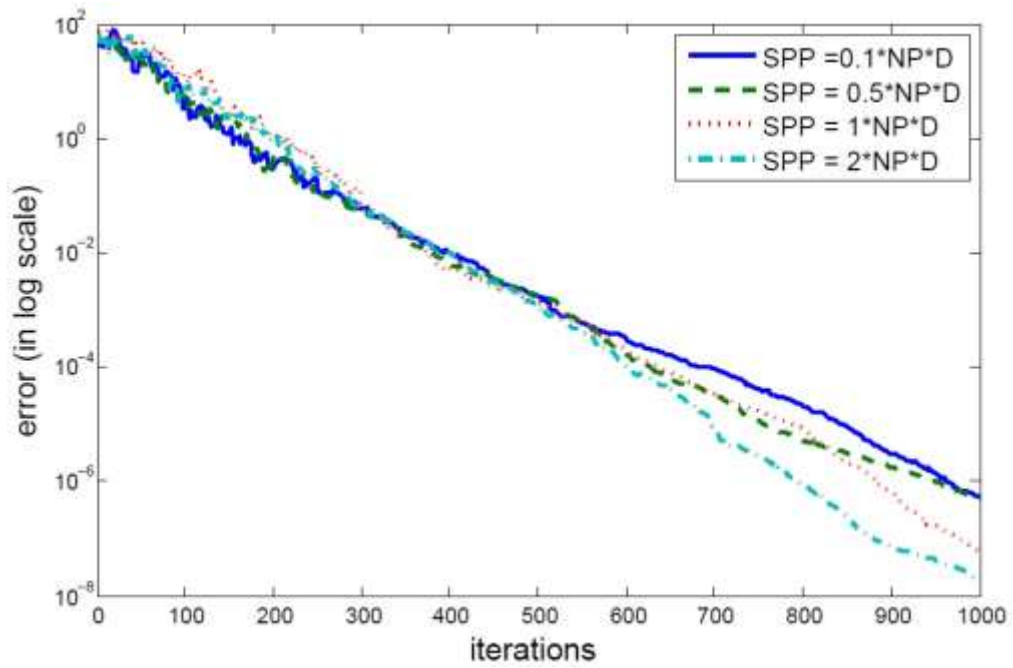


Figure 5-11 ZDT1 convergence at four levels of SPP

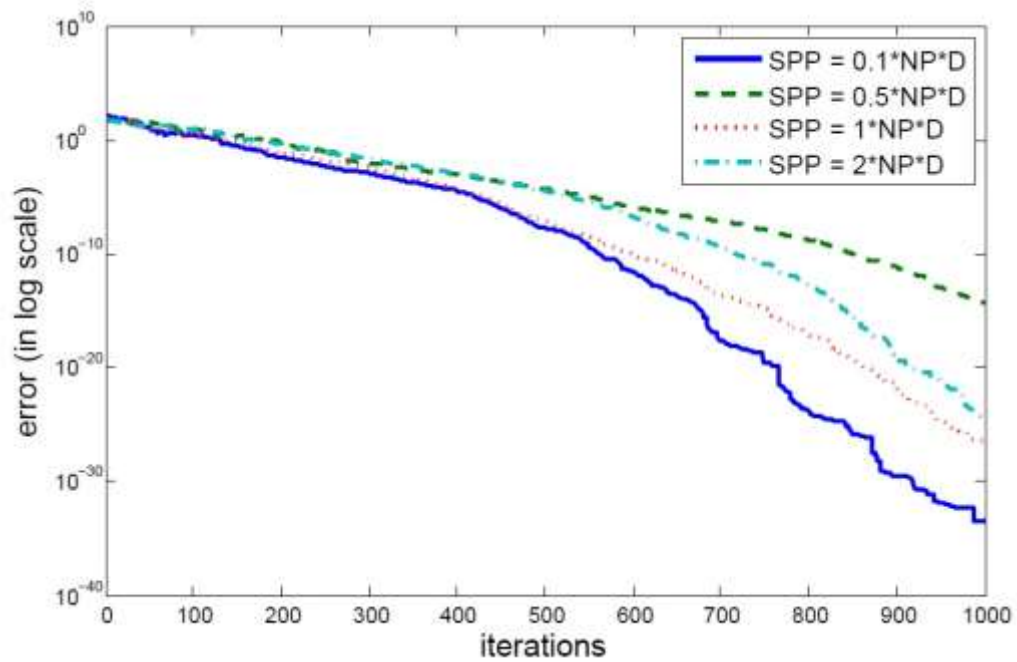


Figure 5-12 ZDT2 convergence at four levels of SPP

5.1.5 Design of a disc brake

Multi-objective design of disc brake is a popular benchmark problem (Ray and Liew, 2002). The problem has four design variables: the inner radius is r , outer radius is R of the disc, the engaging force is F and the number of friction surfaces is S . This is under the design constraints such as the torque, pressure, temperature, and length of the brake. The objectives are to minimize the overall mass and the braking time. The problem is mathematically represented as follows:

$$\text{Minimize} \quad f_1(x) = 4.9 \times 10^{-5}(R^2 - r^2)(s - 1), \quad (\text{Eq. 5-5})$$

$$f_2(x) = \frac{9.82 \times 10^6(R^2 - r^2)}{Fs(R^3 - r^3)}$$

Subject to

$$g_1 = 20 - (R - r) \leq 0,$$

$$g_2 = 2.5 (s + 1) - 30 \leq 0,$$

$$g_3 = \frac{F}{3.14(R^2 - r^2)} - 0.4 \leq 0,$$

$$g_4 = \frac{2.22 \times 10^{-3} F (R^3 - r^3)}{(R^2 - r^2)^2} - 1 \leq 0,$$

$$g_5 = 900 - \frac{0.0266 Fs (R^3 - r^3)}{(R^2 - r^2)} \leq 0,$$

The simple limits are

$$55 \leq r \leq 80, \quad 75 \leq R \leq 110, \quad 1000 \leq F \leq 3000, \quad 2 \leq s \leq 20.$$

To find the Pareto front, the control parameters were set as follows: size of population $NP = 50$ which makes the number of solutions (sn) = 25; Maximum Cycle Number (MCN) = 1000; Scout production period (SPP) = $0.1 \cdot D \cdot NP$; Modification rate (MR) = 0.8, and (D) = 4.

The Pareto front obtained by MOABC with levy flights is shown in Figure 5-13, where we can see that the results are smooth and are the same or better than the results obtained in (Ray and Liew, 2002) and in (Yang 2013).

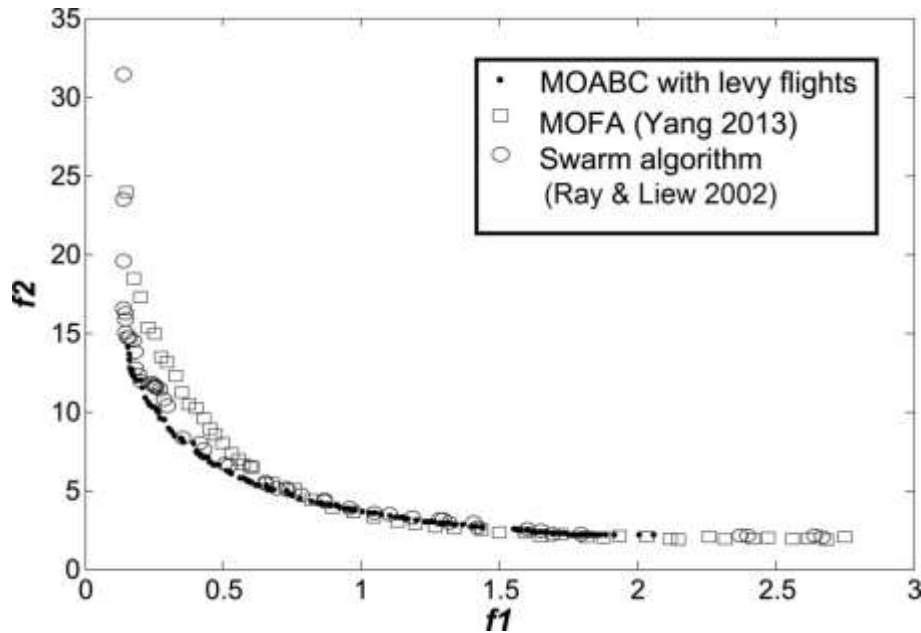


Figure 5-13 Pareto front for the disc brake design

5.1.6 Design of a cantilever beam

Multi-objective design of cantilever beam is another classical benchmark problem (Deb 2001). The problem has two design variables: the diameter d and the length l of the beam as shown in Figure 5-14. The beam has to carry an end load P . The objectives are to minimize the weight and the end deflection of the beam. This is under two design

constraints, these are, the maximum developed stress σ_{max} is less than the allowable strength S_y and the end deflection δ is less than a specified limit δ_{max} . The problem is mathematically represented as follows:

$$\text{Minimize } f_1(x) = \rho \frac{\pi d^2}{4} l, f_2(x) = \delta = \frac{64Pl^3}{3E\pi d^4} \text{ (Eq. 5-6)}$$

Subject to

$$g_1 = \frac{64Pl^3}{3E\pi d^4} \leq \delta_{max},$$

$$g_2 = \sigma_{max} \leq S_y,$$

The simple limits are

$$10 \leq d \leq 50, \quad 200 \leq l \leq 1000, \quad .$$

Where the maximum stress is calculated as follows:

$$\sigma_{max} = \frac{32Pl}{\pi d^3}$$

The parameter values used are as follows:

$$\rho = 7800 \text{ kg/m}^3, \quad P = 1 \text{ kN}, \quad E = 207 \text{ GPa},$$

$$S_y = 300 \text{ MPa}, \quad \delta_{max} = 5 \text{ mm}.$$

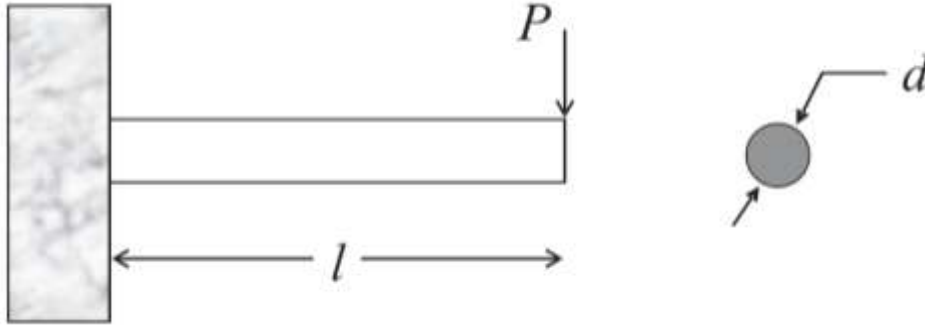


Figure 5-14 Schematic of cantilever beam design (Nakayama , 2005)

To find the Pareto front, the control parameters were set as follows: size of population $NP = 40$ which makes the number of solutions (sn) = 20; Maximum Cycle Number (MCN) = 500; Scout production period (SPP) = $0.1 \cdot D \cdot NP$; Modification rate (MR) = 0.8, and (D) = 2.

The Pareto front obtained by MOABC with levy flights is shown in Figure 5-15, where we can see that the results are smooth and similar to the results obtained by generalized data envelopment analysis (GDEA) (Nakayama 2005), multi-objective genetic algorithm (MOGA) (Nakayama 2005), Elitist Non-dominated Sorting Genetic Algorithm (NSGA II) (Deb 2001).

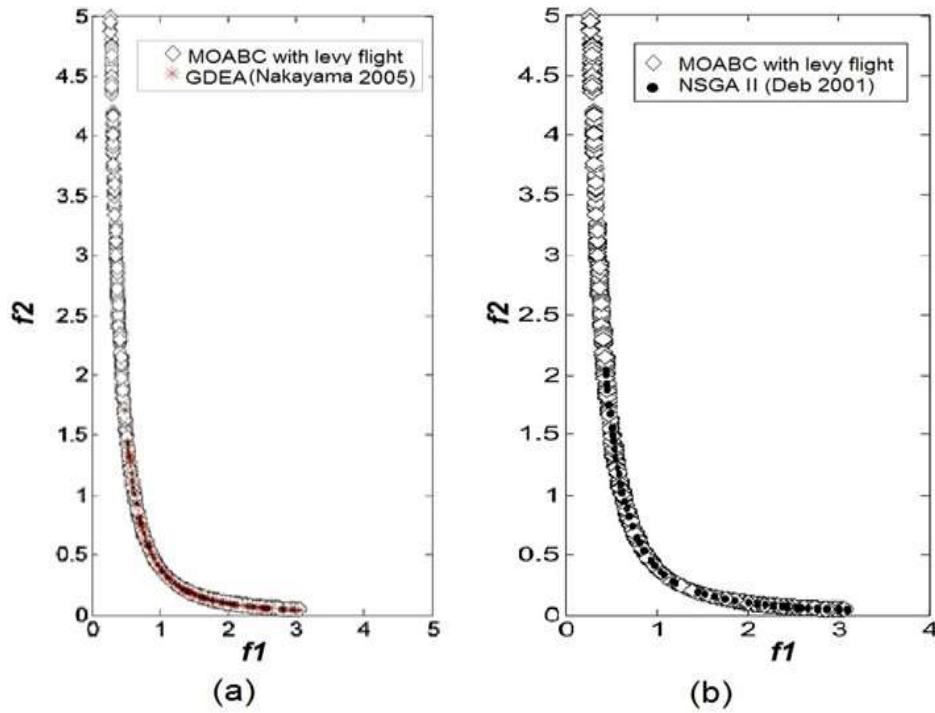


Figure 5-15 Pareto front for the cantilever beam design

5.2 CSLP Case studies

Construction site layout planning model is designed to arrange temporary facilities required at various time intervals (phases) of construction project under set of constraints while achieving multi objective functions. Generally in multi-objective optimization problems, pareto optimal solutions are determined so that decision makers can choose their preferred plan among these pareto optimal solutions (Jiuping Xu and Zongmin Li 2012). The efficiency of CSLP model will be significantly affected by its precision in representing the actual situations.

In this research the construction site layout planning problem has been modeled to be more representative to the actual situation. The proposed MOABC based methodology is used to obtain the solution of the dynamic construction site unequal-area layout problem under multi-objective functions in order to demonstrate its efficiency.

5.2.1 Case study 1

In this case study the performance of the MOABC based method is compared with that of a MMAS-based method which is one of the ant colony optimization (ACO) algorithms (Ning, 2008). The residential building project is located in the city of Beijing, China. There are four residential buildings (Building 1 and Building 4 are sixty stories high; Building 2 and Building 3 are eighty stories high), site offices and construction operational facilities located within the construction site. Other temporary living facilities are located outside the construction site. The task of the proposed model is to find the corresponding locations for the temporary construction operational facilities in the construction site. The temporary facilities (fixed and non-fixed facilities) and their corresponding dimensions, serviced phase are shown in ed vertically or horizontally.

Table 5-2. In Figure 5-16, the site layout plan used in the original contractor's plan document is shown. The main works carried out in each construction phase are foundation and underground works (Phase 1), superstructure (Phase 2), finishing works and external works (Phase 3).

The benchmark problem is solved using the proposed MOABC algorithms. The travel distance between two facilities in cost related function is measured using the modified rectangular, the presence of obstructions are considered, dimensions are measured at scale 1:5 and the facilities can be aligned vertically or horizontally.

Table 5-2 Site facilities and their dimensions of case study 1

No.	Temporary and fixed facilities	Facility type	Dimension (m ²)	Service phase
F1	Sample room	non-fixed	25	3
F2	Equipment maintenance plant	non-fixed	25	1,2,3
F3	Electrician hut	non-fixed	25	1,2,3
F4	Tool shed	non-fixed	50	1,2,3
F5	Rebar bending yard	non-fixed	100	1,2,3
F6	Carpentry workshop	non-fixed	100	1,2
F7	#1 Material laydown area	non-fixed	100	1,2,3
F8	#2 Material laydown area	non-fixed	100	1,2,3
F9	#1 Tower crane	fixed	50	1,2
F10	#2 Tower crane	fixed	50	1,2
F11	#1 Material hoist	fixed	25	2,3
F12	#2 Material hoist	fixed	25	2,3
F13	#3 Material hoist	fixed	25	2,3
F14	#4 Material hoist	fixed	25	2,3
F15	Site office	fixed	50	1,2,3
F16	Security hut at site entrance	fixed	9	1,2,3
F17	Security hut at site exit	fixed	9	1,2,3

5.2.1.1 Decision Variables

(x_i, y_i) are the Cartesian coordinates of the centroid of facility i respectively are selected as design variables, where $i = 1, 2, \dots, 17$. R_i is a binary digit represent the orientation of facility i , i.e. $R_i = 0$ if facility i aligned vertically, and $R_i = 1$ if facility i aligned horizontally, where $i = 1, 2, \dots, 17$.

5.2.1.2 Objective Functions

The objective in this case study is to minimize the total handling cost of interaction flows between facilities (f_1) and to minimize safety hazards /environmental concerns (f_2) . The objective functions are mathematically defined as follows:

$$\text{Min } f_1 = 2(\sum_{p=1}^3 \sum_{i=1}^{16} \sum_{j=i+1}^{17} C_{ijp} r_{ij} P_{ijp}) \quad (\text{Eq. 5-7})$$

$$\text{Min } f_2 = 2(\sum_{p=1}^3 \sum_{i=1}^{16} \sum_{j=i+1}^{17} SE_{ijp} \frac{1}{e_{ij}} P_{ijp}) \quad (\text{Eq. 5-8})$$

5.2.1.3 Facility Closeness Relationship

The facilities closeness relationship for the total relationship C_{ijp} and the safety/Environmental concerns SE_{ijp} in the three phases are documented in Appendix A.

5.2.1.4 Overlapping constrains

$$\min \{0.5(b_i + b_j) + h_{ij} - |x_i - x_j|, 0.5(l_i + l_j) + v_{ij} - |y_i - y_j|\} \leq 0 \quad (\text{Eq. 5-9})$$

$$i = 1, 2, \dots, 17; j = 2, 3, \dots, 21; i \neq j; \text{ and } i < j. \quad ;$$

Where $i = 1, 2, \dots, 17$ refers to temporary facilities and $i = 18, 19, 20, 21$ refers to the four buildings. And $j = 1, 2, \dots, 17$ refers to temporary facilities and $j = 18, 19, 20, 21$ refers to the four buildings. To maintain a minimum distance of 3 meters between buildings under construction and any temporary facilities, h_{ij} and v_{ij} are set to 3/5 for any i or $j = 18, 19, 20$ or 21 .

5.2.1.5 Site boundary constrains (BC):

$$\text{BC(1): } 14 - 1.4 \left(x_i - \frac{b_i}{2} \right) - \left(y_i - \frac{l_i}{2} \right) \leq 0 \quad (\text{Eq. 5-9})$$

$$\text{BC(2): } \left(-y_i + \frac{l_i}{2} \right) + 2 \leq 0 \quad (\text{Eq. 5-10})$$

$$\text{BC(3): } -19.25 + 1.25 \left(x_i + \frac{b_i}{2} \right) - \left(y_i - \frac{l_i}{2} \right) \leq 0 \quad (\text{Eq. 5-11})$$

$$\text{BC(4): } \left(x_i + \frac{b_i}{2} \right) - 21 \leq 0 \quad (\text{Eq. 5-12})$$

$$\text{BC(5)a: } -16 + \left(x_i + \frac{b_i}{2} \right) \leq 0, \quad \text{if} \quad \left(y_i + \frac{l_i}{2} \right) > 24,$$

$$\text{BC(5)b: } 3 \left(x_i + \frac{b_i}{2} - 17 \right) + \left(y_i + \frac{l_i}{2} - 21 \right) \leq 0, \quad 24 \geq \left(y_i + \frac{l_i}{2} \right) \geq 21,$$

$$\text{BC(5)c: } 2 \left(x_i + \frac{b_i}{2} - 17 \right) + \left(y_i + \frac{l_i}{2} - 21 \right) \leq 0, \quad 21 \geq \left(y_i + \frac{l_i}{2} \right) \geq 19,$$

$$\text{BC(5)d: } \frac{4}{3} \left(x_i + \frac{b_i}{2} - 18 \right) + \left(y_i + \frac{l_i}{2} - 19 \right) \leq 0, \quad \left(y_i + \frac{l_i}{2} \right) \leq 19, (\text{Eq. 5-13})$$

$$\text{BC(6): } \left(y_i + \frac{l_i}{2} \right) - 25 \leq 0 \quad (\text{Eq. 5-14})$$

$$\text{BC(7)a: } 4 - \left(x_i - \frac{b_i}{2} \right) \leq 0, \quad \text{if} \quad \left(y_i + \frac{l_i}{2} \right) < 14,$$

$$\text{BC(7)b: } 8 - \left(x_i - \frac{b_i}{2} \right) \leq 0 \quad \text{if} \quad 14 \leq \left(y_i + \frac{l_i}{2} \right) \leq 15$$

$$\text{BC(7)c: } 7 - \left(x_i - \frac{b_i}{2} \right) \leq 0 \quad \text{if} \quad \left(y_i + \frac{l_i}{2} \right) > 15, \quad (\text{Eq. 5-15})$$

(x_j, y_j) are positive real numbers; $i = 1, 2, \dots, 17$.

5.2.1.6 Tower crane constraints

The two tower cranes are with 45 meter reachable radius. Rebar bending yards (F5), carpentry workshop (F6), #1 material laydown area (F7), and #2 material laydown area (F8), have to be positioned within the reachable radius of one of the two tower cranes.

$$\min \left\{ \sqrt{(FCX_i - TX_1)^2 + (FCY_i - TY_1)^2} - \frac{45}{5}, \sqrt{(FCX_i - TX_2)^2 + (FCY_i - TY_2)^2} - \frac{45}{5} \right\} \leq 0;$$

$$i = 5, 6, 7, 8. \quad (\text{Eq. 5-16})$$

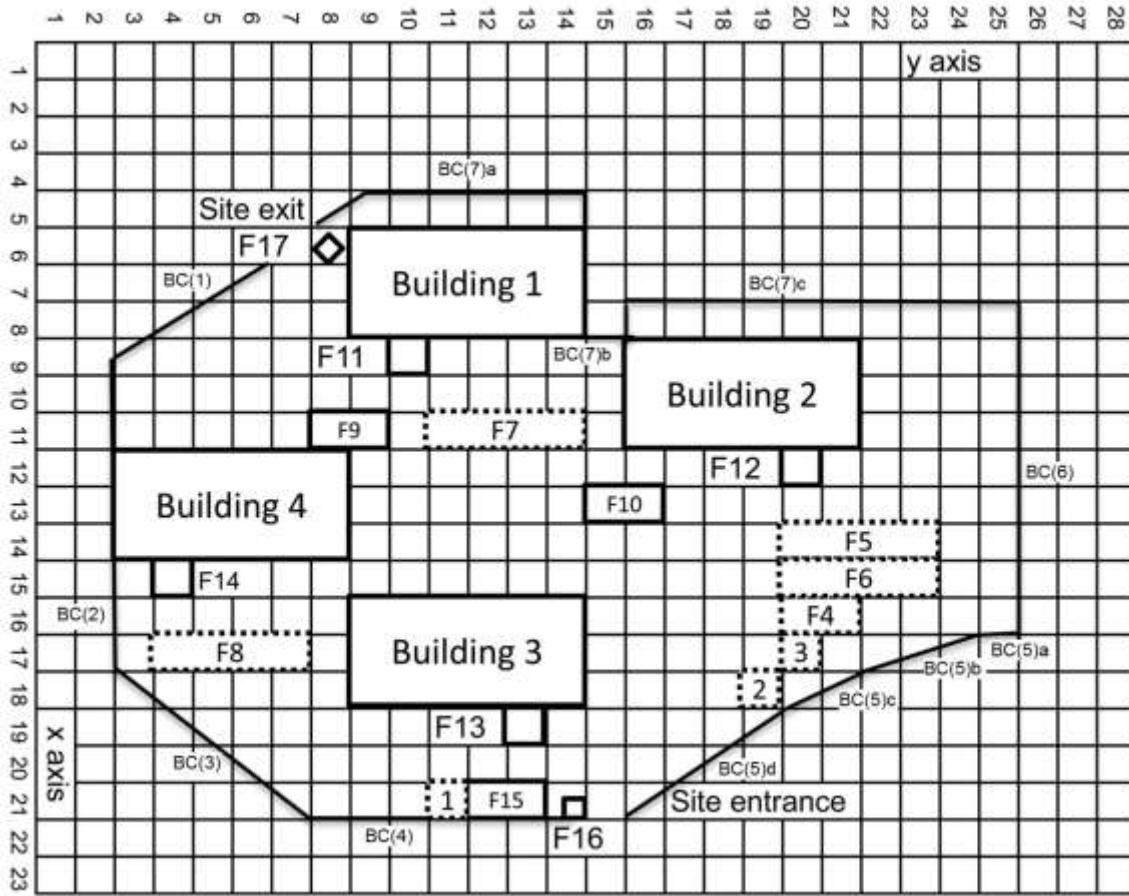


Figure 5-16 Original construction site layout for case study 1

5.2.1.7 Results and discussions

Two computer programs are prepared in MATLAB one for Basic-MOABC and the other for MOABC via Levy Flights in order to find the solution of the optimal layout design for the example considered. The control parameters were set as follows: size of population $NP = 26$; Maximum Cycle Number (MCN) = 3000; Scout production period (SPP) = $0.1 \cdot D \cdot NP$; Modification rate (MR) = 0.8. In this case study there are 8 non-fixed facilities and 5 rectangular facilities, therefore the number of parameters (D) = $8 \times 2 + 5 = 21$.

The results of optimal CSLP alternatives (pareto fronts) generated by both Basic-MOABC model and MOABC via Levy Flights are presented in Figure 5-17. Each point in a pareto line represents the values for the two objective functions of a pareto optimal layout alternative. For comparison and evaluation the objective functions for the original site layout and a pareto front of MMAS (Ning, 2008) that has only one solution are calculated and shown in Figure 5-17 as well. The alternatives generated by MOABC via Levy Flights are significantly larger in number and wider in range than those generated by the Basic-MOABC and MMAS. The alternatives generated by MOABC via Levy Flights are dominating all other alternatives. Three alternatives from the thirty alternatives of MOABC via Levy Flights are numerically compared with the original layout in Table 5-3. Alternative L1, illustrated in Figure 5-18, has 16.5% reduction rate in handling cost, and 3.3% reduction rate in safety concerns compared to the original site layout. Alternative L2, illustrated in Figure 5-19 has 5.7% reduction rate in handling cost and 14.4% reduction rate in safety concerns. Alternative has 17.9 reduction rate in safety concerns and the handling cost is approximately same as the original site layout. Also shown in Table 5-3, the MMAS alternative reduces the handling cost by 14.2% and 4.1% reduction rate in safety concerns. MMAS produced single solution which is dominated by MOABC with levy flights.

Table 5-3 Comparison with original site layout of case study 1

Layout	F1	F2	$\Delta\%$ for F1	$\Delta\%$ for F2
Original Layout (OL)	102040	1691	-	-
L1	85216	1635	-16.5	-3.3
L2	96211	1448	-5.7	-14.4
L3	102046	1388	0.0	-17.9
MMAS	87534	1622	-14.2	-4.1

Note: $\Delta\%$ for F1 = $(F1(L1) - F1(OL))/F1(OL)$, $\Delta\%$ for F2 = $(F2(L1) - F2(OL))/F2(OL)$

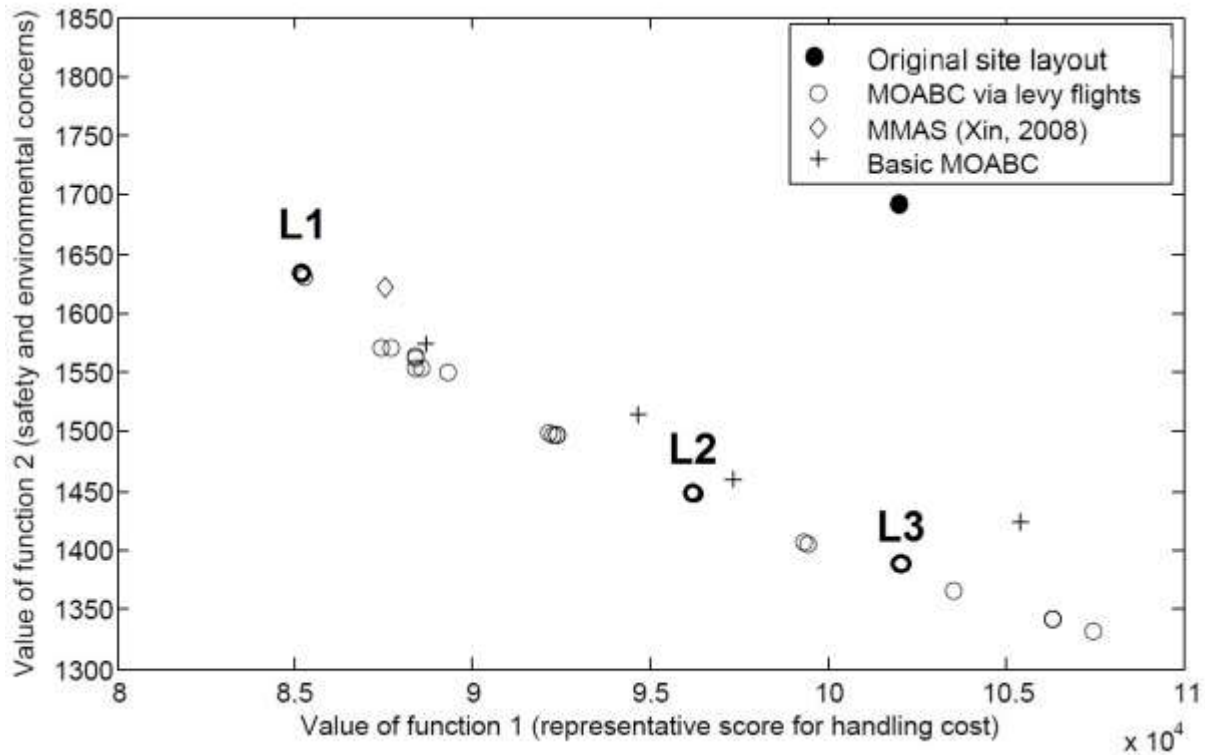


Figure 5-17 Pareto front for the CSLP problem of case 1

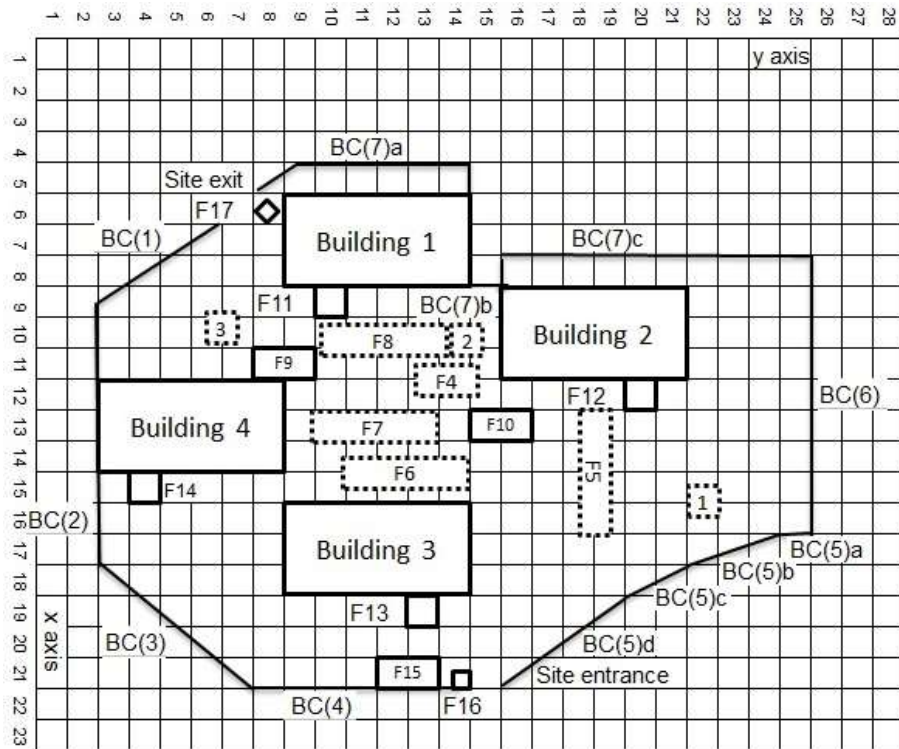


Figure 5-18 Construction site layout alternative L1 for case study 1

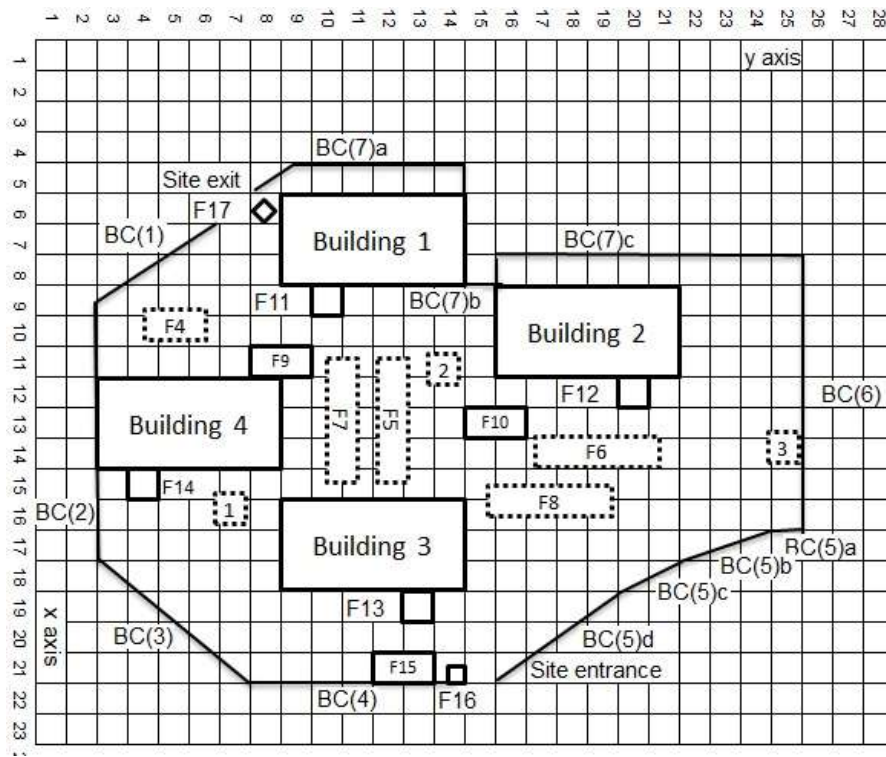


Figure 5-19 Construction site layout alternative L2 for case study 1

5.2.2 Case study 2

A private hospital project that is located in the city of Riffa, Bahrain is considered as a second case study. The building is three stories high. Site offices and construction operational facilities are located within the construction site. Other temporary living facilities are located outside the construction site. The task of the proposed model is to find the corresponding locations for the temporary construction operational facilities in the construction site. The temporary facilities (fixed and non-fixed facilities) and their corresponding dimensions and serviced phases are shown in Table 5-4. Figure 5-20 shows the site layout plan used in the original contractor's plan document. The main works carried out in each construction phase are foundation and underground works (Phase 1), superstructure (Phase 2), finishing works and external works (Phase 3).

The problem is solved using the proposed MOABC with levy flights algorithm. The travel distance between two facilities in cost related function is measured using the modified rectangular distance, the presence of obstructions is considered, and the facilities can be aligned vertically or horizontally.

5.2.2.1 Decision Variables

(x_i, y_i) represent the Cartesian coordinates of the centroid of facility i which are selected as design variables, where $i = 1, 2, \dots, 15$. R_i is a binary digit represent the orientation of facility i , i.e. $R_i = 0$ if facility i aligned vertically, and $R_i = 1$ if facility i aligned horizontally, where $i = 1, 2, \dots, 15$.

5.2.2.2 Objective Functions

Minimize the total handling cost of interaction flows between facilities (f_1) and minimize safety hazards /environmental concerns (f_2) . The objective functions are mathematically defined as follows:

$$\text{Min } f_1 = 2 \left(\sum_{p=1}^3 \sum_{i=1}^{14} \sum_{j=i+1}^{15} C_{ijp} r_{ij} P_{ijp} \right) \quad (\text{Eq. 5-17})$$

$$\text{Min } f_2 = 2 \left(\sum_{p=1}^3 \sum_{i=1}^{14} \sum_{j=i+1}^{15} \frac{-1}{SE_{ijp}} e_{ij} P_{ijp} \right) \quad (\text{Eq. 5-18})$$

5.2.2.3 Facility Closeness Relationship

Based on input from the project manager, the proximity weight matrices were developed. Table 5-5 presents proximity relationships for layout cost for phase 1. Table 5-6 presents proximity relationships for safety and environmental concerns for phase 1. The values for the proximity weights follow the six-value scale previously presented in Table 3-1. These matrices were then transformed into their quantitative equivalents before being input to the automated system.

The facilities closeness relationship for the layout cost relationship C_{ijp} and the safety and environmental concerns SE_{ijp} in the three phases are recorded in Appendix B

Table 5-4 Site facilities and their dimensions of case study 2

No.	Temporary and fixed facilities	Facility type	Dimension (m²)	Service phase
F1	Steel yard	non-fixed	25x12	1,2
F2	Carpentry	non-fixed	25x12	1,2,3
F3	Mixer machine yard	non-fixed	20x12	1,2
F4	Material laydown area	non-fixed	25x12	1,2,3
F5	Tool shed	non-fixed	12x9	1,2,3
F6	Diesel generator	non-fixed	10x4	1,2,3
F7	Sample room	fixed	10x10	3
F8	Rest room	fixed	12.5x12.5	1,2,3
F9	Contractor office	fixed	9x12	1,2,3
F10	PM office	fixed	6x12	1,2,3
F11	Material hoist	fixed	5x10	2,3
F12	#1 Tower crane	fixed	5x5	1,2
F13	#2 Tower crane	fixed	5x5	1,2
F14	Security hut at site entrance	fixed	3x3	1,2,3
F15	Security hut at site exit	fixed	3x3	1,2,3

Table 5-5 Proximity relationships for layout cost for phase 1

	F1	F2	F3	F4	F5	F6	F8	F9	F10	F12	F13	F14	F15
F1	-	A	U	U	I	E	I	O	U	A	X	I	I
F2	A	-	X	U	I	I	I	O	U	A	X	I	I
F3	U	X	-	A	O	I	O	O	U	X	A	I	I
F4	U	U	A	-	O	U	U	O	U	X	A	E	E
F5	I	I	O	O	-	U	U	U	U	O	O	U	U
F6	E	I	I	U	U	-	O	O	O	E	E	U	U
F8	I	I	O	U	U	O	-	U	U	U	U	U	U
F9	O	O	O	O	U	O	U	-	A	X	X	O	O
F10	U	U	U	U	U	O	U	A	-	X	X	O	O
F12	A	A	X	X	O	E	U	X	X	-	I	U	U
F13	X	X	A	A	O	E	U	X	X	I	-	U	U
F14	I	I	I	E	U	U	U	O	O	U	U	-	O
F15	I	I	I	E	U	U	U	O	O	U	U	O	-

Table 5-6 proximity relationships for safety and environmental concerns for phase 1

	F1	F2	F3	F4	F5	F6	F8	F9	F10	F12	F13	F14	F15
F1	-	I	I	I	I	A	A	A	A	O	O	E	E
F2	I	-	O	I	I	A	A	A	A	O	O	E	E
F3	I	O	-	O	O	I	E	A	A	O	O	I	I
F4	I	I	O	-	O	A	E	A	A	E	E	A	A
F5	I	I	O	O	-	I	I	I	I	U	U	O	O
F6	A	A	I	A	I	-	A	A	A	I	I	A	A
F8	A	A	E	E	I	A	-	U	U	I	I	A	A
F9	A	A	A	A	I	A	U	-	U	I	I	E	E
F10	A	A	A	A	I	A	U	U	-	I	I	E	E
F12	O	O	O	E	U	I	I	I	I	-	A	E	E
F13	O	O	O	E	U	I	I	I	I	A	-	E	E
F14	E	E	I	A	O	A	A	E	E	E	E	-	E
F15	E	E	I	A	O	A	A	E	E	E	E	E	-

5.2.2.4 Overlapping constrains

$$\min \{0.5(b_i + b_j) + h_{ij} - |x_i - x_j|, 0.5(l_i + l_j) + v_{ij} - |y_i - y_j|\} \leq 0 \text{ (Eq. 5-19)}$$

$i = 1, 2, \dots, 15; j = 2, 3, \dots, 16; i \neq j; \text{ and } i < j.$

Where $i = 1, 2, \dots, 15$ refer to the 15 temporary facilities and $i = 16$ for the building.

Similarly $j = 1, 2, \dots, 15$ refer to the temporary facilities and $j = 16$ refer to the building. To

maintain a minimum distance of 3 meters between the building under construction and any temporary non-fixed facilities, h_{ij} and v_{ij} are set to 3 for any i or $j = 16$.

5.2.2.5 Site boundary constrains (BC):

$$BC(1): 115 - 6.571 \left(x_i - \frac{b_i}{2} \right) - \left(y_i - \frac{l_i}{2} \right) \leq 0 \quad (\text{Eq. 5-20})$$

$$BC(2): \left(-y_i + \frac{l_i}{2} \right) \leq 0 \quad (\text{Eq. 5-21})$$

$$BC(3): -1067.625 + 6.571 \left(x_i + \frac{b_i}{2} \right) - \left(y_i - \frac{l_i}{2} \right) \leq 0 \quad (\text{Eq. 5-22})$$

$$BC(4): \left(y_i + \frac{l_i}{2} \right) - 115 \leq 0 \quad (\text{Eq. 5-23})$$

(x_j, y_j) are positive real numbers; $i = 1, 2, \dots, 6$.

5.2.2.6 Tower crane constraints

In the site there are two tower cranes (F12 and F13) which are with 45 meter reachable radius. Steel yards (F1), carpentry workshop (F2), have to be positioned within the reachable radius of (F12) where the mixer machine yard (F3), and material laydown area (F4), have to be positioned within the reachable radius of (F13). Accordingly the tower crane constraints are represented as follows.

For Tower crane no.1 (F12):

$$\sqrt{(FCX_i - TX_1)^2 + (FCY_i - TY_1)^2} - 45 \leq 0 \quad i = 1, 2. \quad (\text{Eq. 5-24})$$

For Tower crane no.2 (F13):

$$\sqrt{(FCX_i - TX_2)^2 + (FCY_i - TY_2)^2} - 45 \leq 0 \quad i = 3, 4. \quad (\text{Eq. 5-25})$$

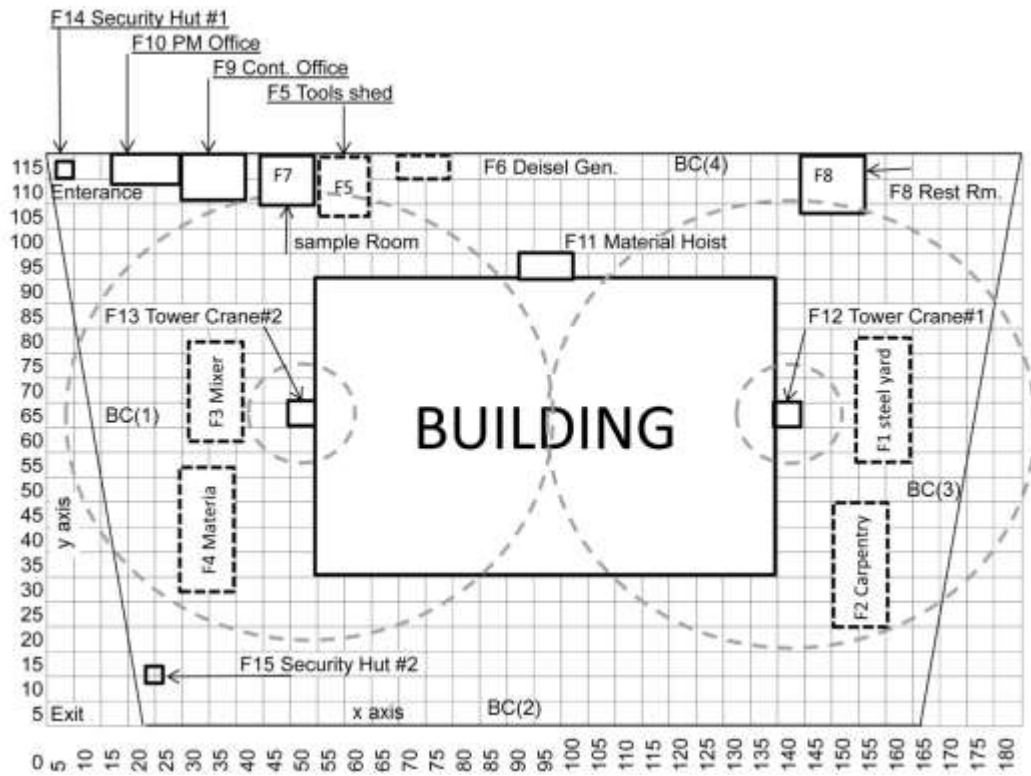


Figure 5-20 Original construction site layout for case study 2

5.2.2.7 Results and discussions

MOABC via Levy Flights algorithm is used in order to find the solution of the optimal layout design of the second case study considered. The control parameters were set as follows: size of population $NP = 50$; Maximum Cycle Number (MCN) = 1500; Scout production period (SPP) = $0.1 \cdot D \cdot NP$; Modification rate (MR) = 0.8. In this case study there are 6 non-fixed rectangular facilities, therefore the number of parameters(D) = $6 \times 2 + 6 = 18$. The matlab code is recorded in Appendix C.

The results of optimal CSLP alternatives (pareto fronts) generated by MOABC via Levy Flights are presented in Figure 5-22. Each point in a pareto line represents the values for the two objective functions of a pareto optimal layout alternative. For

comparison and evaluation the objective functions for the original site layout are calculated and shown in the same figure as well. The alternatives generated by MOABC via Levy Flights are large in number, wide in range, and dominating the original layout. Three alternatives from the 25 alternatives of MOABC via Levy Flights are numerically compared with the original layout in

Table 5-7 and Figure 5-22. Alternative L1, illustrated in Figure 5-21, has 12.6% reduction rate in handling cost, and the safety concerns is slightly increased by just 1% higher than the original site layout. Alternative L2, illustrated in Figure 5-23, has 7.9% reduction rate in handling cost and 4.4% reduction rate in safety concerns of the original site layout. Alternative L3, illustrated in Figure 5-24 has slight increment by just 0.1% in handling cost, and 9.9% reduction rate in safety concerns compared to the original site layout.

Table 5-7 Three solutions for CSLP problem for case study 2

Alternative 1			Alternative 2			Alternative 3		
x_i	y_i	R_i	x_i	y_i	R_i	x_i	y_i	R_i
152.2398	67.0193	0	152.6132	51.5591	0	156.9361	69.2743	0
151.5839	48.3682	1	152.9479	71.0472	1	155.5965	44.0191	1
34.2486	52.0518	1	33.214	46.3267	1	34.0961	45.7724	1
26.8614	64.076	1	32.247	62.1412	1	27.6557	58.777	1
70.3498	100.4307	1	114.5056	100.399	0	126.7361	99.1225	0
106.1234	95.6041	1	126.428	19.6214	0	118.8911	4.1026	1

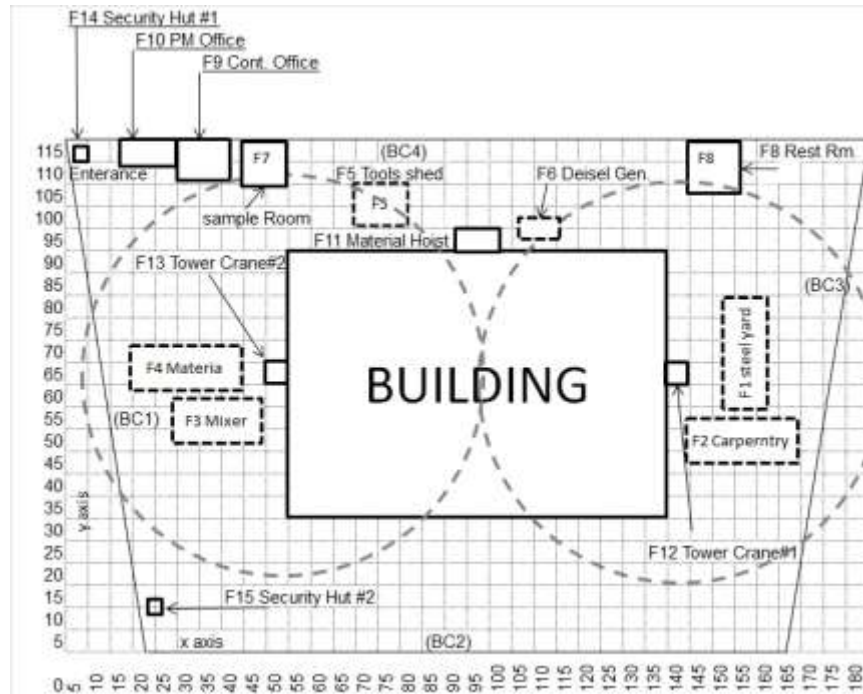


Figure 5-21 Construction site layout alternative L1 for case study 2

Table 5-8 Comparison with original site layout of case study 2

Layout	F1	F2	$\Delta\%$ for F1	$\Delta\%$ for F2
Original Layout (OL)	316875	203.777	-	-
L1	277081	205.909	-12.6	1.0
L2	291966	194.868	-7.9	-4.4
L3	317075	183.668	0.1	-9.9

Note: $\Delta\%$ for F1 = $(F1(L1) - F1(OL))/F1(OL)$, $\Delta\%$ for F2 = $(F2(L1) - F2(OL))/|F2(OL)|$

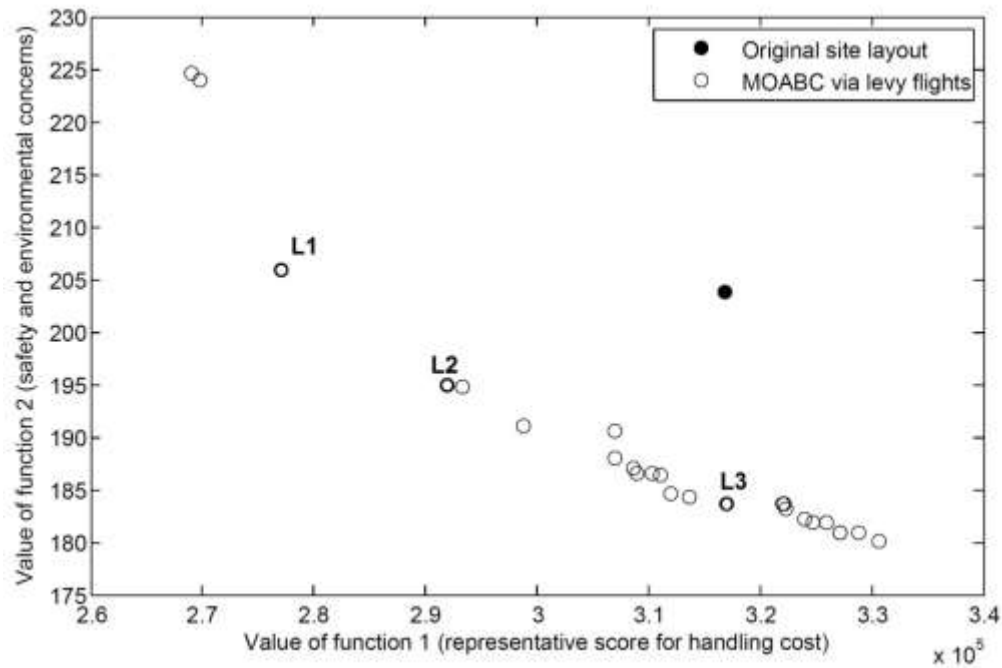


Figure 5-22 Pareto front for the CSLP problem of design example 2

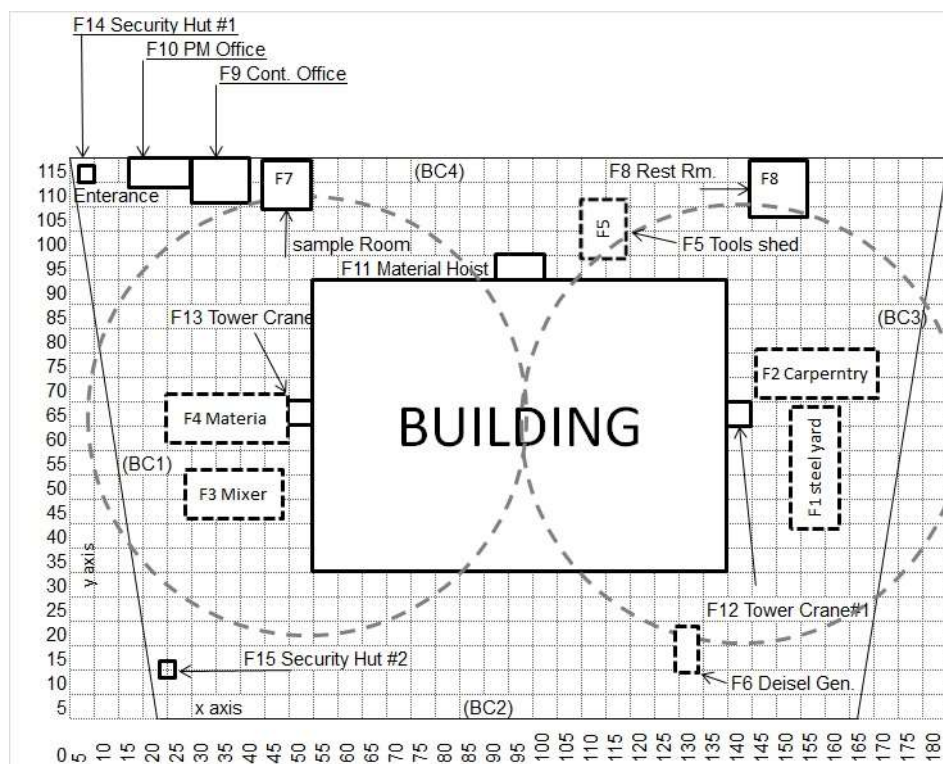


Figure 5-23 Construction site layout alternative L2 for case study 2

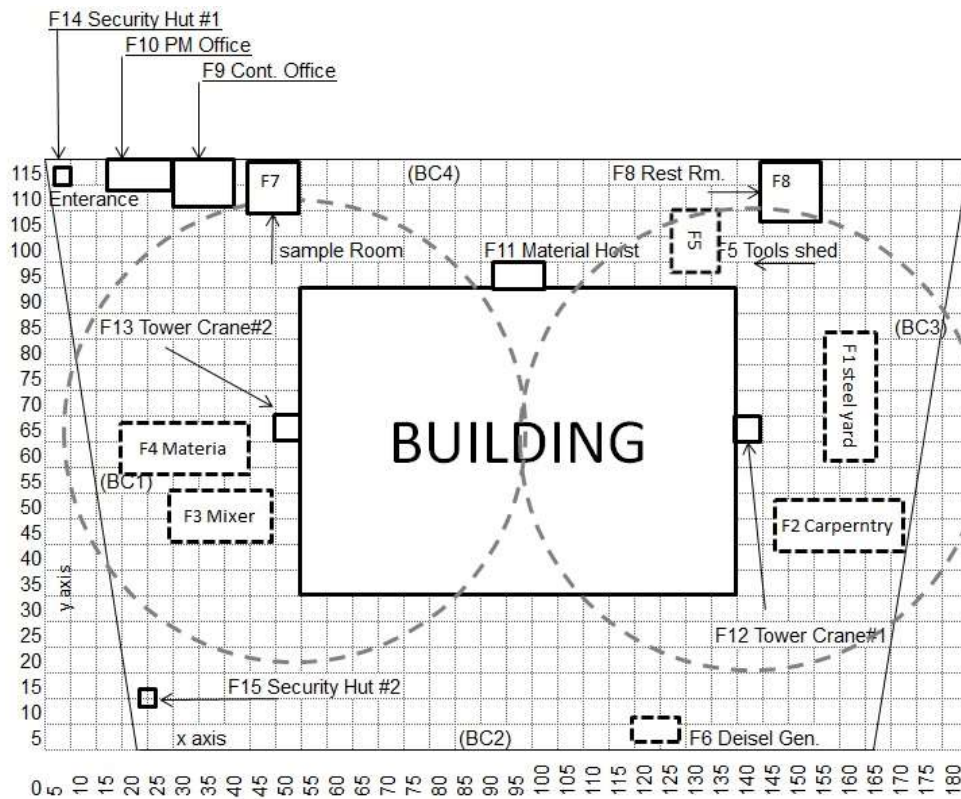


Figure 5-24 Construction site layout alternative L3 for case study 2

Chapter 6 Conclusion

In this thesis a multi objective decision making model for dynamic construction site layout planning problem is proposed which makes use of novel MOABC via Levy flights algorithm. The performance of the proposed MOABC via levy flights algorithm is verified by considering mathematical benchmark problems as well as benchmark engineering design problems such as cantilever beam and disk break design problems taken from the literature. The comparison of the results obtained by the proposed method with those attained by the most recent meta-heuristic algorithms revealed the efficiency of the proposed algorithm in solving multi-objective optimization problems. The results have shown that the proposed algorithm outperforms those algorithms considered in this study in solving the considered benchmark problems. For optimizing the construction site layout problem, the CSLP model based on multi objective artificial bee colony via levy flights (MOABC via Levy flights) is compared with Basic-MOABC model, max-min Ant system (MMAS) model, and the original construction site layout of the studied problem. Results show that MOABC via Levy Flights performs better than the other two algorithms. Using levy flights by employed bees has improved the efficiency of MOABC algorithm as it contains balanced combination of a local random search of onlooker bees and the global explorative random search of employed bees. The main contributions of this thesis are as follows: firstly, alternatives generated by the proposed CSLP model are more practical due to its feature of measuring travelling distances which represent the actual situation more realistically considering obstructions. Secondly, the proposed algorithm allows vertical and horizontal alignment of temporary facilities. Thirdly, it considers CSLP problem as continuous search problem

rather than simplified assignment problem. Finally, it gives the option to maintain minimum distances between the facilities as well as between facilities and buildings. The proposed model and the algorithm are successfully applied to two practical cases and proved to be robust and efficient.

Appendices

Appendix A Facilities closeness relationship matrices for case study 1

Table A. 1 The facility relationship weights representing total handling cost in phase 1

	F2	F3	F4	F5	F6	F7	F8	F9	F10	F15	F16	F17
F2	0	7.23	7.07	9	9.52	22.96	11.45	7.88	6.73	11.92	17.02	17.02
F3	7.23	0	13.05	9.99	13.15	10.57	10.57	10.98	10.98	13.59	15.91	15.91
F4	7.07	13.05	0	10.98	11.92	9.16	12.77	17.78	17.78	12.41	11.37	11.37
F5	9	9.99	10.98	0	9.66	20.86	20.86	12.77	16.55	11.92	12.41	12.41
F6	9.52	13.15	11.92	9.66	0	20.86	20.86	13.15	12.41	11.37	9.66	9.66
F7	22.96	10.57	9.16	20.86	20.86	0	6.39	13.59	11.37	8.01	13.15	13.15
F8	11.45	10.57	12.77	20.86	20.86	6.39	0	13.59	15.91	9.99	13.15	13.15
F9	7.88	10.98	17.78	12.77	13.15	13.59	13.59	0	13.59	10.54	11	11
F10	6.73	10.98	17.78	16.55	12.41	11.37	15.91	13.59	0	15.92	16.26	16.26
F15	11.92	13.59	12.41	11.92	11.37	8.01	9.99	10.54	15.92	0	18.59	18.59
F16	17.02	15.91	11.37	12.41	9.66	13.15	13.15	11	16.26	18.59	0	14.59
F17	17.02	15.91	11.37	12.41	9.66	13.15	13.15	11	16.26	18.59	14.59	0

Table A. 2 The facility relationship weights representing safety concerns in phase 1

	F2	F3	F4	F5	F6	F7	F8	F9	F10	F15	F16	F17
F2	0	4.95	4.95	6.75	14.85	24.3	13.5	5.85	2.1	16.2	20.25	20.25
F3	4.95	0	17.55	2.1	16.2	8.1	8.1	6.3	6.3	5.4	14.85	14.85
F4	4.95	17.55	0	18.9	16.2	1.8	14.85	7.65	7.65	16.2	14.85	14.85
F5	6.75	2.1	18.9	0	18.9	1.65	1.65	14.85	2.1	16.2	16.2	16.2
F6	14.85	16.2	16.2	18.9	0	1.65	1.65	16.2	16.2	14.85	18.9	18.9
F7	24.3	8.1	1.8	1.65	1.65	0	8.1	16.2	14.85	6.3	5.4	5.4
F8	13.5	8.1	14.85	1.65	1.65	8.1	0	16.2	14.85	2.1	5.4	5.4
F9	5.85	6.3	7.65	14.85	16.2	16.2	16.2	0	16.2	6.75	7.65	7.65
F10	2.1	6.3	7.65	2.1	16.2	14.85	14.85	16.2	0	16.2	16.2	16.2
F15	16.2	5.4	16.2	16.2	14.85	6.3	2.1	6.75	16.2	0	7.65	7.65
F16	20.25	14.85	14.85	16.2	18.9	5.4	5.4	7.65	16.2	7.65	0	5.85
F17	20.25	14.85	14.85	16.2	18.9	5.4	5.4	7.65	16.2	7.65	5.85	0

Table A. 3 The facility relationship weights representing total handling cost in phase 2

	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15	F16	F17
F2	0	15.4	18.2	16.1	10.3	22.4	15.4	18.4	15.2	26.5	13.3	15	15.2	13.4	13.1	13.1
F3	15.4	0	14.5	21	14.0	10.5	10.5	20.6	21	21.1	18.5	17.7	23.1	9.67	10.6	10.6
F4	18.2	14.5	0	13.4	11.9	10.6	11	17.7	17.7	10.5	10.5	10.5	10.5	8.2	11.9	11.9
F5	16.1	21	13.4	0	12.4	20.8	20.8	12.1	13.1	13.7	15.3	15.2	15.9	12.8	10.2	10.2
F6	10.3	14.0	11.9	12.4	0	16.9	16.9	8.2	11.9	13.0	9.67	10.6	8.2	21	12.6	12.6
F7	22.4	10.5	10.6	20.8	16.9	0	6.39	21.1	21.1	21.1	21.1	21.1	21.1	13.4	13.1	13.1
F8	15.4	10.5	11	20.8	16.9	6.39	0	21.1	21.1	21.1	21.1	21.1	21.1	11	14.0	14.0
F9	18.4	20.6	17.7	12.1	8.2	21.1	21.1	0	21	21	21	21	21	13.4	13.1	13.1
F10	15.2	21	17.7	13.1	11.9	21.1	21.1	21	0	21	21	21	21	15.9	17.1	17.1
F11	26.5	21.1	10.5	13.7	13.0	21.1	21.1	21	21	0	21	21	21	13.1	17.5	17.5
F12	13.3	18.5	10.5	15.3	9.67	21.1	21.1	21	21	21	0	21	21	13.4	10.3	10.3
F13	15	17.7	10.5	15.2	10.6	21.1	21.1	21	21	21	21	0	21	11.1	14.0	14.0
F14	15.2	23.1	10.5	15.9	8.2	21.1	21.1	21	21	21	21	21	0	12.0	10.3	10.3
F15	13.4	9.67	8.2	12.8	21	13.4	11	13.4	15.9	13.1	13.4	11.1	12.0	0	18.5	18.5
F16	13.1	10.6	11.9	10.2	12.6	13.1	14.0	13.1	17.1	17.5	10.3	14.0	10.3	18.5	0	14.3
F17	13.1	10.6	11.9	10.2	12.6	13.1	14.0	13.1	17.1	17.5	10.3	14.0	10.3	18.5	14.3	0

Table A. 4 The facility relationship weights representing safety concerns in phase 2

	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15	F16	F17
F2	0	21.6	7.65	4.95	5.4	8.55	21.6	22.95	14.85	5.4	9	7.2	7.65	14.85	16.2	16.2
F3	21.6	0	25.65	7.2	22.95	8.1	8.1	8.55	7.2	7.65	4.95	1.8	9	21.6	22.95	22.95
F4	7.65	25.65	0	14.85	16.2	8.55	21.6	7.65	7.65	5.85	5.85	5.85	5.85	14.85	16.2	16.2
F5	4.95	7.2	14.85	0	27	1.65	1.65	1.65	1.8	8.55	7.2	7.65	4.95	16.2	25.65	25.65
F6	5.4	22.95	16.2	27	0	1.65	1.65	14.85	16.2	25.65	21.6	22.95	14.85	81	25.65	25.65
F7	8.55	8.1	8.55	1.65	1.65	0	8.1	7.65	7.65	7.65	7.65	7.65	7.65	14.85	5.4	5.4
F8	21.6	8.1	21.6	1.65	1.65	8.1	0	7.65	7.65	7.65	7.65	7.65	7.65	7.2	22.95	22.95
F9	22.95	8.55	7.65	1.65	14.85	7.65	7.65	0	81	81	81	81	81	14.85	5.4	5.4
F10	14.85	7.2	7.65	1.8	16.2	7.65	7.65	81	0	81	81	81	81	16.2	27	27
F11	5.4	7.65	5.85	8.55	25.65	7.65	7.65	81	81	0	81	81	81	5.4	25.65	25.65
F12	9	4.95	5.85	7.2	21.6	7.65	7.65	81	81	81	0	81	81	14.85	5.4	5.4
F13	7.2	1.8	5.85	7.65	22.95	7.65	7.65	81	81	81	81	0	81	21.6	22.95	22.95
F14	7.65	9	5.85	4.95	14.85	7.65	7.65	81	81	81	81	81	0	4.95	1.8	1.8
F15	14.85	21.6	14.85	16.2	81	14.85	7.2	14.85	16.2	5.4	14.85	21.6	4.95	0	7.65	7.65
F16	16.2	22.95	16.2	25.65	25.65	5.4	22.95	5.4	27	25.65	5.4	22.95	1.8	7.65	0	6.3
F17	16.2	22.95	16.2	25.65	25.65	5.4	22.95	5.4	27	25.65	5.4	22.95	1.8	7.65	6.3	0

Table A. 5 The facility relationship weights representing total handling cost in phase 3

	F1	F2	F3	F4	F6	F7	F8	F11	F12	F13	F14	F15	F16	F17
F1	0	8.54	17.16	10.05	17.16	17.16	17.16	14.1	17.53	17	17.9	14.1	14.1	14.1
F2	8.54	0	15.31	15.78	24.7	17.16	17.16	13.32	14.87	15	15	14.1	18.48	18.48
F3	17.16	15.31	0	9.52	17.16	10.57	10.57	8.23	10	18	18	13.67	5.9	5.9
F4	10.05	15.78	9.52	0	11.92	17.16	17.16	10.55	10.55	10.55	10.55	13.67	5.94	5.94
F6	17.16	24.7	17.16	11.92	0	20.86	20.86	17.16	17.16	17.16	17.16	17.16	17.16	17.16
F7	17.16	17.16	10.57	17.16	20.86	0	6.39	21.18	21.18	21.18	21.18	17.16	17.16	17.16
F8	17.16	17.16	10.57	17.16	20.86	6.39	0	21.18	21.18	21.18	21.18	14.23	14.41	14.41
F11	14.1	13.32	8.23	10.55	17.16	21.18	21.18	0	21	21	21	14.23	17.4	17.4
F12	17.53	14.87	10	10.55	17.16	21.18	21.18	21	0	21	21	14.23	17.4	17.4
F13	17	15	18	10.55	17.16	21.18	21.18	21	21	0	21	14.23	17.4	17.4
F14	17.9	15	18	10.55	17.16	21.18	21.18	21	21	21	0	14.23	17.4	17.4
F15	14.1	14.1	13.67	13.67	17.16	17.16	14.23	14.23	14.23	14.23	14.23	0	18.59	18.59
F16	14.1	18.48	5.9	5.94	17.16	17.16	14.41	17.4	17.4	17.4	17.4	18.59	0	18.32
F17	14.1	18.48	5.9	5.94	17.16	17.16	14.41	17.4	17.4	17.4	17.4	18.59	18.32	0

Table A. 6 The facility relationship weights representing safety concerns in phase 3

	F1	F2	F3	F4	F6	F7	F8	F11	F12	F13	F14	F15	F16	F17
F1	0	2.1	22.95	6.75	22.95	22.95	22.95	7.65	6.75	5.4	6.3	7.65	7.65	7.65
F2	2.1	0	22.95	2.4	24.3	22.95	22.95	4.95	2.1	22.95	22.95	22.95	20.25	20.25
F3	22.95	22.95	0	16.2	22.95	8.1	8.1	6.3	22.95	22.95	22.95	6.75	5.85	5.85
F4	6.75	2.4	16.2	0	16.2	22.95	22.95	5.85	5.85	5.85	5.85	6.75	5.85	5.85
F6	22.95	24.3	22.95	16.2	0	1.65	1.65	22.95	22.95	22.95	22.95	22.95	22.95	22.95
F7	22.95	22.95	8.1	22.95	1.65	0	8.1	7.65	7.65	7.65	7.65	22.95	22.95	22.95
F8	22.95	22.95	8.1	22.95	1.65	8.1	0	7.65	7.65	7.65	7.65	16.2	18.9	18.9
F11	7.65	4.95	6.3	5.85	22.95	7.65	7.65	0	81	81	81	16.2	44.55	44.55
F12	6.75	2.1	22.95	5.85	22.95	7.65	7.65	81	0	81	81	16.2	44.55	44.55
F13	5.4	22.95	22.95	5.85	22.95	7.65	7.65	81	81	0	81	16.2	44.55	44.55
F14	6.3	22.95	22.95	5.85	22.95	7.65	7.65	81	81	81	0	16.2	44.55	44.55
F15	7.65	22.95	6.75	6.75	22.95	22.95	16.2	16.2	16.2	16.2	16.2	0	7.65	7.65
F16	7.65	20.25	5.85	5.85	22.95	22.95	18.9	44.55	44.55	44.55	44.55	7.65	0	6.3
F17	7.65	20.25	5.85	5.85	22.95	22.95	18.9	44.55	44.55	44.55	44.55	7.65	6.3	0

Appendix B Facilities closeness relationship matrices for case study 2

Note: Tables of Appendix B are obtained by converting the relationships of Tables 5-5, 5-6 and other relationship data obtained from the project team into weights using Table 3-1.

Table B. 1 The facility relationship weights representing total handling cost in phase 1

	F1	F2	F3	F4	F5	F6	F8	F9	F10	F12	F13	F14	F15
F1	0	81	1	1	9	27	9	3	1	81	0	9	9
F2	81	0	0	1	9	9	9	3	1	81	0	9	9
F3	1	0	0	81	3	9	3	3	1	0	81	9	9
F4	1	1	81	0	3	1	1	3	1	0	81	27	27
F5	9	9	3	3	0	1	1	1	1	3	3	1	1
F6	27	9	9	1	1	0	3	3	3	27	27	1	1
F8	9	9	3	1	1	3	0	1	1	1	1	1	1
F9	3	3	3	3	1	3	1	0	81	0	0	3	3
F10	1	1	1	1	1	3	1	81	0	0	0	3	3
F12	81	81	0	0	3	27	1	0	0	0	9	1	1
F13	0	0	81	81	3	27	1	0	0	9	0	1	1
F14	9	9	9	27	1	1	1	3	3	1	1	0	3
F15	9	9	9	27	1	1	1	3	3	1	1	3	0

Table B. 2 The facility relationship weights representing safety concerns in phase 1

	F1	F2	F3	F4	F5	F6	F8	F9	F10	F12	F13	F14	F15
F1	0	9	9	9	9	81	81	81	81	3	3	27	27
F2	9	0	3	9	9	81	81	81	81	3	3	27	27
F3	9	3	0	3	3	9	27	81	81	3	3	9	9
F4	9	9	3	0	3	81	27	81	81	27	27	81	81
F5	9	9	3	3	0	9	9	9	9	1	1	3	3
F6	81	81	9	81	9	0	81	81	81	9	9	81	81
F8	81	81	27	27	9	81	0	1	1	9	9	81	81
F9	81	81	81	81	9	81	1	0	1	9	9	27	27
F10	81	81	81	81	9	81	1	1	0	9	9	27	27
F12	3	3	3	27	1	9	9	9	9	0	81	27	27
F13	3	3	3	27	1	9	9	9	9	81	0	27	27
F14	27	27	9	81	3	81	81	27	27	27	27	0	27
F15	27	27	9	81	3	81	81	27	27	27	27	27	0

Table B. 3 The facility relationship weights representing total handling cost in phase 2

	F1	F2	F3	F4	F5	F6	F8	F9	F10	F11	F12	F13	F14	F15
F1	0	81	1	3	27	27	9	3	1	1	81	0	9	9
F2	81	0	0	3	9	9	9	3	1	3	81	0	9	9
F3	1	0	0	81	3	9	3	3	1	9	0	81	9	9
F4	3	3	81	0	3	1	1	3	1	27	0	81	27	27
F5	27	9	3	3	0	1	1	1	1	9	3	3	1	1
F6	27	9	9	1	1	0	3	3	3	27	27	27	1	1
F8	9	9	3	1	1	3	0	1	1	27	1	1	1	1
F9	3	3	3	3	1	3	1	0	81	3	0	0	3	3
F10	1	1	1	1	1	3	1	81	0	3	0	0	3	3
F11	1	3	9	27	9	27	27	3	3	0	9	9	1	1
F12	81	81	0	0	3	27	1	0	0	9	0	9	1	1
F13	0	0	81	81	3	27	1	0	0	9	9	0	1	1
F14	9	9	9	27	1	1	1	3	3	1	1	1	0	3
F15	9	9	9	27	1	1	1	3	3	1	1	1	3	0

Table B. 4 The facility relationship weights representing safety concerns in phase 2

	F1	F2	F3	F4	F5	F6	F8	F9	F10	F11	F12	F13	F14	F15
F1	0	9	9	9	9	81	81	81	81	9	3	3	27	27
F2	9	0	3	9	9	81	81	81	81	9	3	3	27	27
F3	9	3	0	3	3	9	27	81	81	9	3	3	9	9
F4	9	9	3	0	3	81	27	81	81	9	27	27	81	81
F5	9	9	3	3	0	9	9	9	9	9	1	1	3	3
F6	81	81	9	81	9	0	81	81	81	9	9	9	81	81
F8	81	81	27	27	9	81	0	1	1	27	9	9	81	81
F9	81	81	81	81	9	81	1	0	1	81	9	9	27	27
F10	81	81	81	81	9	81	1	1	0	81	9	9	27	27
F11	9	9	9	9	9	9	27	81	81	0	27	27	27	27
F12	3	3	3	27	1	9	9	9	9	27	0	81	27	27
F13	3	3	3	27	1	9	9	9	9	27	81	0	27	27
F14	27	27	9	81	3	81	81	27	27	27	27	27	0	27
F15	27	27	9	81	3	81	81	27	27	27	27	27	27	0

Table B. 5 The facility relationship weights representing total handling cost in phase 3

	F2	F4	F5	F6	F7	F8	F9	F10	F11	F14	F15
F2	0	1	3	3	1	9	1	1	9	9	9
F4	1	0	1	1	9	3	3	1	81	27	27
F5	3	1	0	1	3	1	1	1	27	1	1
F6	3	1	1	0	3	3	3	3	27	1	1
F7	1	9	3	3	0	1	81	81	3	81	3
F8	9	3	1	3	1	0	1	1	27	1	1
F9	1	3	1	3	81	1	0	81	9	3	3
F10	1	1	1	3	81	1	81	0	9	3	3
F11	9	81	27	27	3	27	9	9	0	9	9
F14	9	27	1	1	81	1	3	3	9	0	3
F15	9	27	1	1	3	1	3	3	9	3	0

Table B. 6 The facility relationship weights representing safety concerns in phase 3

	F2	F4	F5	F6	F7	F8	F9	F10	F11	F14	F15
F2	0	9	27	81	1	81	81	81	9	27	27
F4	9	0	1	81	3	9	81	81	9	81	81
F5	27	1	0	9	1	9	9	9	9	3	3
F6	81	81	9	0	9	81	81	81	9	81	81
F7	1	3	1	9	0	1	1	1	1	1	1
F8	81	9	9	81	1	0	1	1	1	81	81
F9	81	81	9	81	1	1	0	1	9	1	1
F10	81	81	9	81	1	1	1	0	9	1	1
F11	9	9	9	9	1	1	9	9	0	9	9
F14	27	81	3	81	1	81	1	1	9	0	27
F15	27	81	3	81	1	81	1	1	9	27	0

Appendix C MOABC with Levy flights algorithm coded using MATLAB

language to solve CSLP problem of case study 2

C.1 MOABC with Levy flights algorithm

```
function [ParetoSol,f1,f2,Foods]= moabcLevy(NP)

if nargin<1,
    NP=26;%The number of colony size (employed bees+onlooker bees)
end

objfun ='RMH'; % Problem or function to be optimized
D=18; %The number of parameters of the problem to be optimized
%upper bound of the parameters.
ub= [180.*ones(1,2) 95.*ones(1,2) 180.*ones(1,2) 115.*ones(1,6) 0.51.*ones(1,6)];
%lower bounds of the parameters.
lb= [95.*ones(1,2) 0.*ones(1,4) 20.*ones(1,4) 0.*ones(1,2) 0.49.*ones(1,6)];

% Control Parameters of the algorithm
FoodNumber=NP/2; %The number of food sources equals the half of the colony size
maxCycle= 1500; %The number of cycles for foraging {a stopping criteria}
tol = 0.0001;
RM=0.8;
SPP= 0.1*NP*D;

limit= 0.1*NP*D; %A food source which could not be improved through "limit" trials is abandoned
by its employed bee

%Foods [FoodNumber][D];Foods is the population of food sources. Each row of
%Foods matrix is a vector holding D parameters to be optimized. The number of rows of Foods
matrix equals to the FoodNumber

%ObjVal[FoodNumber]; f is a vector holding objective function values associated with food sources
%Fitness[FoodNumber]; fitness is a vector holding fitness (quality) values associated with food
sources
%trial[FoodNumber]; trial is a vector holding trial numbers through which solutions can not be
improved
%prob[FoodNumber]; prob is a vector holding probabilities of food sources(solutions) to be chosen
%solution [D]; New solution (neighbour) produced by  $v_{ij}=x_{ij}+\phi_{ij}*(x_{kj}-x_{ij})$  j is
a randomly chosen parameter and k is a randomly chosen solution different from i
```

```

%ObjValSol; Objective function value of new solution
%FitnessSol; Fitness value of new solution
%neighbour, param2change; param2change corresponds to j, neighbour corresponds to k in equation
 $v_{ij}=x_{ij}+\phi_{ij}*(x_{kj}-x_{ij})$ 

% All food sources are initialized
%Variables are initialized in the range [lb,ub]

Range = repmat((ub-lb),[FoodNumber 1]);
Lower = repmat(lb, [FoodNumber 1]);
Foods = rand(FoodNumber,D) .* Range + Lower;

[ObjVal violation]=feval(objfun,Foods);
Fitness = calculateDomFit(ObjVal);

%reset trial counters
trial=zeros(1,FoodNumber);

nonDom = ones(1,FoodNumber);
for s = 1:FoodNumber
    for so = 1: FoodNumber
        if s ~= so
            if (Violation(s)<= tol)&(Violation(so)<= tol) %both feasible
                fDomFit = calculateDomFit( [ObjVal(:,s) ObjVal(:,so)]);
                nonDom(s) = (fDomFit(1)>=(fDomFit(2)))*nonDom(s);
            elseif (Violation(so) <= tol)&( violation(s) >= tol) %so feasible and s isnot
                nonDom(s) = 0;
            else
                % Among two infeasible solutions (violation[i] > 0, violation[j] > 0), the one
having smaller
                % constraint violation is preferred (violation[i] <
                % violation[j], solution i is dominant).}
                if (Violation(s)>= tol)&(Violation(s) >= tol) %both infeasible
                    nonDom(s) = (Violation(s)<= (Violation(so)))*nonDom(s);
                end;
            end;
        end;
    end;
end;

```

```

        end;
    end;
end;

end;

ParetoObjVal = ObjVal(:,nonDom == 1);
ParetoFitness = calculateDomFit(ParetoObjVal);
ParetoViolation = Violation(nonDom == 1) ;
ParetoSol = Foods(nonDom == 1,:) ;

iter=1;
while ((iter <= maxCycle))

    %%%%%%%%%% EMPLOYED BEE PHASE %%%%%%%%%%

    for i=1:(FoodNumber)

        %The parameter to be changed is determined randomly
        [ObjVal Violation]=feval(objfun,Foods);
        ic = fix(rand*(FoodNumber))+1;
        while(ic==i)
            ic=fix(rand*(FoodNumber))+1;
        end;

        sol= get_levy(Foods(i,:),Foods(ic,:),lb,ub);
        FoodsSol = Foods;
        FoodsSol(i,:) = sol;
        [ObjValSol ViolationSol] =feval(objfun,FoodsSol);

        % After producing a new food source (solution), MOABC-Levy flights algorithm makes a
        selection by applying Modified Deb' s rules

        if (Violation(i)<= tol)&(ViolationSol(i) <= tol) %both feasible
            fDomFit = calculateDomFit( [ObjVal(:,i) ObjValSol(:,i)]);
            if fDomFit(2) > fDomFit(1) %
                Foods(i,:)=sol;
            end
        end
    end
end

```

```

        objVal(:,i)=objValSol(:,i);
        violation(i)= ViolationSol(i);
        trial(i)=0;
    else
        trial(i)=trial(i)+1; %if the solution i can not be improved, increase its trial
counter
    end;
elseif (violation(i)>= tol)&(ViolationSol(i) <= tol) %new feasible
    Foods(i,:)=sol;
    objVal(:,i)=objValSol(:,i);
    violation(i)= ViolationSol(i);
    trial(i)=0;
else
    % Among two infeasible solutions (violation[i] > 0, violation[j] > 0), the one having
smaller
    % constraint violation is preferred (violation[i] < violation[j], solution i is
dominant).}
    trial(i)=trial(i)+1;
    if (Violation(i)>= tol)&(ViolationSol(i) >= tol) %both infeasible
        if (ViolationSol(i)<Violation(i))
            Foods(i,:)=sol;
            objVal(:,i)=objValSol(:,i);
            violation(i)= ViolationSol(i);
            trial(i)=0;
        elseif (ViolationSol(i)== Violation(i))
            fDomFit = calculateDomFit( [objVal(:,i) objValSol(:,i)]);
            if fDomFit(2) > fDomFit(1)
                Foods(i,:)=sol;
                objVal(:,i)=objValSol(:,i);
                violation(i)= ViolationSol(i);
                trial(i)=0;
            end;
        end;
    end;
end;
end;
end;
end;

```

```

Fitness = calculateDomFit2(ObjVal,violation,tol);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%CalculateProbabilities%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

InfeasibleIndex= find(Violation > tol);
FeasibleIndex= find(Violation <= tol);
prob=zeros(size(Violation));
prob(InfeasibleIndex)= (1 - (Violation(InfeasibleIndex)./ sum(Violation))).* 0.5;
prob(FeasibleIndex)= 0.5 + Fitness(FeasibleIndex)./ sum(Fitness).* 0.5;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% ONLOOKER BEE PHASE %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
t=0;
while (t <= FoodNumber)
    i=fix(rand*(FoodNumber))+1;
    probrand = rand;
    if probrand < prob(i)
        t = t + 1;
        %The parameters to be changed is determined randomly
        Rj=rand(1,D);
        Param2chIndL = find(Rj<RM);
        %A randomly chosen solution is used in producing a mutant solution of the solution i
        neighbour=fix(rand*(FoodNumber))+1;
        %Randomly selected solution must be different from the solution i
        while(neighbour==i)
            neighbour=fix(rand*(FoodNumber))+1;
        end;
        sol=Foods(i,:);
        %  $v_{ij}=x_{ij}+\phi_{ij}*(x_{kj}-x_{ij})$ 
        sol(Param2chIndL)=Foods(i,Param2chIndL)+(Foods(i,Param2chIndL)-
Foods(neighbour,Param2chIndL))*(rand-0.5)*2;

        %if generated parameter value is out of boundaries, it is shifted onto the boundaries
        sol=simplebounds(sol,lb,ub);

        %evaluate new solution

```

```

FoodsSol = Foods;
FoodsSol(i,:) = sol;
[ObjvalSol ViolationSol] = feval(objfun,FoodsSol);
FitnessSol=calculateDomFit(ObjvalSol);

% After producing a new food source (solution), MOABC-Levy flights algorithm makes a
selection by applying Modified Deb' s rules

if (Violation(i)<= tol)&(ViolationSol(i) <= tol) %both feasible
    fDomFit = calculateDomFit( [Objval(:,i) ObjvalSol(:,i)]);
    if fDomFit(2) > fDomFit(1)
        Foods(i,:)=sol;
        Fitness(i)=FitnessSol(i);
        Objval(:,i)=ObjvalSol(:,i);
        Violation(i)= ViolationSol(i);
        trial(i)=0;
    else
        trial(i)=trial(i)+1; %if the solution i can not be improved, increase its
trial counter
    end;
elseif (Violation(i)>= tol)&(ViolationSol(i) <= tol) %new feasible
    Foods(i,:)=sol;
    Fitness(i)=FitnessSol(i);
    Objval(:,i)=ObjvalSol(:,i);
    Violation(i)= ViolationSol(i);
    trial(i)=0;
else
    % Among two infeasible solutions (violation[i] > 0, violation[j] > 0), the one
having smaller
    % constraint violation is preferred (violation[i] < violation[j], solution i is
dominant).}
    trial(i)=trial(i)+1;
    if (Violation(i)>= tol)&(ViolationSol(i) >= tol) %both infeasible
        if (ViolationSol(i)<Violation(i))
            Foods(i,:)=sol;
            Fitness(i)=FitnessSol(i);
            Objval(:,i)=ObjvalSol(:,i);

```



```

end

Arch0 = [];
ArchV = [];
ArchS = [] ;
for s = 1:size(ArchObjVal,2)
    if ~isnan(ArchObjVal(:,s))
        Arch0 = [Arch0 ArchObjVal(:,s)];
        ArchV = [ArchV ArchViolation(:,s)];
        ArchS = [ArchS; ArchSol(s,:)];
    end
end

ArchObjVal = Arch0;
ArchViolation = ArchV;
ArchSol = ArchS ;
ArchFitness = calculateDomFit(ArchObjVal);
ArchNumber = size(ArchViolation,2);
ParetoObjVal =[];
ParetoFitness = [];
ParetoViolation = [];
ParetoSol = [];
nonDom = ones(1,ArchNumber);
for s = 1:ArchNumber
    for so = 1: ArchNumber
        if s ~= so
            if (ArchViolation(s)<= tol)&(ArchViolation(so)<= tol) %both feasible
                fDomFit = calculateDomFit( [ArchObjVal(:,s) ArchObjVal(:,so)]);
                nonDom(s) = (fDomFit(1)>=(fDomFit(2)))*nonDom(s);
            elseif (ArchViolation(so) <= tol)&( ArchViolation(s) >= tol) %so feasible and s
isnot
                nonDom(s) = 0;
            else
                % Among two infeasible solutions (violation[i] > 0, violation[j] > 0), the
one having smaller

```

```

is dominant).}      % constraint violation is preferred (violation[i] < violation[j], solution i
                    if (ArchViolation(s)>= tol)&(ArchViolation(s) >= tol) %both infeasible
                        nonDom(s) = (ArchViolation(s)<= (ArchViolation(so)))*nonDom(s);
                    end;
                end;
            end;
        end;
    end;

ParetoObjVal = ArchObjVal(:,nonDom == 1) ;
ParetoFitness = calculateDomFit(ParetoObjVal);
ParetoViolation = ArchViolation(nonDom == 1);
ParetoSol = ArchSol(nonDom == 1,:);

%%%%%%%%%% SCOUT BEE PHASE %%%%%%%%%%%
%determine the food sources whose trial counter exceeds the "limit" value.
if (mod(iter,SPP)==0)
    RFoods = rand(FoodNumber,D) .* Range + Lower;%empty_nests(Foods,lb,ub);
    Foods(trial>limit)= RFoods(trial>limit);
    trial(trial>limit)=0;
end;
end; % End of MOABC-Levy flights

f1 = ParetoObjVal(1,:);
f2 = ParetoObjVal(2,:);

plot(f1,f2,'d')

```

C.2 Get levy flights

```

function sn = get_levy(si,sj,Lb,Ub,t)
beta=3/2;
sigma=(gamma(1+beta)*sin(pi*beta/2)/(gamma((1+beta)/2)*beta*2^((beta-1)/2)))^(1/beta);
alphao =2.*(rand(size(si)) - 0.5);

```

```

alpha = alphao.*(sj - si);
% Levy flights by Mantegna's algorithm
u=randn(size(si))*sigma;
v=randn(size(si));
step=u./abs(v).^(1/beta);
stepsize= step;
sn=si +alpha.* stepsize;
% Apply simple bounds/limits
sn = simplebounds(sn,Lb,Ub);

```

C.3 Application of simple constraints

```

function s=simplebounds(s,Lb,Ub)
% Apply the lower bound
ns_tmp=s;
rLb = repmat(Lb,size(s,1),1);
I=ns_tmp<rLb;
ns_tmp(I)=rLb(I);

% Apply the upper bounds
rUb = repmat(Ub,size(s,1),1);
J=ns_tmp>rUb;
ns_tmp(J)=rUb(J);
% Update this new move
s=ns_tmp;

```

C.4 Find the current Pareto

```

function
[ParetoObjVal,ParetoFitness,ParetoViolation,ParetoSol,nonDom]=getPareto(Foods,ObjVal,Violation,to
1)
FoodNumber = size(Violation,2);
nonDom = ones(1,FoodNumber);
for s = 1:FoodNumber

```



```

for j = 1:SN
    for jo = 1:SN
        if j~= jo
            fDom(j) = fDom(j) + min(fObjV(:,j)>=(fObjV(:,jo))).*
~(min(fObjV(:,j)==(fObjV(:,jo)))==1);
        end
    end
end

fDom = SN - fDom;
fDomFit = fDom/SN;

```

C.6 Calculate the fitness of feasible solution only

```

function fDomFit=calculateDomFit2(fObjV,violation,tol)

%this function calculate the fitness of feasible solutions only
% fDom(j) is number of food sources dominates foodsource j
Feasible=zeros(size(violation));
Feasible(violation <= tol)= 1;
nf = size(fObjV,1);
SN = size(fObjV,2);
fDom=zeros(1,SN);
for j = 1:SN
    for jo = 1:SN
        if j~= jo
            fDom(j) = fDom(j) + Feasible(jo)*min(fObjV(:,j)>=(fObjV(:,jo))).*
~(min(fObjV(:,j)==(fObjV(:,jo)))==1);
        end
    end
end

fDom = Feasible.*(SN - fDom);
fDomFit = fDom/SN;

```

C.7 CSLP model of case study 2

```

function [ObjVal fViolation]=RMH(Colony)

```

```

x1 = Colony(:,1);
x2 = Colony(:,2);
x3 = Colony(:,3);
x4 = Colony(:,4);
x5 = Colony(:,5);
x6 = Colony(:,6);

```

```

y1 = Colony(:,7);
y2 = Colony(:,8);
y3 = Colony(:,9);
y4 = Colony(:,10);
y5 = Colony(:,11);
y6 = Colony(:,12);

```

```

R1 = Colony(:,13);
R2 = Colony(:,14);
R3 = Colony(:,15);
R4 = Colony(:,16);
R5 = Colony(:,17);
R6 = Colony(:,18);

```

%Facilities closeness relationship matrices

```
fc1 =
```

```

[ 0    81    1    1    9    27    9    3    1    81    0    9    9
   81    0    0    1    9    9    9    3    1    81    0    9    9
   1    0    0   81    3    9    3    3    1    0   81    9    9
   1    1   81    0    3    1    1    3    1    0   81   27   27
   9    9    3    3    0    1    1    1    1    3    3    1    1
  27    9    9    1    1    0    3    3    3   27   27    1    1
   9    9    3    1    1    3    0    1    1    1    1    1    1
   3    3    3    3    1    3    1    0   81    0    0    3    3
   1    1    1    1    1    3    1   81    0    0    0    3    3
  81   81    0    0    3   27    1    0    0    0    9    1    1
   0    0   81   81    3   27    1    0    0    9    0    1    1

```

```

9 9 9 27 1 1 1 3 3 1 1 0 3
9 9 9 27 1 1 1 3 3 1 1 3 0];

```

fs1 =

```

[0 9 9 9 9 81 81 81 81 3 3 27 27
9 0 3 9 9 81 81 81 81 3 3 27 27
9 3 0 3 3 9 27 81 81 3 3 9 9
9 9 3 0 3 81 27 81 81 27 27 81 81
9 9 3 3 0 9 9 9 9 1 1 3 3
81 81 9 81 9 0 81 81 81 9 9 81 81
81 81 27 27 9 81 0 1 1 9 9 81 81
81 81 81 81 9 81 1 0 1 9 9 27 27
81 81 81 81 9 81 1 1 0 9 9 27 27
3 3 3 27 1 9 9 9 9 0 81 27 27
3 3 3 27 1 9 9 9 9 81 0 27 27
27 27 9 81 3 81 81 27 27 27 27 0 27
27 27 9 81 3 81 81 27 27 27 27 27 0
];

```

fc2 =

```

[0 81 1 3 27 27 9 3 1 1 81 0 9
9
81 0 0 3 9 9 9 3 1 3 81 0 9
9
1 0 0 81 3 9 3 3 1 9 0 81 9
9
3 3 81 0 3 1 1 3 1 27 0 81 27
27
27 9 3 3 0 1 1 1 1 9 3 3 1
1
27 9 9 1 1 0 3 3 3 27 27 27 1
1
9 9 3 1 1 3 0 1 1 27 1 1 1
1
3 3 3 3 3 1 3 1 0 81 3 0 0 3
3
1 1 1 1 1 3 1 81 0 3 0 0 3
3

```

```

1  3      9      27      9      27      27      3      3      0      9      9      1
  1
81 81      0      0      3      27      1      0      0      9      0      9      1
  1
0  0      81      81      3      27      1      0      0      9      9      0      1
  1
9  9      9      27      1      1      1      3      3      1      1      1      0
  3
9  9      9      27      1      1      1      3      3      1      1      1      3
  0 ];

fs2 =
[0      9      9      9      9      81      81      81      81      9      3      3      27
  27
  9      0      3      9      9      81      81      81      81      9      3      3      27
  27
  9      3      0      3      3      9      27      81      81      9      3      3      9
  9
  9      9      3      0      3      81      27      81      81      9      27      27      81
  81
  9      9      3      3      0      9      9      9      9      9      1      1      3
  3
  81 81      9      81      9      0      81      81      81      9      9      9      81
  81
  81 81      27      27      9      81      0      1      1      27      9      9      81
  81
  81 81      81      81      9      81      1      0      1      81      9      9      27
  27
  81 81      81      81      9      81      1      1      0      81      9      9      27
  27
  9      9      9      9      9      9      27      81      81      0      27      27      27
  27
  3      3      3      27      1      9      9      9      9      27      0      81      27
  27
  3      3      3      27      1      9      9      9      9      27      81      0      27
  27
  27 27      9      81      3      81      81      27      27      27      27      27      0
  27
  27 27      9      81      3      81      81      27      27      27      27      27      27
  0
];

fc3 =
[0      1      3      3      1      9      1      1      9      9      9

```

1	0	1	1	9	3	3	1	81	27	27
3	1	0	1	3	1	1	1	27	1	1
3	1	1	0	3	3	3	3	27	1	1
1	9	3	3	0	1	81	81	3	81	3
9	3	1	3	1	0	1	1	27	1	1
1	3	1	3	81	1	0	81	9	3	3
1	1	1	3	81	1	81	0	9	3	3
9	81	27	27	3	27	9	9	0	9	9
9	27	1	1	81	1	3	3	9	0	3
9	27	1	1	3	1	3	3	9	3	0];

fs3 =

[0	9	27	81	1	81	81	81	9	27	27
9	0	1	81	3	9	81	81	9	81	81
27	1	0	9	1	9	9	9	9	3	3
81	81	9	0	9	81	81	81	9	81	81
1	3	1	9	0	1	1	1	1	1	1
81	9	9	81	1	0	1	1	1	81	81
81	81	9	81	1	1	0	1	9	1	1
81	81	9	81	1	1	1	0	9	1	1
9	9	9	9	1	1	9	9	0	9	9
27	81	3	81	1	81	1	1	9	0	27
27	81	3	81	1	81	1	1	9	27	0];

%Building coordinates

XB = 92.5;

YB = 60;

BL = 60;%Building length(Vertical dimention)

BW = 85;%Buiding width(Horizontal dimention)

RTC1 = 45;%Reachable raduis for tower crane no.1

RTC2 = 45; %Reachable raduis for tower crane no.2

dv =

```

[0      0      0      0      0      0      0      0      0      0      0      0      0
      0      0      0      1      0      0      0      0      0      0      0      0      0
      0      0      0      0      0      0      0      0      0      0      0      0      0
      0      0      0      1      0      0      0      0      0      0      0      0      0
      0      0      0      0      0      0      0      0      0      0      0      0      0
      0      0      0      1      0      0      0      0      0      0      0      0      0
      0      0      0      0      0      0      0      0      0      0      0      0      0
      0      0      0      1      0      0      0      0      0      0      0      0      0
      0      0      0      0      0      0      0      0      0      0      0      0      0
      0      0      0      0      0      0      0      0      0      0      0      0      0
      0      0      0      0      0      0      0      0      0      0      0      0      0
      0      0      0      0      0      0      0      0      0      0      0      0      0
      0      0      0      0      0      0      0      0      0      0      0      0      0
      0      0      0      0      0      0      0      0      0      0      0      0      0
      0      0      0      0      0      0      0      0      0      0      0      0      0
      0      0      0      0      0      0      0      0      0      0      0      0      0
      0      0      0      0      0      0      0      0      0      0      0      0      0
      0      0      0      0      0      0      0      0      0      0      0      0      0
      0      0      0      0      0      0      0      0      0      0      0      0      0
      0      0      0      0      0      0      0      0      0      0      0      0      0
      1      1      1      1      1      1      1      0      0      0      0      0      0
      0      0      0      0 ];

```

```
dv = 3.*dv;
```

```
dh = dv;
```

```
xM = [x1 x2 x3 x4 x5 x6];
```

```
xx7 = 45.*ones((size(Colony,1)),1);
```

```
xx8 = 145.*ones((size(Colony,1)),1);
```

```
xx9 = 31.*ones((size(Colony,1)),1);
```

```
xx10 = 19.*ones((size(Colony,1)),1);
```

```
xx11 = 92.5.*ones((size(Colony,1)),1);
```

```
xx12 = 142.5.*ones((size(Colony,1)),1);
```

```
xx13 = 47.5.*ones((size(Colony,1)),1);
```

```

xx14 = 3.5.*ones((size(Colony,1)),1);
xx15 = 20.*ones((size(Colony,1)),1);
xxB = XB.*ones((size(Colony,1)),1);
XM = [XM xx7 xx8 xx9 xx10 xx11 xx12 xx13 xx14 xx15 xxB];
YM = [y1 y2 y3 y4 y5 y6];
yy7 = 110.*ones((size(Colony,1)),1);
yy8 = 108.75.*ones((size(Colony,1)),1);
yy9 = 110.5.*ones((size(Colony,1)),1);
yy10 = 112.*ones((size(Colony,1)),1);
yy11 = 92.5.*ones((size(Colony,1)),1);
yy12 = 62.5.*ones((size(Colony,1)),1);
yy13 = 62.5.*ones((size(Colony,1)),1);
yy14 = 111.5.*ones((size(Colony,1)),1);
yy15 = 10.*ones((size(Colony,1)),1);
yyB = YB.*ones((size(Colony,1)),1);
YM = [YM yy7 yy8 yy9 yy10 yy11 yy12 yy13 yy14 yy15 yyB];
FLengthST = [25 25 20 25 12 10 10 12.5 9 6 5 5 5 3 3 BL];
FWidthST = [12 12 12 12 9 4 10 12.5 12 12 10 5 5 3 3 BW];
FLengthST = repmat(FLengthST,(size(Colony,1)),1);
FWidthST = repmat(FWidthST,(size(Colony,1)),1);
FLengthM = FLengthST;
FWidthM = FWidthST;
R = [R1 R2 R3 R4 R5 R6];

ph1 = [1 1 1 1 1 1 0 1 1 1 0 1 1 1 1];
ph2 = [1 1 1 1 1 1 0 1 1 1 1 1 1 1 1];
ph3 = [0 1 0 1 1 1 1 1 1 1 0 0 1 1 1];
R = round(R);
FLengthM(R==1) = FWidthST(R==1);
FWidthM(R==1) = FLengthST(R==1);
v=zeros((size(Colony,1)),1);
f1 = 0;
f2 = 0;
s = zeros(size(Colony,1),2);

```

```

CM = zeros(size(Colony,1),1);
for k = 1:size(Colony,1)
    f11 = 0;f12 = 0; f13 = 0;
    f21 = 0;f22 = 0; f23 = 0;
    FLength = FLengthM(k,:);
    Fwidth = FwidthM(k,:);
    FLength1 = FLength(ph1 == 1);
    Fwidth1 = Fwidth(ph1 == 1);
    FLength2 = FLength(ph2 == 1);
    Fwidth2 = Fwidth(ph2 == 1);
    FLength3 = FLength(ph3 == 1);
    Fwidth3 = Fwidth(ph3 == 1);
    X = XM(k,:);
    Y = YM(k,:);
    X1 = X(ph1 == 1);
    Y1 = Y(ph1 == 1);
    X2 = X(ph2 == 1);
    Y2 = Y(ph2 == 1);
    X3 = X(ph3 == 1);
    Y3 = Y(ph3 == 1);
    %Objective function f1 for phase 1
    for i = 1:12
        for j = (i+1):13
            if i ~= j
                rdistance = (abs(X1(i) - X1(:,j)) + abs(Y1(:,i) - Y1(:,j)));
                odisy = zeros(1,14);
                odisx = zeros(1,14);
                for o = 1:14
                    if (o ~= i && o ~= j)
                        deltax = X1 - X1(o);
                        deltaY= Y1 - Y1(o);
                        if ((abs(deltaY(i))< (FLength1(o)/2))&&(abs(deltaY(j))< (FLength1(o)/2)))
                            if ((deltax(i)*(deltax(j)))<0)

```

```

                                odisy(o) = 2*(FLength1(o)/2 -
max((abs(deltaY(i))), (abs(deltaY(j)))));
                                end

                                elseif ((abs(deltaX(i))< (Fwidth1(o)/2))&&(abs(deltaX(j))<
(Fwidth1(o)/2)))

                                if ((deltaY(i)*(deltaY(j)))<0)

                                odisx(o) = 2*(Fwidth1(o)/2 -
max((abs(deltaX(i))), (abs(deltaX(j)))));
                                end

                                end

                                end

                                end

                                end

                                rdistance = rdistance + max(odisy) + max(odisx);

                                f11 = f11 + fc1(i,j)*rdistance;

                                end

                                end

                                end

                                %Objective function 2 (safety) for phase 1

                                for i = 1:12

                                for j = (i+1):13

                                euclideanist = sqrt((X1(i) - X1(j)).^2 + (Y1(i) - Y1(j)).^2);

                                f21 = f21 + fs1(i,j)/euclideanist;

                                end

                                end

                                end

                                %Objective function f1 for phase 2

                                for i = 1:13

                                for j = (i+1):14

                                if i ~= j

                                rdistance = (abs(X2(i) - X2(:,j)) + abs(Y2(:,i) - Y2(:,j)));

                                odisy = zeros(1,15);

                                odisx = zeros(1,15);

                                for o = 1:15

                                if (o ~= i && o ~= j)

                                deltaX = X2 - X2(o);

                                deltaY= Y2 - Y2(o);

                                if ((abs(deltaY(i))< (FLength2(o)/2))&&(abs(deltaY(j))< (FLength2(o)/2)))

```

```

        if ((deltax(i)*(deltax(j)))<0)
            odisy(o) = 2*(FLength2(o)/2 -
max((abs(deltay(i))), (abs(deltay(j)))));
        end
        elseif ((abs(deltax(i))< (Fwidth2(o)/2))&&(abs(deltax(j))<
(Fwidth2(o)/2)))
            if ((deltay(i)*(deltay(j)))<0)
                odisx(o) = 2*(Fwidth2(o)/2 -
max((abs(deltax(i))), (abs(deltax(j)))));
            end
        end
    end
end
end
    rdistance = rdistance + max(odisy) + max(odisx);
    f12 = f12 + fc2(i,j)*rdistance;
end
end
end
%Objective function 2 (safety) for phase 2
for i = 1:13
    for j = (i+1):14
        euclideanist = sqrt((X2(i) - X2(j)).^2 + (Y2(i) - Y2(j)).^2);
        f22 = f22 + fs2(i,j)/euclideanist;
    end
end
end
%Objective function f1 for phase 3
for i = 1:10
    for j = (i+1):11
        if i ~= j
            rdistance = (abs(X3(i) - X3(:,j)) + abs(Y3(:,i) - Y3(:,j)));
            odisy = zeros(1,12);
            odisx = zeros(1,12);
            for o = 1:12
                if (o ~= i && o ~= j)
                    deltax = X3 - X3(o);
                    deltay= Y3 - Y3(o);

```

```

        if ((abs(deltaY(i))< (FLength3(o)/2))&&(abs(deltaY(j))< (FLength3(o)/2)))
            if ((deltaX(i)*deltaX(j))<0)
                odisy(o) = 2*(FLength3(o)/2 -
max((abs(deltaY(i))), (abs(deltaY(j))))) );
            end

elseif ((abs(deltaX(i))< (Fwidth3(o)/2))&&(abs(deltaX(j))<
(Fwidth3(o)/2)))
            if ((deltaY(i)*deltaY(j))<0)
                odisx(o) = 2*(Fwidth3(o)/2 -
max((abs(deltaX(i))), (abs(deltaX(j))))) );
            end
        end
    end
    end
    end
    rdistance = rdistance + max(odisy) + max(odisx);
    f13 = f13 + fc3(i,j)*rdistance;
end
end
end
%Objective function 2 (safety) for phase 3
for i = 1:10
    for j = (i+1):11
        euclideanist = sqrt((X3(i) - X3(j)).^2 + (Y3(i) - Y3(j)).^2);
        f23 = f23 + fs3(i,j)/euclideanist;
    end
end
%Total objective functions
S(k,:) = [(f11+f12+f13).*2 (f21+f22+f23).*2];
f11 = 0;f12 = 0; f13 = 0;
f21 = 0;f22 = 0; f23 = 0;
C = [];
r = 0;

%prevent overlapping
for i = 1:15

```

```

for j = (i+1):16
    r = r + 1;
    Cx(r) = 0.5.*((Fwidth(i) + Fwidth(j))) + dv(i,j)- abs(X(i) - X(j));
    Cy(r) = 0.5.*((FLength(i) + FLength(j))) + dh(i,j)- abs(Y(i) - Y(j)) ;
end
C = min(Cx,Cy);
C(C<0) = 0;
end
CB1 = zeros(1,6);CB2 = zeros(1,6);CB3 = zeros(1,6);CB4 = zeros(1,6);
%Boudary constrains
for i = 1:6
    CB1(i) = 115 - 6.571.*(X(i) - Fwidth(i)./2) - (Y(i) - FLength(i)./2);
    CB2(i) = 0 -(Y(i) - FLength(i)./2);
    CB3(i) = -1067.625 + 6.571.*(X(i) + Fwidth(i)./2) - (Y(i) - FLength(i)./2);
    CB4(i) = -115 +(Y(i) + FLength(i)./2);
    CB1(CB1<0) = 0;CB2(CB2<0) = 0;CB3(CB3<0) = 0;CB4(CB4<0) = 0;
end
%Tower crane constrains
%facility 1
COR1 = [(X(1) - Fwidth(1)/2) (Y(1) + FLength(1)/2);(X(1) - Fwidth(1)/2) (Y(1) - FLength(1)/2);(X(1) + Fwidth(1)/2) (Y(1) - FLength(1)/2);(X(1) + Fwidth(1)/2) (Y(1) + FLength(1)/2)];
for q = 1:4
    CT1TC1(q) = sqrt((COR1(q,1) - X(12)).^2 + ((COR1(q,2) - Y(12)).^2));
end
CT1 = max(CT1TC1)-RTC1;

%facility 2
COR2 = [(X(2) - Fwidth(2)/2) (Y(2) + FLength(2)/2);(X(2) - Fwidth(2)/2) (Y(2) - FLength(2)/2);(X(2) + Fwidth(2)/2) (Y(2) - FLength(2)/2);(X(2) + Fwidth(2)/2) (Y(2) + FLength(2)/2)];
for q = 1:4
    CT2TC1(q) = sqrt((COR2(q,1) - X(12)).^2 + ((COR2(q,2) - Y(12)).^2));
end
CT2 = max(CT2TC1)-RTC1;

```

```

%facility 3
COR3 = [(X(3) - Fwidth(3)/2) (Y(3) + FLength(3)/2);(X(3) - Fwidth(3)/2) (Y(3) -
FLength(3)/2);(X(3) + Fwidth(3)/2) (Y(3) - FLength(3)/2);(X(3) + Fwidth(3)/2) (Y(3) +
FLength(3)/2)];
for q = 1:4
    CT3TC2(q) = sqrt((COR3(q,1) - X(13)).^2 + ((COR3(q,2) - Y(13)).^2));
end
CT3 = max(CT3TC2)-RTC2;

%facility 4
COR4 = [(X(4) - Fwidth(4)/2) (Y(4) + FLength(4)/2);(X(4) - Fwidth(4)/2) (Y(4) -
FLength(4)/2);(X(4) + Fwidth(4)/2) (Y(4) - FLength(4)/2);(X(4) + Fwidth(4)/2) (Y(4) +
FLength(4)/2)];
for q = 1:4
    CT4TC2(q) = sqrt((COR4(q,1) - X(13)).^2 + ((COR4(q,2) - Y(13)).^2));
end
CT4 = max(CT4TC2)-RTC2;
CT1(CT1<0) = 0;CT2(CT2<0) = 0;CT3(CT3<0) = 0;CT4(CT4<0) = 0;

%Total Violation
CM(k) = sum(C)+ sum(CB1) + sum(CB2) + sum(CB3) + sum(CB4) + sum(CB1) + sum(CB2)+ sum(CB3)+
sum(CT1)+ sum(CT2)+ sum(CT3)+ sum(CT4);
end
V = CM;
ObjVal= S';
fviolation = V';

```

References

Akbari R., Hedayatzadeh R., Ziarati K., and Hassanizadeh B. (2011). "A multi-objective artificial bee colony algorithm." *Swarm Evol Comput*, 2, 39-52.

Askin R.G., Standridge C.R. (1993). "Modeling and Analysis of Manufacturing Systems", Wiley, New York.

Barth F. G. (1982). "Insects and Flowers: The Biology of a Partnership." N.J.: *Princeton University Press*, Princeton.

Cheung S.O., Tong T.K.L., and Tam C.M. (2002). "Site pre-cast yard layout arrangement through genetic algorithms" *Automation in Construction*, 11, 35–46.

Deb K. (2000). "An efficient constraint handling method for genetic algorithms" *Computer Methods in Applied Mechanics and Engineering*, 186, 311–338.

Deb K. (2001). "Multi-objective optimization using evolutionary algorithms." *John Wiley & Sons*, New York.

Elbeltagi E., Hagazy T. (2001). "A Hybrid AI-Based System for Site Layout Planning in Construction" *Computer-Aided Civil and Infrastructure Engineering*, 16, 79–93.

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

Frisch K. V. (1976) "Bees: Their Vision, Chemical Senses, and Language." *Cornell University Press*, Revised Edition, Ithaca, N.Y.

Hegazy T., and Elbeltagi E. (1999). "EvoSite - Evolution-Based Model for Site Layout Planning." *Journal of Computing in Civil Engineering*, 13(3), 198-206.

Holland, J., 1975. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI.

Jiuping X., and Zongmin L. (2012). "Multi-Objective Dynamic Construction Site Layout Planning in Fuzzy Random Environment." *Automation in Construction*, 27, 155–169.

Karaboga D. (2005). "An idea based on honey bee swarm for numerical optimization". Technical Report TR06, Erciyes University, Engineering Faculty, Computer Engineering Department.

Karaboga D., Akay B. (2011). "A modified artificial bee colony (abc) algorithm for constrained optimization problems." *Appl Soft Comput*, 1, 3021–3031.

- Karaboga D., and Akay B. (2009). "A comparative study of artificial bee colony algorithm." *Appl Math Comput*, 214, 108–132.
- Karaboga D., Gorkemli B., and Ozturk C. (2012). "A comprehensive survey: artificial bee colony (ABC) algorithm and applications" *Artif Intell Rev*, DOI 10.1007/s10462-012-9328-0.
- Kennedy, J., and Eberhart, R. C. _1995_. "Particle swarm optimization." *Proc., IEEE Conf. on Neural Networks, IV*, IEEE, Nagoya, Japan, 1942–1948.
- Khalafallah A., and El-Rayes K. (2011). "Automated multi-objective optimization system for airport site layouts" *Automation in Construction*, 20, 313–320.
- Lam K.C., Ning X., Ng T. (2007). "The application of the ant colony optimization algorithm to the construction site layout planning problem." *Construction Management and Economics*, 25 (4), 359–374.
- Li H., and Love P. (2000). "Genetic search for solving construction site-level unequal-area Construction site Layout problems" *Automation in Construction*, 9, 217-226.
- Li H., and Love P.E.D. (1998). "Site-lever facilities layout using genetic algorithms." *Journal of Computing in Civil Engineering*, 12 (4), 227–231.
- Lien L. C. and Cheng M. Y. (2012). "A hybrid swarm intelligence based particle-bee algorithm for construction site layout optimization" *Expert Systems with Applications*, 39, 9642–9650.
- Liggett R. S. (2000). "Automated facilities layout: past, present and future." *Automation in construction*, 9(2), 197-215.
- Mawdesley M., Al-jibouri S., and Yang H. (2002). "Genetic Algorithms for Construction Site Layout in Project Planning." *J. Constr. Eng. Manage.*, 128(5), 418–426
- Nakayama H. (2005). "Practical Approaches to Multi-Objective Optimization: Multi-objective Optimization and its Engineering Applications." *Dagstuhl Seminar Proceedings*, 04461, <http://drops.dagstuhl.de/opus/volltexte/2005/234>
- Ning X. (2008). "A Development of Dynamic Construction Site Layout Planning Decision-making System" *Ph.D. Thesis*, City University of Hong Kong.
- Ning X., Lamb K.C., and Lam M.C.K. (2010). "Dynamic construction site layout planning using max-min ant system." *Automation in Construction*, 19, 55-65.
- Ning X., Lamb K.C., and Lam M.C.K. (2011). "A decision-making system for construction site layout planning." *Automation in Construction*, 20, 459–473.

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*, 12, 749-764.

Ray L., and Liew K.M. (2002). "A swarm metaphor for multiobjective design optimization." *Engineering Optimization*, 34(2), 141–53.

Saka M. and Dogan P. E. (2012). "Recent Developments in Metaheuristic Algorithms, A Review" *Computational Technology Reviews*, 5, 31-78.

Samdani, S.A., Bhakal, L. and Singh, A.K. (2006). "Site layout of temporary construction facilities using ant colony optimization." *Los Angeles Section International Committee 4th International Engineering and Construction Conference, ASCE, California State University, USA*.

Sanad M., Ammar M. A., and Ibrahim M.E. (2008). "Optimal construction site layout considering safety and environmental aspects." *Journal of Construction Engineering and Management*, 134(7), 536-544.

Yang X. S. (2010a). "Nature-Inspired Metaheuristic Algorithms", Second Edition, *Luniver Press, Frome, UK*.

Yang X.S. (2010b). "Engineering Optimization: An introduction with meta heuristic applications." *John Wiley and Sons*.

Yang X.S., Deb S. (2009). "Cuckoo search via Levy flights." *Proceedings of world congress on nature & biologically inspired computing (NaBIC2009India)*, USA: IEEE Publications, 210–214.

Yang XS. (2013). "Multiobjective firefly algorithm for continuous optimization." *Engineering with Computers*, 29(2), 175-184.

Yang XS., and Deb S. (2013). "Multiobjective cuckoo search for design optimization." *Computers & Operations Research*, 40(6), 1616–1624

Yeh I.C. (1995). "Construction-site layout using annealed neural network." *Journal of Computing in Civil Engineering*, 9 (3), 201–208.

Zhang H., and Wang J.Y. (2008). "Particle swarm optimization for construction site unequal-area layout" *Journal of Construction Engineering and Management*, 134, 739-748.

Zou W., Zhu Y., Chen H. , and Zhang B. (2012). "Solving multiobjective optimization problems using artificial bee colony algorithm." *Discret Dyn Nat Soc*, doi:10.1155/2011/569784

Zouein P.P. and Tommelein I.D. (1994). "Automating Dynamic Layout Construction." *Automation and Robotics in Construction* , XI, 409-416

Zouein P.P., Harmanani H., and Hajar A. (2002). "Genetic algorithm for solving site layout problem with unequal size and constrained facilities." *Journal of Computing in Civil Engineering*, 16(2), 143-151.